ATEMATIČKI INSTITUT - BEOGRAD

Savremena računska tehnika i njena primena

Knjiga 1

NEDELJKO PAREZANOVIĆ

ALGORITMI
I
PROGRAMSKI JEZIK FORTRAN IV

Drugo neizmenjeno izdanje

BEOGRAD 1972

ATEMATIČKI INSTITUT - BEOGRAD

Savremena računska tehnika i njena primena

Knjiga 1

NEDELJKO PAREZANOVIĆ

ALGORITMI
I
PROGRAMSKI JEZIK FORTRAN IV

Drugo neizmenjeno izdanje

BEOGRAD 1972

Savremena računska tehnika nalazi sve širu primenu u različitim oblastima čovečije delatnosti. Ovo za sobom povlači potrebu upoznavanja širokog kruga ljudi sa načinom rada i mogućnostima računara, kao i sa matematičkim metodama za rešavanje složenih zadataka pomoću računara. Za ovo je neophodno postojanje odgovarajuće literature. I serija Matematičkog instituta

SAVREMENA RAČUNSKA TEHNIKA I NJENA PRIMENA

ima osnovni zadatak da na potrebnom teorijskom i praktičnom nivou upozna čitaoca sa dostignućima u ovoi oblasti.

Ova publikacija nije periodična.

Rukopise opremljene za štampu slati na adresu: Matematički institut, 11000 Beograd, Knez Mihailova 35.

Izdaje: Matematički institut — Beograd, Knez Mihailova 35

ŠTAMPA: ŠTAMPARIJA RADIO-TELEVIZIJE BEOGRAD
Batajnički put 24, telefon 607-073

MATEMATIČKI INSTITUT - BEOGRAD

Savremena računska tehnika i njena primena

Knjiga 1

NEDELJKO PAREZANOVIĆ

ALGORITMI I PROGRAMSKI JEZIK FORTRAN IV

Drugo neizmenjeno izdanje

BEOGRAD 1972

PREDGOVOR

Problem komunikacije izmedju čoveka i računara, spada u vrlo aktuelne probleme današnje računske tehnike. Ovaj problem najčešće se danas rešava uvodjenjem programskih jezika, koji su lako prihvatljivi od strane čoveka, a sa kojih je moguće izvršiti formalno prevodjenje na mašinski jezik. Kako je ovo prevodjenje formalizovano, to znači da može biti izvršeno od strane računara. Prema tome, programski jezik mora zadovoljiti dva osnovna uslova:

- da je što prihvatljiviji za čoveka, i
- da je tako definisan da se za prevodjenje na mašinski jezik može naći postupak pomoću kojeg ovo prevodjenje može izvršavati računar.

Od 1954. godine, kada su nastali prvi radovi u oblasti programskih jezika, pa do danas definisano je nekoliko stotina programskih jezika. Neki od ovih jezika usavršavani su i danas se koriste, a mnogi od njih pripadaju mrtvim jezicima. FORTRAN-jezik pripada prvoj grupi jezika. To je programski jezik razvijen u okviru američke firme IBM. Prva varijanta ovog jezika pojavila se 1954. godine, i od tada je ovaj jezik usavršavan, tako da se danas najviše koristi četvrta varijanta, tzv. FORTRAN IV. Danas 80% svih računara u svetu koristi FORTRAN kao programski jezik.

U ovoj knjizi izloženi su algoritmi i programski jezik FORTRAN IV. Karakteristike ovog izlaganja su:

- Algoritmi su izloženi sa aspekta algoritmizacije problema, pre njihovog prenošenja na računar. Detaljno su obradjene moguće algoritamske strukture kao i grafički način njihovog prikazivanja.

- Izlaganje FORTRAN-jezika ide od prostih prema složenim pojmovima koje sadrži jezik.

- Izložene su opšte važeće definicije FORTRAN-jezika, a samo tamo gde je to bilo potrebno navedene su specifičnosti primene na računaru IBM--360/44.

Autor se zahvaljuje M. Čavčiću na tehničkoj opremi materijala, kao i B. Živković na jezičkoj redakturi teksta.

1.03.1972. god.
Beograd

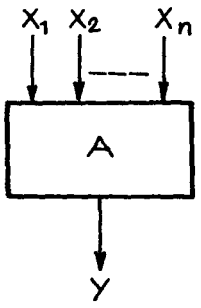
Autor

SADRŽAJ

1. ALGORITMI I NJIHOVE STRUKTURE

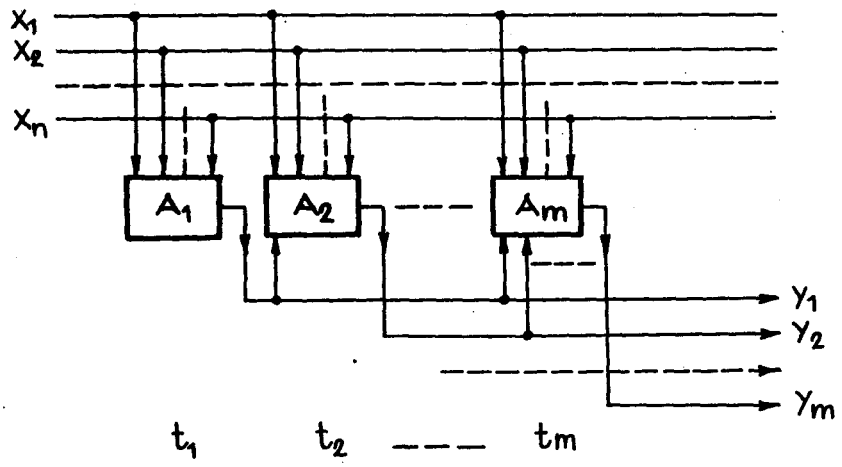
1.1. Algoritam i algoritamski korak

Pod algoritmom, u najopštijem smislu ove reči, podrazumevamo skup svih pravila formulisanih u cilju rešavanja odredjene vrste problema. Jasno je da se svi mi u svakodnevnom životu srećemo sa velikim brojem algoritama. Neke od ovih algoritama nesvesno izvršavamo, a mnoge od njih pamtimo i prema potrebi koristimo. Posebno matematika obiluje velikim brojem algoritama. Svaka definicija funkcije predstavlja u stvari algoritam za njeno izračunavanje, za zadate vrednosti argumenata. Ako ovako uopštenu definiciju algoritama primenimo na matematičke probleme, mogli bi reći da pod algoritmom podrazumevamo tačan opis računskog procesa, koji primenjen na izvesne brojne podatke dovodi do konačnih rezultata. Neka je skup polaznih podataka $x_1, x_2, \ldots, x_n$, i neka je zadat algoritam A koji ovaj skup podataka prevodi u rezultat y. Kaže se da skup x_i, $i = 1, 2, \ldots, n$ predstavlja ulazne podatke za algoritam A a broj y izlaznu vrednost algoritma. Ne ulazeći u to kakvim su aritmetičkim operacijama vezani brojni podaci x_i, $i = 1, 2, \ldots, n$ medju sobom u cilju dobijanja broja y, možemo grafički prikazati algoritam kao blok sa ulazima $x_i, i = 1, 2, \ldots, n$ i izlazom y (sl. 1.1.1). Izračunavanje veličine y može se sastojati iz niza aritmetičkih operacija, tako da se u toku primene algoritma A dolazi do medjurezultata $y_1, y_2, \ldots, y_m$. Drugim rečima, algoritam A se u suštini



Sl.1.1.1

može razbiti na niz algoritama A_i, i=1, 2, ... m, čiji su izlazi y_i, i=1, 2,..., m. Ulazi u algoritam A_p, mogu biti:proizvoljan broj elemenata skupa $\{x_i\}$, i = = 1, 2, . . . , n, kao i skupa $\{y_i\}$, i = 1, 2, . . . , k, gde je k < p (sl. 1.1.2). Izlaz iz algoritma A_m je y_m što ćemo uzeti da je i rezultat algoritma A, tj.



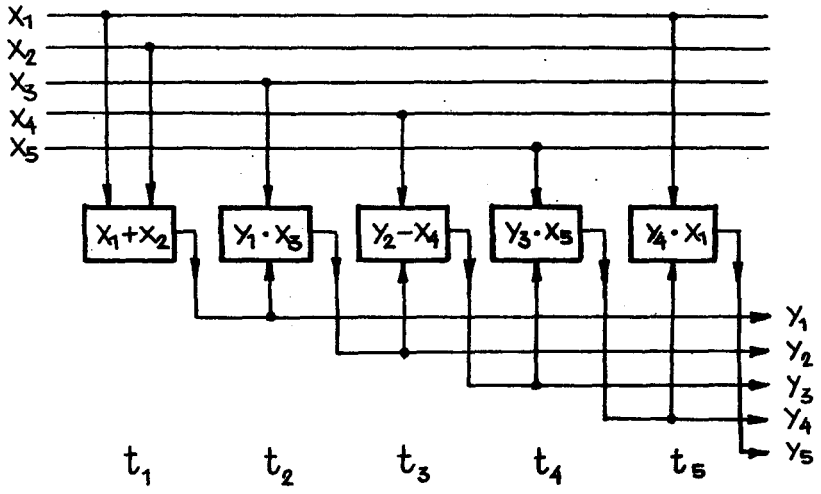
Sl. 1.1.2

$y = y_m$. Ako su algoritmi A_i, i = 1, 2, . . . , m dovoljno prosti u smislu potpunog razumevanja algoritma A kaže se da su A_i i = 1, 2, . . . , m koraci algoritma A. Prema tome, svaki algoritamski korak ima svoje ulaze i svoj izlaz, s tim što su ulazi u prvi algoritamski korak samo elementi iz skupa x_i, i = 1, 2, . . . , n, dok se u sve druge korake mogu na ulazu pojaviti i izlazi iz prethodnih koraka. Prema ovome, izgleda kao da je svaki sledeći korak u algoritmu sve složeniji jer se pojavljuje sve veći broj ulaza. Medjutim, u praktičnim primerima koraci se biraju tako da svaki od njih bude dovoljno prost, sa malim brojem ulaza, a izvršavanjem pojedinih koraka sve više se približavamo konačnom rezultatu algoritma, tako da se i broj ulaza u algoritamske korake najčešće smanjuje. Takodje treba imati u vidu da se kao rezultat algoritma i pojedinih algoritamskih koraka može pojaviti veći broj izlaznih veličina, dok smo mi radi prostijeg objašnjenja uzeli da postoji samo jedna izlazna veličina za svaki algoritamski korak, kao i algoritam u celini.

Radi ilustracije ovoga što smo do sada rekli o algoritmima, prosledimo algoritam izračunavanja funkcije

$$y = \left[(x_1 + x_2)\, x_3 - x_4\right] x_5 \cdot x_1 \qquad (1.1.1)$$

Algoritamski korak neka bude jedna aritmetička operacija (+, - ili x). Onda, izračunavanje vrednosti funkcije (1.1.1), za date vrednosti argumenata x_i,



Sl. 1.1.3

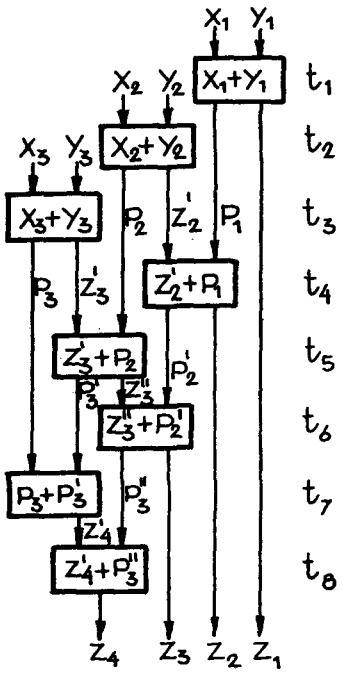
i = 1, 2, 3, 4, 5, može se raščlaniti na sledeće operacije:

$$\begin{aligned} y_1 &= x_1 + x_2 \\ y_2 &= y_1 \cdot x_3 \\ y_3 &= y_2 - x_4 \\ y_4 &= y_3 \cdot x_5 \\ y_5 &= y = y_4 \cdot x_1 \end{aligned} \qquad (1.1.2)$$

Niz operacija (1.1.2) može se grafički prikazati, što je učinjeno na slici 1.1.3.

U ovom primeru algoritamski korak je bio jedna od aritmetičkih operacija, i za svakog ko poznaje aritmetičke operacije sabiranja, oduzimanja

i množenja, algoritam na sl. 1.1.3, za izračunavanje funkcije y, je dovoljno jasno napisan. Izvršavanjem pojedinih algoritamskih koraka, u ovom slučaju aritmetičkih operacija, sa leva na desno (sl. 1.1.3) dolazimo do vrednosti funkcije y. Medjutim, ako ne bi poznavali kako se izvode pojedine aritmetičke operacije algoritam na sl. 1.1.3 ne bi bio jasno zapisan, jer ga praktično ne bi mogli koristiti. Tako, ako za trenutak zamislimo da ne znamo operaciju sabiranja višecifrenih brojeva, već samo sabiranje jednocifrenih dekadnih brojeva, onda je potrebno opisati algoritam sabiranja višecifrenih dekadnih brojeva. Zaista mi znamo napamet zbir dva jednocifrena dekadna broja, medjutim zbir dva višecifrena broja ne znamo napamet, ali znamo algoritam po kojem se dolazi do rezultata, koristeći činjenicu da znamo zbir dva jednocifrena broja. Neka treba sabrati trocifreni dekadni broj x sa ciframa $x_3\ x_2\ x_1$ i trocifreni broj y sa ciframa $y_3\ y_2\ y_1$. Rezultat sabiranja može biti četvorocifreni dekadni broj z sa ciframa $z_4 z_3 z_2 z_1$. Imajući u vidu pretpostavku da poznajemo rezultate sabiranja dva jednocifrena broja, to bi sabiranje brojeva x+y izvršili na sledeći način:



Sl. 1.1.4

$x_3\ x_2\ x_1$
$y_3\ y_2\ y_1$

$p_1\ z_1$
$p_2\ z'_2$
$p_3\ z'_3$

$p'_2\ z_2\ z_1$
$p'_3\ z''_3$
p_3

$p''_3\ z_3\ z_2\ z_1$
z'_4

$z_4\ z_3\ z_2\ z_1$

gde su p_i i p'_i, i = 1,2,3 cifre 1 ili 0 u zavisnosti od toga da li je zbir odgovarajućih cifara dvocifren ili jednocifren broj. Na slici 1.1.4 prikazan je grafički algo-

ritam za sabiranje trocifrenih dekadnih brojeva. Na slici je pretpostavljeno da se algoritam izvršava odozgo nadole, dok smo na sl. 1.1.3 pretpostavili da algoritamski koraci slede jedan za drugim sleva nadesno. Važno je uočiti činjenicu da uvek moramo znati koji je prvi korak, kao i koji je sledeći korak pri izvršavanju algoritma. Ako uvedemo pojam vremena u proces izvršavanja algoritma, onda možemo vremenski trenutak za izvršavanje prvog algoritamskog koraka označiti sa t_1 a za ostale algoritamske korake redom t_2 , t_3 , . . . , t_m (sl. 1.1.2), gde je $t_1 < t_2$, $t_2 < t_3$, . . . , $t_{m-1} < t_m$. Na sličan način mogu se uvesti diskretna vremena t_1 , t_2 , . . . , za algoritme na sl. 1.1.3 i sl. 1.1.4. Uopšte uzevši t_1 , t_2 , . . t_m mogu pretstavljati vremenske intervale, koji slede jedan za drugim na vremenskoj skali.

Na osnovu svega što je rečeno o algoritmima možemo navesti sledeće osobine algoritama:

1) Diskretnost algoritma. Proces izvršavanja algoritma odvija se u diskretnim vremenskim intervalima. Svakom algoritamskom koraku pripada odredjen vremenski interval na vremenskoj skali.
2) Determinisanost algoritma. Skup izlaznih veličina izračunatih u ma kom algoritamskom koraku, jednoznačno je odredjen na osnovu ulaznih veličina u dotičnom algoritamskom koraku.
3) Elementarnost algoritamskog koraka. Zakon dobijanja izlaznih veličina, na osnovu ulaznih veličina algoritamskog koraka mora biti prost i jasan.
4) Usmerenost algoritma. Za svaki mogući skup ulaznih veličina u algoritmu mora biti definisano šta treba smatrati rezultatom, odnosno izlaznom veličinom, algoritma.
5) Masovnost algoritma. Skup ulaznih veličina može biti izabran podskup, skupa sa neograničeno velikim brojem elemenata.

1.2. Algoritamske strukture i njihovo grafičko prikazivanje

U primerima navedenim u prethodnom odeljku, predpostavljali smo da se, pri izvršavanju algoritma, algoritamski koraci izvršavaju jedan za drugim sa leva na desno (sl. 1.1.2 i sl. 1.1.3), odnosno odozgo na dole (sl. 1.1.4). Kao ulaze u algoritamske korake navodili smo podatke nad ko-

jima se vrši obrada u dotičnom algoritamskom koraku, a izlazna veličina je bila rezultat algoritamskog koraka. Ovakvo prikazivanje algoritama nekada može biti pogodno, ako se želi istaći tok podataka pri izvršavanju algoritamskog procesa. Medjutim, mi ćemo uvesti pojam upravljanja izvršavanjem algoritma i u daljim izlaganjima grafički prikazivati algoritme u cilju preglednog uvida u redosled algoritamskih koraka pri izvršavanju algoritma. Pod upravljanjem izvršavanjem algoritma podrazumevaćemo jednoznačno definisanje početnog koraka, kao i svakog sledećeg koraka, nakon izvršenog jednog algoritamskog koraka. U primerima na sl. 1.1.2 i sl. 1.1.3 upravljanje se sastojalo u definiciji prvog sa leva algoritamskog koraka kao početnog i susednog desnog kao sledećeg koraka, dok u primeru na sl. 1.1.4 prvi odozgo je bio početni, a prvi ispod sledeći algoritamski korak. Medjutim, u sledećim primerima dolazićemo do znatno složenijih algoritamskih struktura i opisani način prikazivanja algoritamskih struktura bio bi nepogodan. U buduće ćemo u pravougaoniku, koji označava algoritamski korak navoditi kompletnu obradu u dotičnom koraku, dok smo ranije navodili samo relaciju kojom su vezani medjusobom ulazni podaci, a rezultat je bio izlazna veličina, od sada će i rezultat biti pisan unutar algoritamskog koraka. Ulaz u algoritamski korak ukazivaće iz koga algoritamskog koraka se prenosi upravljanje na dotični algoritamski korak a izlaz iz algoritamskog koraka ukazivaće na sledeći algoritamski korak.

Da vidimo sada kakve su moguće obrade unutar jednog algoritamskog koraka. Pre svega može se izračunavati vrednost funkcije na osnovu zadatih vrednosti argumenata, ovo je relacija oblika

$$y = f(x_1, x_2, \ldots, x_n) \qquad (1.2.1)$$

gde su argumenti $x_1, x_2, \ldots x_n$ medjusobom povezani aritmetičkim operacijama. Nezavisno promenljive x_i, $i = 1, 2, \ldots, n$ i zavisno promenljivu y zvaćemo zajedničkim imenom promenljive, izuzetno ako je potrebno naglasićemo da li se radi o nezavisnoj ili zavisnoj promenljivoj. Relacija (1.2.1) je uobičajena u matematici i ne treba je posebno objašnjavati, napomenimo samo da iza oznake x_i, $i = 1, 2, \ldots, n$ podrazumevamo konkretne brojne vrednosti na osnovu kojih dolazimo do brojne vrednosti y. Znak jednakosti

(=) konstatuje činjenicu da je brojna vrednost na levoj strani jednaka brojnoj vrednosti na desnoj strani (i obratno). Medjutim, kod praktičnog izvršavanja algoritama, a posebno kada se radi o njihovom izvršavanju pomoću cifarskih računara, potrebno je osim konstatacije izraziti tok podataka u računaru. U ovom cilju koristićemo simbol $\Rightarrow$. Ovaj simbol će označavati da brojna vrednost na levoj strani simbola se uzima, kao brojna vrednost promenljive na desnoj strani ovog simbola. Tako relacija (1.2.1) se može napisati u obliku

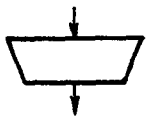
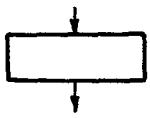
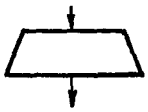
$$f(x_1, x_2, \ldots, x_n) \Rightarrow y \qquad (1.2.2)$$

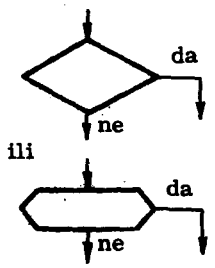
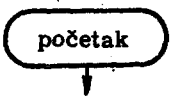
Treba dobro uočiti razliku izmedju relacije (1.2.1) i (1.2.2). Relacija (1.2.2) izražava proces izračunavanja brojne vrednosti na levoj strani simbola, a zatim pripisivanje izračunate brojne vrednosti promenljivoj y. Ovo nam daje mogućnost da pišemo

$$f_1(x_1, x_2, \ldots, x_n, y) \Rightarrow y \qquad (1.2.3)$$

što znači da se na osnovu brojnih vrednosti x_i, $i = 1, 2, \ldots, n$ i prethodne vrednosti y izračunava nova brojna vrednost za promenljivu y.

Za grafičko prikazivanje algoritama, koristićemo grafičke simbole koji će svojim oblikom ukazivati na prirodu pojedinih algoritamskih koraka. Niže su opisani pojedini grafički simboli:

	Grafički simbol za algoritamski korak u kojem se definišu ulazne veličine algoritma.
	Grafički simbol za algoritamski korak u kojem se vrši obrada.
	Grafički simbol za algoritamski korak u kojem se definišu izlazne veličine algoritma.

	Grafički simboli za algoritamske korake u kojima se donose odluke o daljem toku algoritma.
	Oznaka prvog algoritamskog koraka.
	Oznaka zadnjeg algoritamskog koraka.
	Prenošenje upravljanja na tačku n.

1.3. Linijske algoritamske strukture

Opšta karakteristika linijskih algoritamskih struktura je da pri jednom izvršavanju algoritma dolazi samo do jednog izvršavanja svakog algoritamskog koraka. Prema tome, upravljanje u linijskim algoritmima je okarakterisano time što se iz jednog algoritamskog koraka, upravljanje može preneti samo na algoritamski korak koji nije još ni jedanput izvršen.

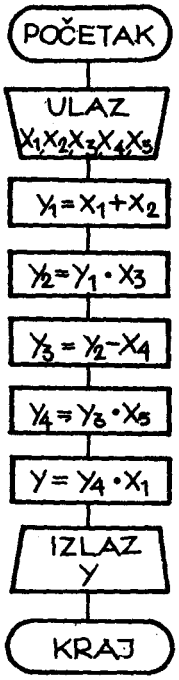
1.3.1. Proste linijske strukture

Pod prostim linijskim strukturama podrazumevaćemo algoritme čiji se koraci sastoje isključivo od obrade nad ulaznim podacima, koja kao rezultat daje brojni podatak bez dejstva na prenošenje upravljanja na sledeći algoritamski korak. Prema tome, u ovakvim strukturama redosled algoritamskih koraka je unapred definisan i ne može biti promenjen u toku rada

algoritma. Ovakve algoritamske strukture redovno se sreću pri izračunavanju aritmetičkih izraza. Tako ranije navedeni primer izračunavanja vrednosti y po formuli

$$y = \left[(x_1 + x_2)\, x_3 - x_4\right] x_5\, x_1 \qquad (1.3.1)$$

predstavlja prostu linijsku algoritamsku strukturu koja je data na sl. 1.3.1.



Sl. 1.3.1

Veza izmedju dva algoritamska koraka obrade, predstavlja bezuslovno prenošenje upravljanja sa jednog na drugi, ali ovo nije potrebno posebno označavati.

1.3.2. Razgranate linijske strukture

Kod prostih linijskih struktura videli smo da prelazak sa jednog algoritamskog koraka na sledeći ne može zavisiti od rezultata obrade u ma kojem algoritamskom koraku. Medjutim, u praktičnom računu vrlo često tok računanja zavisi od medjurezultata dobijenih u toku računanja, ili od konkretnih vrednosti polaznih podataka. U ovakvim algoritmima mora postojati algoritamski korak u kojem se donosi odluka o toku računskog procesa odnosno o prenošenju upravljanja na jedan ili drugi algoritamski korak. Najelementarnija razgranata linijska struktura dobija se komponovanjem tri proste linijske strukture P_1, P_2 i P_3 (sl. 1.3.2). Struktura P_1 neka se sastoji od izračunavanja sledećih medjurezultata

$$\begin{aligned} y_1 &= f_1(x_1, x_2, \ldots, x_n) \\ y_2 &= f_2(x_1, x_2, \ldots, x_n) \\ &\ldots\ldots\ldots\ldots\ldots\ldots \\ y_m &= f_m(x_1, x_2, \ldots, x_n) \end{aligned} \qquad (1.3.2)$$

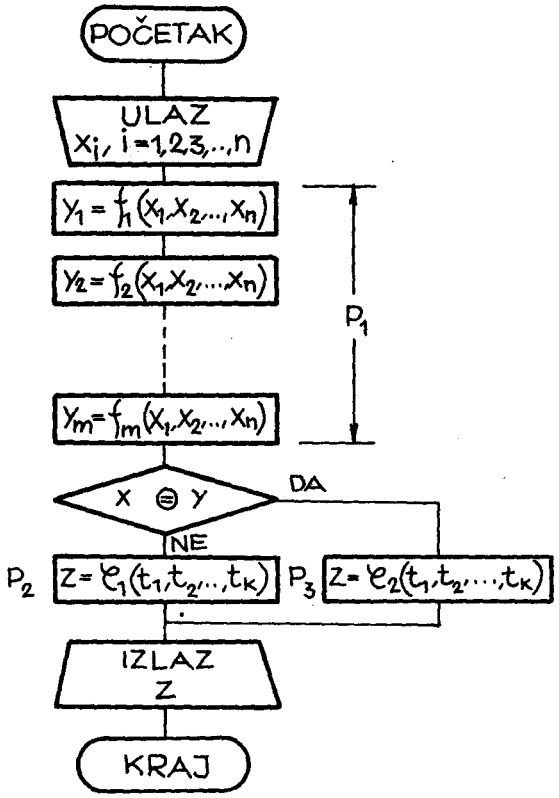
Funkcija Z se računa po jednoj od prostih linijskih struktura P_2 ili P_3, u zavisnosti od istinitosti relacije $x \ominus y$, gde simbol $\ominus$ označava ma koju re-

laciju poredjenja brojeva x i y, a brojevi x i y pripadaju skupu $\{x_i\}$, i = = 1, 2, ..., n, odnosno $\{y_i\}$, i = 1, 2, ..., m, tako da je

$$Z = \begin{cases} \varphi_1(t_1, t_2, \ldots, t_k) & \text{za } x \ominus y \text{ nije ispunjena} \\ \varphi_2(t_1, t_2, \ldots, t_k) & \text{za } x \ominus y \text{ ispunjena} \end{cases} \qquad (1.3.3)$$

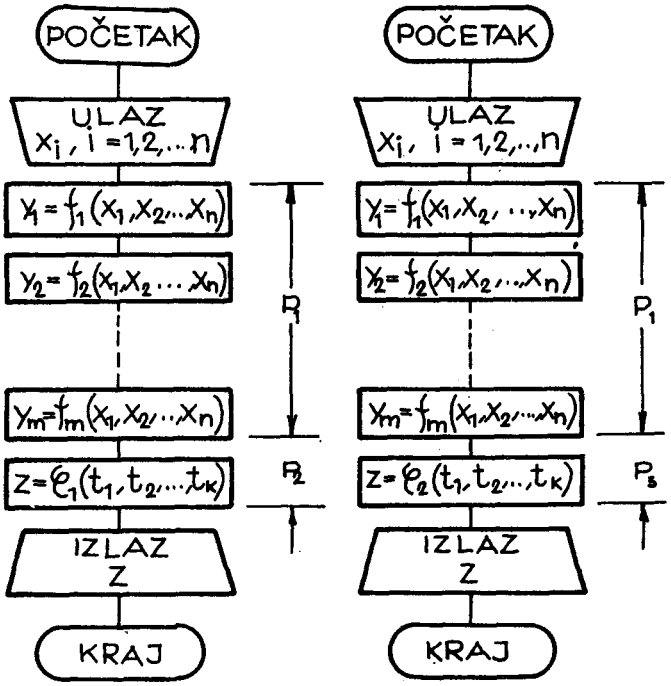
gde argumenti funkcije φ_1 i φ_2 mogu biti iz skupa polaznih veličina $\{x_i\}$, i = 1, 2, ..., n ili iz skupa medjurezultata $\{y_i\}$ i = 1, 2, ..., m. Relacije $\ominus$ izmedju brojeva x i y mogu biti:

1) $x = y$, suprotno $x \neq y$
2) $x < y$, suprotno $x \geqslant y$
3) $x > y$, suprotno $x \leqslant y$



Sl. 1.3.2

Ispitivanje istinitosti naznačene relacije izmedju brojeva x i y, grafički je označeno rombom, u čijoj unutrašnjosti se navodi relacija koja se ispituje. Algoritamski korak u kojem se vrši ispitivanje istinitosti relacije izmedju brojeva ima jedan ulaz i dva izlaza. Jedan izlaz prenosi upravljanje u slučaju da navedena relacija nije zadovoljena (označimo ga sa NE), a drugi izlaz prenosi upravljanje u slučaju da je relacija zadovoljena (označavamo ga sa DA).



Sl. 1.3.3.

Sl.1.3.4

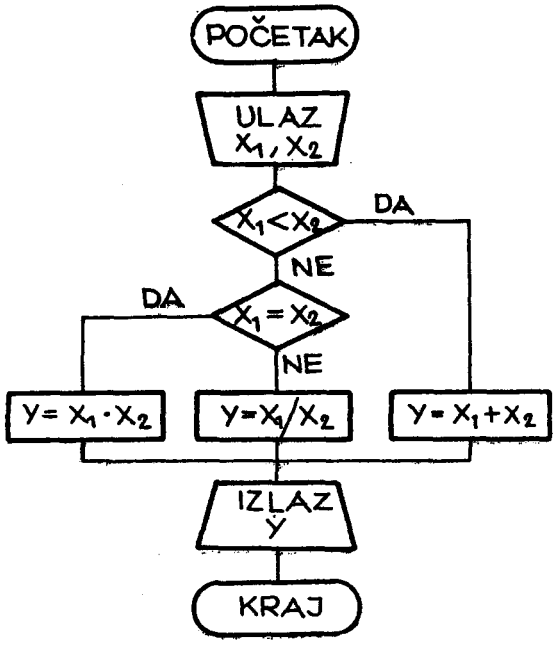
Izvršavanje algoritama sa razgranatom strukturom uvek se sastoji od prostih linijskih struktura. Tako na sl. (1.3.2) nikada se ne izvršavaju obe strukture P_2 i P_3 za konkretne vrednosti polaznih podataka, već će se izvršiti jedna od ovih struktura. U slučaju da naznačena relacija nije ispunjena izvršiće se struktura P_1 , a zatim struktura P_2 (sl. 1.3.3), odnosno ako je naznačena relacija ispunjena, izvršiće se struktura P_1 a zatim P_3 (sl.1.3.4). Povezivanjem elementarnih razgranatih struktura mogu se dobiti vrlo složene razgranate algoritamske strukture.

Primer:

Izračunati vrednost y po formuli

$$y = \begin{cases} x_1 + x_2 & \text{za } x_1 < x_2 \\ x_1 \cdot x_2 & \text{za } x_1 = x_2 \\ x_1/x_2 & \text{za } x_1 > x_2 \end{cases}$$

Na sl. 1.3.5 prikazan je grafički algoritam za izračunavanje vrednosti y. U navedenom primeru dovoljno je ispitati da li je $x_1 < x_2$ i $x_1 = x_2$. Ako nijedan od ovih uslova nije ispunjen znači da je $x_1 > x_2$.



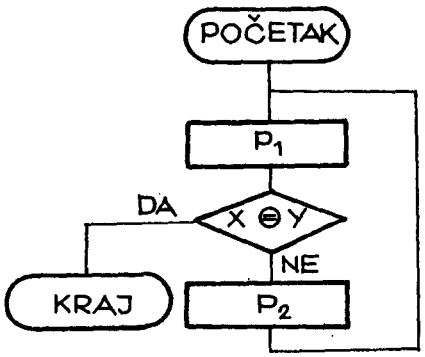
Sl. 1.3.5

1.4. Cikličke algoritamske strukture

Opšta karakteristika cikličkih algoritamskih struktura jeste višestruko izvršavanje jednog algoritamskog koraka ili više njih za razliku od linijskih struktura kod kojih se svaki korak samo jedanput izvršava. Razlikova-

ćemo dva tipa cikličkih struktura:

- konstantne cikličke strukture i
- promenljive cikličke strukture.



Sl. 1.4.1

Svaka od ovih struktura u najprostijem slučaju sastoji se od dve proste linijske strukture (P_1 i P_2), izmedju kojih se nalazi uslov za izlazak iz ciklusa, odnosno za nastavljanje ciklusa (sl. 1.4.1). I ovde, kao i kod razgranatih linijskih struktura, uslov se izražava odnosom dva broja x ⊖ y. Relacija izmedju brojeva x i y može biti ispunjena ili ne; u jednom slučaju upravljanje se predaje algoritamskom koraku van ciklusa, a u drugom slučaju algoritamskom koraku u ciklusu. Uobičajeno je da se cikličke algoritamske strukture zovu još i petlje, a uslov za izlazak iz ciklusa zove se još i izlazni kriterijum. Ciklička struktura može imati i više izlaza, u zavisnosti od broja relacija koje se ispituju u izlaznom kriterijumu.

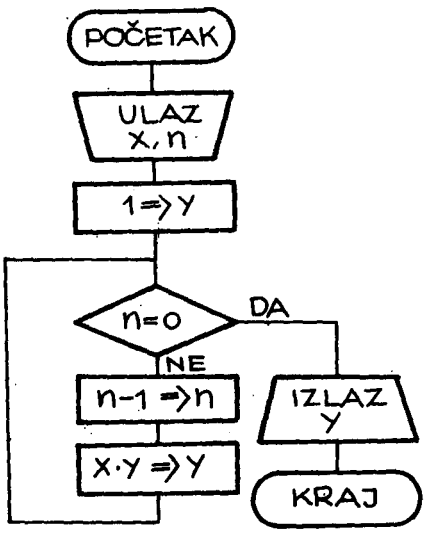
1.4.1. Konstantne cikličke strukture

Ako u toku izvršavanja algoritma ne dolazi do promena zakona obrade u algoritamskim koracima koji čine cikličku strukturu, kažemo da je to konstantna ciklička struktura. Izlazni kriterijum kod konstantnih cikličkih struktura je najčešće broj izvršenih ciklusa ili pak dostignuta tačnost pri računanju po iterativnom postupku. Navešćemo primer jednog i drugog izlaznog kriterijuma.

Primer 1

Izračunati vrednost y ako je

$y = x^n$



Sl. 1.4.2

gde je $x \neq 0$, $n = 0, 1, 2, \ldots$. Ovde je izlazni kriterijum broj ponavljanja ciklusa u cilju stepenovanja broja x u našem slučaju ciklus treba izvršiti n puta. Pošto je vrednost n promenljiva od slučaja do slučaja, to bi bilo nemoguće ovaj algoritam opisati prostom linijskom strukturom koja bi važila za proizvoljno n, već bismo za svako različito n morali imati posebnu linijsku strukturu. Na sl. 1.4.2 prikazan je grafički algoritam za izračunavanje n-tog stepena broja x.

Primer 2

Izračunati kvadratni koren broja z

$$v = \sqrt{z} \tag{1.4.2}$$

po Njutnovoj iterativnoj formuli

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{z}{x_i}\right), \qquad i = 0, 1, 2, \ldots \tag{1.4.3}$$

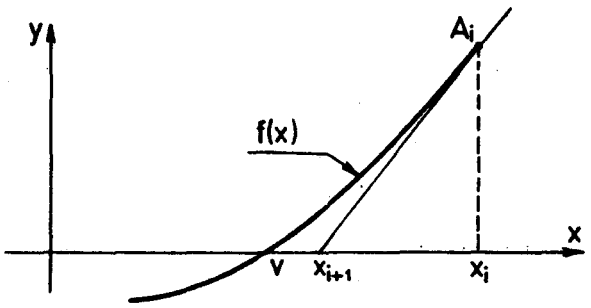
gde ćemo uzeti da je $x_0 = z + 1$. Proces računanja prekinuti kada se dostigne zadata tačnost ε, tako da je

$$x_i - x_{i+1} < \varepsilon \tag{1.4.4}$$

Formula (1.4.3) dobija se kao poseban slučaj odredjivanja nule funkcije f(x) pomoću Njutnove tangentne metode (sl. 1.4.3). Jednačina tangente kroz tačku A_i ima oblik

$$y - f(x_i) = f'(x_i)(x - x_i) \tag{1.4.5}$$

Za $x = x_{i+1}$ jednačina (1.4.5) daje



Sl. 1.4.3

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}, \qquad i = 0, 1, 2, \ldots \tag{1.4.6}$$

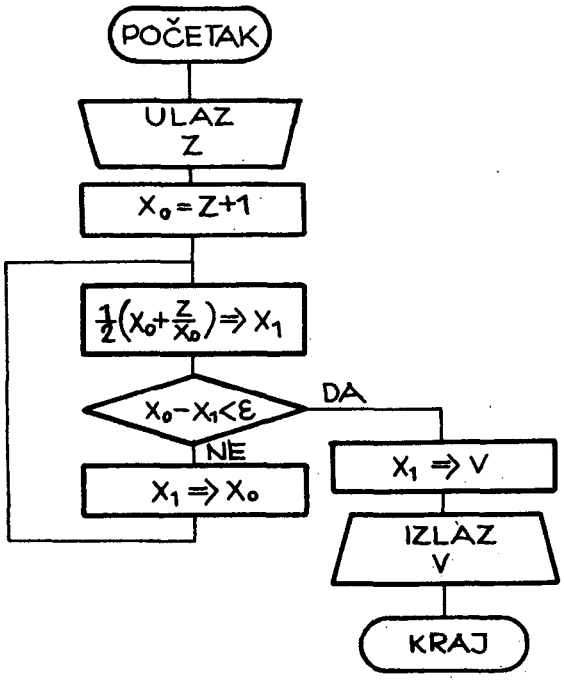
Ako uzmemo da je

$$f(x) = x^2 - z \tag{1.4.7}$$

gde je z broj čiji kvadratni koren treba odrediti, to se iz (1.4.6) dobija (1.4.3). Može se pokazati da je izloženi iterativni postupak izračunavanja kvadratnog korena uvek konvergentan, i da je

$$\lim_{i \to \infty} x_i = v \tag{1.4.8}$$

Na sl. 1.4.4 prikazan je algoritam za izračunavanje kvadratnog korena. Ovde je važno uočiti razliku izmedju matematičke simbolike u iterativnoj formuli (1.4.3) i one koja je korišćena na sl. 1.4.4. Po formuli (1.4.3) u svakom iterativnom ciklusu dolazi do uvodjenja nove promenljive x_1, x_2, ..., kojih može biti veliki broj, što zavisi od brzine konvergencije iterativnog postupka. Medjutim, u svakom iterativnom ciklusu stvarno se pojavljuju samo dva brojna podatka: rezultat prethodne iteracije i rezultat iteracije koja se računa. Pri grafičkom prikazivanju algoritma na sl. 1.4.4 uzeto je da promenljiva x_0 predstavlja rezultat prethodne iteracije, a x_1 rezultat iteracije koja se računa. Ako nije dostignuta zadata tačnost dolazi do zamene izračunate vrednosti x_1 kao nove pretpostavljene vrednosti i iterativni proces se nastavlja.



Sl. 1.4.4

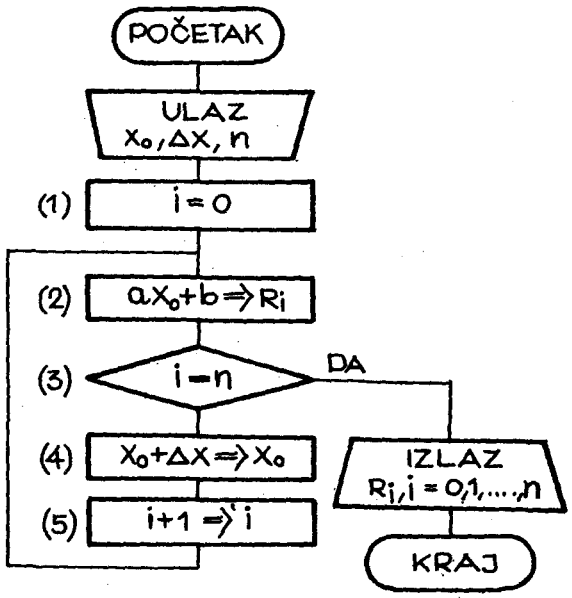
Pri grafičkom prikazivanju algoritama treba težiti za što manjim brojem promenljivih, jer to znači i mali broj angažovanih registara memorije, kada se algoritam prenosi na elektronske računare.

Cikličke algoritamske strukture kod kojih je izlazni kriterijum dostignuta tačnost u iterativnom postupku zovu se još i iterativne cikličke strukture. Važno je uočiti da se iterativne cikličke strukture ne mogu razviti u linijsku strukturu, jer broj ponavljanja iterativnog ciklusa zavisi od polaznih podataka i različit je za različite vrednosti polaznih podataka. Pored toga, broj ponavljanja iterativnog ciklusa zavisiće i od zadate tačnosti sa kojom se želi odrediti traženo rešenje.

1.4.2. Promenljive cikličke strukture

Kod konstantnih cikličkih struktura dolazilo je do promene ulaznih podataka u pojedine algoritamske korake u ciklusu, ali zakon obrade u algo-

ritamskom koraku ostao je nepromenjen za vreme izvršavanja ciklusa. Kod promenljivih cikličkih struktura, u toku rada ciklusa, dolazi do promena zakona obrade u nekom od algoritamskih koraka u okviru ciklusa. Ova promena može biti nad promenljivim koje se javljaju u algoritamskom koraku ili nad operacijama kojima su one povezane medju sobom.



Sl. 1.4.5

Promene u algoritmu učinjene izvršavanjem samog algoritma zvaćemo modifikacija algoritma. Realizacija algoritama na cifarskim elektronskim računarima dozvoljava njihovu modifikaciju, što omogućuje znatno krače zapisivanje algoritama u memoriji računara. Modifikacija algoritma znači mogućnost obrade ne samo polaznih podataka, već i informacija koje čine i sam algoritam.

Primer 1

Izračunati vrednost funkcije

$$f(x_i) = ax_i + b \qquad (1.4.9)$$

za $x_i = x_0 + i \cdot \Delta x$, $i = 0, 1, \ldots, n$. Izračunate vrednosti funkcije $f(x_i)$ dodeliti promenljivim R_i. Na sl. 1.4.5 dat je grafički prikaz algoritma za rešavanje ovog zadatka. Algoritam je sastavljen od pet algoritamskih koraka, na slici označenih sa (1), (2), (3), (4) i (5). Algoritamski koraci (2), (3), (4) i (5) nalaze se unutar cikličke strukture, dok se korak (1) nalazi ispred cikličke strukture. Korak (1) služi za postavljanje početne vrednosti indeksa i, tako da se u prvom prolazu rezultat algoritamskog koraka (2) dodeljuje promenljivoj R_0. U trećem algoritamskom koraku proverava se izlazni kriterijum iz ciklusa, tako da ako je izračunata i vrednost funkcije $f(x_n)$ dolazi se do kraja algoritma. U četvrti algoritamski korak se dolazi ako izlazni kriterijum nije zadovoljen, i izračunava se sledeća vrednost argumenta funkcije. Ovo izračunavanje se sastoji u dodavanju koraka Δx na prethodnu vrednost argumenta. Kako prethodna vrednost argumenta ne mora biti sačuvana, to je dovoljno imati samo jednu promenljivu x_0, čija će se vrednost menjati u toku izvršavanja algoritma. Peti algoritamski korak je karakterističan za promenljive cikličke strukture. U ovom koraku dolazi do promene indeksa i, a to praktično znači do promene imena promenljive kojoj se dodeljuje rezultat drugog algoritamskog koraka. Prema tome, modifikacija algoritma, u ovom slučaju, sastoji se u promeni mesta gde će biti zapisan rezultat algoritamskog koraka, pošto se u memoriji računara svakoj promenljivoj dodeljuje registar u kojem se čuva njena brojna vrednost.

Primer 2

Sastaviti algoritam za izračunavanje vrednosti polinoma

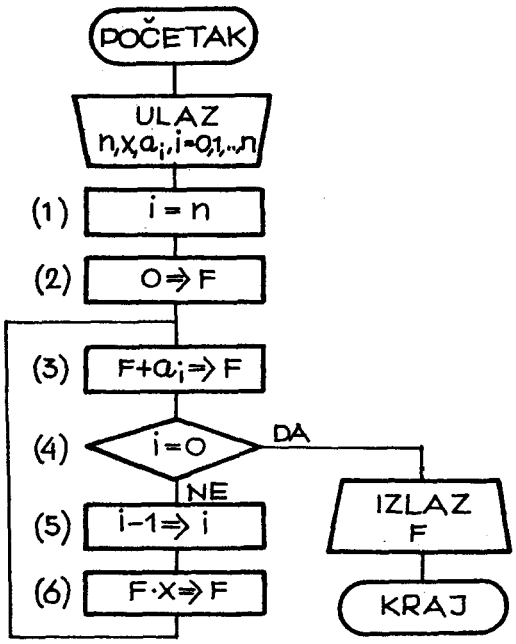
$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 \qquad (1.4.10)$$

za zadatu vrednost x-a. Algoritam sastaviti tako da se može koristiti za ma koji stepen polinoma. Izračunatu vrednost polinoma dodeliti promenljivoj F. Polinom (1.4.10) može se napisati u obliku

$$f(x) = a_0 + x \left\{ a_1 + x \left[a_2 + \ldots + x(a_{n-1} + x \cdot a_n) \right] \right\} \qquad (1.4.11)$$

koji je pogodniji za izračunavanje vrednosti polinoma, jer omogućuje sastavljanje cikličke algoritamske strukture.

Na sl. 1.4.6 dat je algoritam za izračunavanje vrednosti polinoma po formuli (1.4.11). Algoritam pored ulaznih i izlaznih koraka sadrži 6 koraka od kojih su dva pripremna, a ostala 4 se nalaze u ciklusu. Sa prva dva koraka vrši se postavljanje početne vrednosti za i i dodeljivanje brojne vrednosti nula promenljivoj F. Kada se prvi put dodje u treći algoritamski korak, biće F = 0, tako da će se izvršiti operacija dodeljivanja $a_n \Rightarrow F$.



Sl. 1.4.6

U šestom algoritamskom koraku vrši se množenje F sa x i postavljanje proizvoda kao nove vrednosti promenljive F. Promenom indeksa i vrši se modifikacija promenljive a_i redom u promenljive a_n, a_{n-1}, ..., a_1, a_0. Kada indeks i dostigne vrednost 0, to znači da je u trećem koraku izvršeno dodavanje i zadnje konstante a_0 prethodnoj vrednosti promenljive F, čime je vrednost polinoma izračunata, i sa ovim se algoritam završava.

1.5. Složene algoritamske strukture

Do sada smo se upoznali sa elementarnim algoritamskim strukturama: prostim i razgranatim linijskim strukturama, kao i konstantnim i pro-

menljivim cikličkim strukturama. Različitim kompozicijama ovih elementarnih struktura dolazi se do vrlo složenih i raznovrsnih algoritamskih struktura. Jasno je takodje da se za rešavanje istog zadatka može sastaviti više algoritama sa različitim strukturama. Za ovakvè algoritme kažemo da su medjusobom ekvivalentni. Pri praktičnom rešavanju zadataka treba medju ekvivalentnim algoritmima izabrati onaj koji najefikasnije dovodi do rezultata. Ovo je posebno važno kada se radi o primeni računara u cilju izvršavanja algoritama. U ovom slučaju na konačan izbor algoritma može uticati broj angažovanih registara memorije, brzina rada algoritma, složenost algoritamske strukture i sl.

Složenost algoritamske strukture naročito se uvećava prisustvom cikličkih, a posebno promenljivih cikličkih struktura. Dve cikličke strukture, u kompoziciji algoritma, mogu slediti jedna iza druge ili mogu obuhvatiti jedna drugu. Za više cikličnih struktura koje slede jedna iza druge kažemo da čine linijsku kompoziciju cikličkih struktura. Ako se jedna ciklička struktura nalazi unutar druge cikličke strukture kažemo da se radi o koncentričnoj kompoziciji cikličkih struktura.

Primer koncentrične kompozicije cikličkih struktura.

Izračunati vrednost polinoma

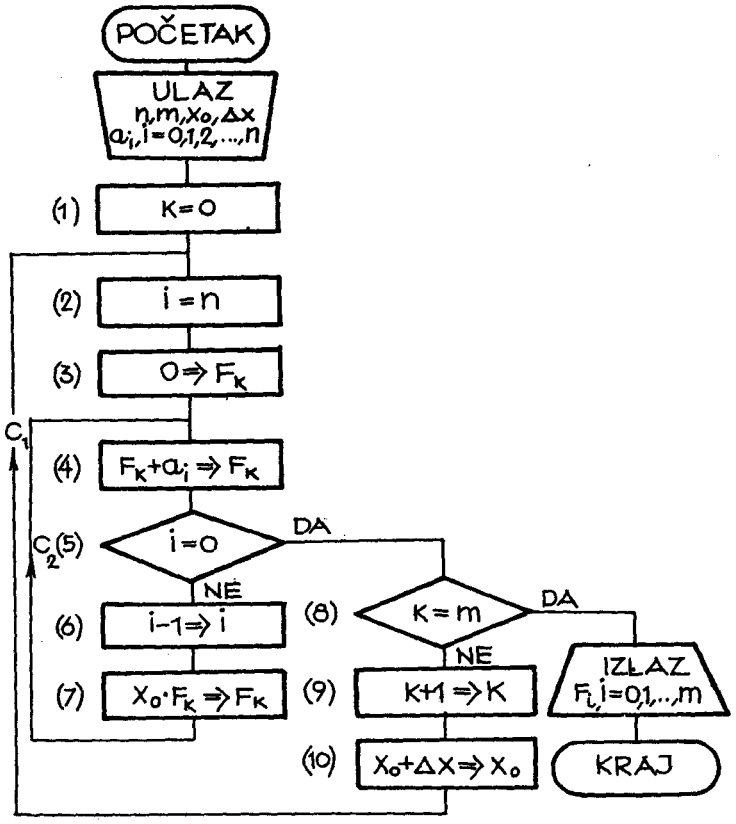
$$f(x) = a_0 + x\left\{ a_1 + x\left[a_2 + \ldots + x(a_{n-1} + xa_n)\right]\right\} \qquad (1.5.1)$$

za $x = x_0 + k \cdot \Delta x$, $k = 0, 1, 2, \ldots, m$.

Izračunate vrednosti polinoma dodeliti promenljivim F_i, $i = 0, 1, \ldots, m$ tako da je

$$F_i = f(x_i), \qquad i = 0, 1, \ldots, m$$

Na sl. 1.5.1. dat je algoritam za izračunavanje vrednosti polinoma, po postupku (1.5.1), a za razne vrednosti argumenta. Algoritam se sastoji od 10 koraka sa dve cikličke strukture (C_1 i C_2), od kojih se jedna ciklička struktura (C_2) nalazi u okviru druge cikličke strukture (C_1). Prvi algoritamski korak je pripremni i vrši postavljanje indeksa k. Algoritamski ko-



Sl. 1.5.1.

raci od (2) do (7) odgovaraju ranijem primeru izračunavanja vrednosti polinoma datom na sl. 1.4.6. Algoritamski korak (8) predstavlja proveru izlaznog kriterijuma za izlazak iz ciklusa C_1, a u koraku (9) vrši se promena indeksa k. Povećanje argumenta x za korak Δx vrši se u desetom algoritamskom koraku. Prema tome, u malom ciklusu C_2 vrši se izračunavanje vrednosti polinoma n-tog stepena, a u velikom ciklusu C_1 vrši se promena argumenta i postavljanje početnih vrednosti za odvijanje ciklusa C_2. U algoritmu na sl. 1.5.1. dolazi do modifikacije u slèdećim algoritamskim koracima (3), (4) i (7), jer dolazi do promene imena promenljivim a_i i F_k pri svakoj promeni indeksa i odnosno k. Postavljanje indeksa i u koraku (2), u cilju pravilnog odvijanja ciklusa C_2 (izračunavanje vrednosti polinoma), zove se restauracija vrednosti promenljive i.

1.6. Algoritmi i programiranje

Videli smo da algoritam možemo jednoznačno opisati koristeći se blok šemom. Medjutim, jasno je da bi smo algoritam takodje mogli opisati ako bi smo se koristili izražavanjem u obliku pisanog teksta. Tako ranije uzeti primer izračunavanja funkcije

$$f(x_i) = ax_i + b$$

za $x_i = x_0 + i \cdot \Delta x$, $i = 0, 1, \ldots, n$ i dodeljivanje izračunatih vrednosti funkcije $f(x_i)$ promenljivim R_i, $i = 0, 1, \ldots n$, može se opisati na sledeći način:

1 korak: Uzeti da je $i = 0$, i preći na korak 2,

2 korak: Izračunati $f(x_0) = ax_0 + b$, i izračunatu vrednost dodeliti promenljivoj R_i, a zatim preći na korak 3,

3 korak: Ako je $i = n$ završiti računanje, a ako je $i \neq n$ preći na korak 4,

4 korak: Povećati vrednost argumenta za Δx, i preći na korak 5,

5 korak: Povećati indeks i za jedan, i preći na korak 2.

Poredjenjem gornjeg opisa algoritma i onog datog na sl. 1.4.5. vidi se da su oba opisa u suštini ista. Medjutim, tekstualni opis algoritama nije pogodan za složenije algoritme, jer se jasno ne uočava struktura algoritma, a to smanjuje mogućnosti u otkrivanju logičkih grešaka. Sa druge strane pri tekstualnom opisu može doći do neprecizno formulisanih rečenica, čime se unosi nejasnoća u opisivanje algoritma.

Grafičko prikazivanje algoritama pomoću blok-šeme odlikuje se sledećim osobinama:

1) Omogućuje prikazivanje toka izvršavanja algoritma, na način koji pruža najveće šanse za otkrivanje logičkih grešaka u postavci algoritma.

2) Omogućuje kraći i jasniji zapis algoritma, nego pisanim jezikom.

3) Daje preglednu vezu izmedju detalja i celine.

4) Pisanje algoritama u obliku blok-šema nezavisno je od načina kasnijeg korišćenja algoritma.

Zadnja osobina je posebno značajna, jer grafičko prikazivanje algoritma nije orijentisano na prenošenje algoritma na odredjeni računar, što omogućuje da algoritam u obliku blok-šeme mogu koristiti i ljudi koji ne poznaju računare. Detaljisanje algoritma pri grafičkom prikazivanju može biti različito i zavisi od namene blok-šeme algoritma. Kada se radi o prenošenju algoritma na cifarske računare, radi se u stvari o takvoj detaljizaciji algoritma, koja će omogućiti jednoznačno prihvatanje algoritma od strane računara. Ovakav način pisanja algoritama zove se programiranje, a ovako zapisan algoritam zvaćemo programski algoritam ili program. Prema tome, programski algoritam je takav zapis algoritma koji omogućuje njegovo prenošenje na cifarske računare. Algoritam zapisan u obliku blok-šeme ne može biti prihvaćen od strane računara, jer detaljizacija algoritma nije strogo definisana.

Algoritam zapisan preko skupa naredbi računara prihvatljiv je za računar, ali ne odgovara čoveku. Nivo mašinskog jezika za programiranje podrazumeva takvu detaljizaciju algoritma da to predstavlja vrlo mukotrpan posao za čoveka, a što je najvažnije on je veoma podležan greškama. Za pisanje programa na mašinskom jeziku potrebno je poznavanje konstruktivnih osobina računara, što bi takodje veoma ograničilo broj ljudi koji bi mogli koristiti računar.

Sledeća faza u rešavanju problema komunikacije izmedju čoveka i računara bila je uvodjenje simboličnog programiranja. Simbolično programiranje podrazumeva uvodjenje mnemotehničkih skraćenica za kodove operacija, kao i simbolično pisanje adresa. Svakako da je ovakav simboličan jezik prihvatljiviji za čoveka nego mašinski jezik. Medjutim, i ovaj način programiranja zahteva veliku detaljizaciju, koja nosi sve nedostatke mašinskog jezika, samo u blažoj formi.

Simboličan jezik mora biti tako definisan da je moguće sastaviti algoritam za formalno prevodjenje na mašinski jezik. Pošto je ovo prevodjenje jednoznačno to ga je moguće izvršiti pomoću računara.

1.7. Uopšte o programskim jezicima

Jezik je sredstvo za komunikaciju izmedju najmanje dva korisnika. Očigledno je da jezik mora biti definisan tako da je prihvatljiv za sve korisni-

ke. Ako se posmatra problem komunikacije izmedju čoveka i računara moraju se uzeti u obzir dobre i loše strane jednog i drugog i na bazi toga mora se formirati jezik za njihovu komunikaciju. Jezik računara je veoma siromašan i zbog toga ne prihvatljiv za čoveka. Medjutim, pisani ili govorni jezik čoveka nije dovoljno precizan da bi bio prihvatljiv za računar. Prema tome, mora se tražiti nekakav novi jezik koji će biti prihvatljiv i za čoveka i za računar. Jezici koji su definisani da zadovolje ovaj uslov zovu se programski jezici. Programski jezik mora da odgovori sledećim zahtevima:

1) Da pruži što je moguće veći komfor za čoveka, pri prenošenju algoritama na računar,

2) Da omogući lako praćenje programskog algoritma od što većeg broja ljudi, i

3) Da je moguće formalno prevodjenje sa programskog jezika na mašinski jezik.

Prvi zahtev znači da programski jezik mora obezbediti lako izražavanje o problemu koji se želi rešiti pomoću računara. Medjutim, primena računara je jako široka, a samim tim i problemi su raznovrsni. U takvoj situaciji definisani su programski jezici za pojedine oblasti primene računara. Danas su najpoznatiji programski jezici:

FORTRAN - namenjen naučno-tehničkim problemima (FORmula TRANslating),

ALGOL - namenjen naučno-tehničkim problemima (ALGOritam Language),

COBOL - namenjen poslovnoj obradi podataka (COmon Business Orientid Language),

PL/I - namenjen naučnim, tehničkim i poslovnim problemima (Programming Language I)

Pored ovih jezika postoji još na desetine programskih jezika. Medjutim, od svih ovih jezika najviše je rasprostranjen FORTRAN. Na skoro 80% današnjih računara može se koristiti programski jezik FORTRAN.

Drugi navedeni zahtev za programski jezik, treba da omogući razmenu programskih algoritama medju stručnjacima koji se bave odgovarajućim problemima. Prema tome, programski jezik mora biti izgradjen na uobičajenom skupu tipografskih simbola, i konstrukcije u jeziku moraju biti lako shvatljive za što širi krug stručnjaka.

Treći zahtev omogućuje izradu programa, koji će obezbediti da računar vrši prevodjenje sa programskog na mašinski jezik. Ovaj program se zove program za prevodjenje, i kao ulazne podatke dobija konstrukcije iz programskog jezika i prevodi ih u skup naredbi u mašinskom jeziku. Kada je ceo program preveden sa programskog na mašinski jezik, tada može početi njegovo izvršavanje na računaru.

2. PRETHODNE NAPOMENE O FORTRAN-JEZIKU

2.1. Opšti pojmovi

Osnovna, nedeljiva jedinica jezika zove se simbol. FOTRAN je veštački jezik, koji se definiše nad izabranim skupom simbola. Skup simbola je izabran tako da odgovara uobičajenim tipografskim simbolima. Novi simboli su konstruisani od postojećih tipografskih simbola, a njihova konstrukcija je lako shvatljiva za širi krug stručnjaka različitih profila.

Simbol kao jedinica jezika ne izražava ništa drugo, osim što predstavlja samog sebe.

Elementarna konstrukcija u FORTRAN-jeziku sačinjena je od niza simbola. Elementarna konstrukcija ima odredjeno značenje, ali sama za sebe ne egzistira u programu. Elementarne konstrukcije FORTRAN-jezika su:

- konstante,
- promenljive,
- nizovi i
- izrazi.

Složena konstrukcija, u FORTRAN jeziku sačinjena je od niza simbola i elementarnih konstrukcija. Ona egzistira u programu i ima odredjeni smisao sama za sebe. Složene konstrukcije u FORTRAN-jeziku jesu:

- naredbe,
- potprogrami i
- programi.

Naredbe FORTRAN-jezika se dele na: izvršne i opisne. Izvršne naredbe odredjuju koja operacija treba da se izvrši i nad kojim podacima, pa prema tome one predstavljaju akciju koju računar treba da sprovede. Opisane naredbe pružaju sve dodatne informacije potrebne za izvršne naredbe, koje se odnose na to kako treba izvršiti odredjenu akciju, ili daju informacije o programu u celini, što omogućuje lakše prevodjenje programa sa FORTRAN-jezika na mašinski jezik.

Pravila kako se nad skupom simbola jezika grade elementarne i složene konstrukcije čine gramatiku jezika. Prema tome, poznavanjem gramatike možemo reći o svakoj zadatoj konstrukciji u jeziku da li je korektna ili nije, ne ulazeći u njeno značenje. Sintaksa jezika izučava gramatički korektne konstrukcije i daje mogućnost formalnog otkrivanja grešaka u konstrukcijama. Odmah treba uočiti da sintaksičke greške u programu mogu biti otkrivene u programu za prevodjenje, jer su formalne prirode.

Semantika jezika izučava značenje pojedinih konstrukcija u jeziku.

Program sastavlja čovek, i sve konstrukcije FORTRAN-jezika koje čine program, napisane su prema algoritmu sastavljenom za rešavanje odredjenog problema. Prema tome, semantičke greške u programu, po pravilu, ne mogu biti formalno otkrivene, jer su to najčešće greške u algoritmu, a samim tim ove greške ne može otkriti program za provodjenje, već jedino čovek.

Pravila po kojima se grade pojedine konstrukcije FORTRAN-jezika lako se pamte, ako se uoče razlozi zašto su ona uvedena. Zato treba imati u vidu sledeće:

1) svaka konstrukcija u FORTRAN-jeziku, mora biti tako definisana da se jednoznačno razlikuje od svih ostalih konstrukcija,

2) konstrukcije se moraju tako definisati da se omogući što lakše njihovo prevodjenje na mašinski jezik, i

3) konstrukcije moraju biti što razumljivije za čoveka.

Pri definisanju svakog programskog jezika mora se voditi računa o ova tri aspekta jezika. Kako je to sprovedeno u FORTRAN-jeziku biće izloženo u materijalu koji sledi.

2.2. Način pisanja programa

Sve naredbe FORTRAN-programa moraju biti prenete na kartice da bi program bio unet u računar. Da bi program za prevodjenje pravilno prihvatao naredbe sa kartica, moraju se poštovati sledeća pravila o bušenju kartica:

1) Od 1. do 5. kolone zaključno buši se broj koji predstavlja obeležje naredbe.

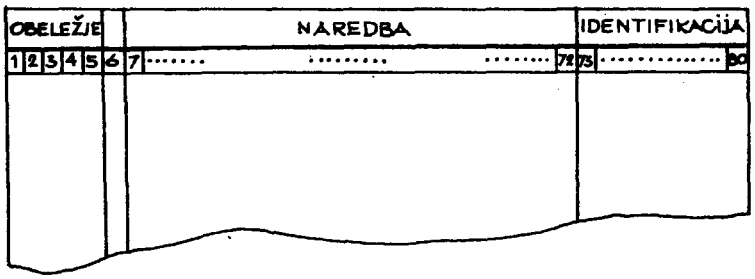
2) Od 7. do 72. kolone zaključno buši se niz simbola koji čine naredbu.

3) Ako neka naredba sadrži više od 66 simbola, odnosno ne može biti bušena na jednoj kartici, može se koristiti ukupno 20 kartica za njeno bušenje, ali svaka kartica, osim prve, mora sadržati znak različit od nule i blanka u 6. koloni.

4) Od 73. do 80. kolone može se bušiti ma kakav tekst. Najčešće je to tekst koji služi za identifikaciju programa i za redosled kartica u programu. Sadržaj ovih kolona ne uzima se u obzir od programa za prevodjenje.

5) Objašnjenja u programu koja se ne odnose na program za prevodjenje, već olakšavaju praćenje programa od strane čoveka, pišu se od 2. do 80. kolone, pri čemu se u prvoj koloni mora nalaziti slovo C. Ovakav tekst se može pisati na bilo kojem mestu programa, i bez značaja je pri formiranju mašinskog programa.

Radi preglednog sprovodjenja navedenih pravila o pisanju programa, pogodno je pisati FORTRAN-program u formularima oblika prikazanog na sl. 2.2.1.



Sl. 2.2.1

Naredbe FORTRAN-programa pišu se odozgo nadole onim redom kako se izvršavaju u programu. U okviru naredbe može se nalaziti proizvoljan broj simbola blanko. Ovo omogućuje pregledno pisanje naredbe, a ne menja njeno značenje.

3. SIMBOLI FORTRAN-JEZIKA

Skup simbola FORTRAN-jezika sastavljen je od

- velikih slova engleske azbuke,
- cifara dekadnog brojnog sistema,
- logičkih konstanti,
- znakova za aritmetičke operacije,
- znakova za operacije poredjenja,
- znakova za logičke operacije,
- specijalnih znakova i
- službenih reči.

3.1. Velika slova engleske azbuke

Velika slova engleske azbuke čine 26 simbola FORTRAN-jezika, i to

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

gde vertikalna crta nije simbol FORTRAN-jezika, već razdvaja pojedine simbole jezika.

3.2. Cifre dekadnog brojnog sistema

Cifre dekadnog brojnog sistema čine 10 simbola FOTRAN-jezika i to

0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

3.3. Logičke konstante

Logičke konstante, koje se u algebri logike najčešće označavaju simbolima 0 i 1, u FORTRAN-jeziku se pišu sa

FALSE. | .TRUE.

gde konstanta .FALSE. (čita se fo:ls, a znači laž) odgovara 0, a .TURE. (čita se tru:, a znači istina) odgovara 1 u algebri logike.

3.4. Znaci aritmetičkih operacija

Znaci aritmetičkih operacija u FORTRAN-jeziku su

+ | - | * | / | **

pri čemu imaju sledeće značenje

- \+ sabiranje
- \- oduzimanje
- * množenje
- / delenje
- ** stepenovanje

3.5. Znaci za operacije poredjenja

Znaci za operacije poredjenja u FORTRAN-jeziku su

.LT. | .LE. | .EQ. | .NE. | .GT. | .GE.

pri čemu imaju sledeće značenje:

- .LT. odgovara simbolu < u matematici (simbol je sastavljen od prvih slova engleskih reči "Less Than", što znači "manje od"),
- .LE. odgovara simbolu ≤ u matematici (simbol je sastavljen od prvih slova engleskih reči "Less than or Equal to", što znači "manje ili jednako"),
- .EQ. odgovara simbolu = u matematici (simbol je sastavljen od prvih slova engleske reči "EQual to", što znači "jednako"),

.NE. odgovara simbolu $\neq$ u matematici (simbol je sastavljen od prvih slova engleskih reči "Not Equal to", što znači "nije jednako"),

.GT. odgovara simbolu > u matematici (simbol je sastavljen od prvih slova engleskih reči "Greater Than", što znači "veće od"),

.GE. odgovara simbolu $\geq$ u matematici (simbol je sastavljen od prvih slova engleskih reči "Greater than or Equal to", što znači "veće ili jednako").

3.6. Znaci za logičke operacije

Znaci za logičke operacije u FORTRAN-jeziku su

.OR. | .AND. | .NOT.

pri čemu imaju sledeće značenje

.OR. logička "ili" operacija,

.AND. logička "i" operacija,

.NOT. logička "ne" operacija.

3.7. Specijalni znaci

Specijalni znaci, obuhvataju interpunkcijske znake, koji se mogu koristiti u FORTRAN-jeziku i to su

(|) | = | , | . | $ | b | ' | &

gde simbol b označava medjuprostor ili blanko izmedju tipografskih simbola.

3.8. Službene reči

Službene reči su engleske reči koje se u FORTRAN-jeziku koriste kao simboli. To znači da se te reči mogu pisati samo u obliku datom u tabeli 3.1. U tabeli je dat oblik pisanja službene reči u FORTRAN-jeziku, a pošto značenje reći ukazuje na funkciju simbola u FORTRAN-jeziku to je dat i prevod reči na srpskohrvatski jezik radi lakšeg korišćenja za one či-

taoce koji ne poznaju engleski jezik. Takdoje je u tabeli dat i izgovor reči, zapisan u fonetskoj transkripciji.

Tabela 3.1.

Red. br.	Službena reč	značenje	čita se
1.	ASSIGN	dodeli	əsain
2.	BLOCK DATA	paket podataka	blok deitə
3.	CALL	pozovi	kol
4.	COMMON	zajednički	kɔmən
5.	COMPLEX	kompleksan	kɔmpleks
6.	CONTINUE	nastavi	kəntinju:
7.	DATA	podaci	deitə
8.	DIMENSION	dimenzija	dimenšən
9.	DO	izvrši	du:
10.	DOUBLE PRECISION	dvostruka tačnost	dʌbl presižen
11.	EQVIVALENCE	odgovarajući	ikwivələns
12.	EXTERNAL	spoljašnji	ikstə:nəl
13.	FORMAT	raspored	fɔ:mat
14.	FUNCTION	funkcija	fʌnkšən
15.	GO TO	predji na	gou tu
16.	IF	ako	if
17.	INTEGER	ceo broj	intidžə
18.	LOGICAL	logički	lodžikal
19.	PAUSE	pauza	pɔ:z
20.	READ	čitaj	ri:d
21.	REAL	realan	riəl
22.	RETURN	povratak	ritə:n
23.	STOP	zaustavi	stop
24.	SUBROUTINE	potprogram	sʌbru:ti:n
25.	WRITE	piši	rait

4. ALGORITMI SA REALNIM KONSTANTAMA I PROMENLJIVIM

Svaki računski proces može se raščlaniti na niz formula, po kojima se vrši izračunavanje medjurezultata i konačnih rezultata. Formula u uobičajenoj matematičkoj notaciji sadrži promenljive i konstante medjusobom povezane aritmetičkim operacijama. Promenljivim, koje se nalaze na desnoj strani neke formule, moraju biti dodeljene brojne vrednosti pre računanja po toj formuli. Ove promenljive dobijaju brojne vrednosti, pre početka rada po algoritmu, kao polazne veličine ili su to medjurezultati izračunavanja po prethodnim formulama.

Tako, formula

$$y = x_1^2 + x_2 (3,7 + 4 \cdot x_3) \qquad (4.1)$$

sadrži promenljive x_1, x_2, x_3, čije brojne vrednosti moraju biti poznate pre izračunavanja veličine y. Na desnoj strani formule (4.1) figurišu i konstante 2; 3,7 i 4. Izračunavanje po formuli (4.1) predstavlja izračunavanje vrednosti aritmetičkog izraza desno od znaka jednakosti, i dodeljivanje izračunate brojne vrednosti promenljivoj y, na levoj strani znaka jednakosti.

Sve ovo je opšte poznato u matematici i predstavlja uobičajeni način korišćenja formula od strane širokog broja ljudi različitih profesija.

U ovoj glavi će biti izloženo kako se pišu formule u FORTRAN-jeziku. Imajući u vidu šta sve sadrže formule, to se u FORTRAN-jeziku moraju definisati sledeći pojmovi:

- kako se pišu konstante i promenljive,
- kako se pišu aritmetički izrazi,
- kako se dodeljuju vrednosti promenljivim,

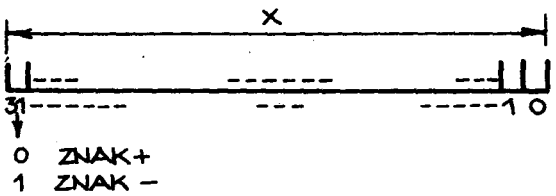
- kako se izračunavaju medjurezultati i rezultati, i
- kako se izdaju konačni rezultati.

4.1. Definicija brojne konstante

U FORTRAN-jeziku brojna konstanta može biti ceo broj ili mešovit broj. Ova stroga podela konstanti na cele i mešovite uslovljena je načinom registrovanja konstanti u memoriji računara. Celobrojna konstanta registruje se kao ceo binarni broj, a mešovita konstanta se registruje kao broj u pokretnom zarezu.

4.1.1. Celi brojevi

Ceo broj se piše kao niz dekadnih cifara, ispred kojeg može stajati znak + za pozitivan broj, a obavezno znak - za negativan broj. Ovako zapisana celobrojna konstanta mora biti u brojnom intervalu $[-2^{31}, 2^{31}-1]$. Ovaj brojni interval je odredjen kapacitetom jednog memorijskog registra (sl. 4.1.1.). Registar sadrži 32 ćelije označene, na sl. 4.1.1, sa 0, 1, ..., 31. Sadržaj ćelije 31 je najveće težine i registruje znak broja. Ceo broj x



Sl. 4.1.1.

registrovan u ovakvom registru mora biti u intervalu

$$-2^{31} \leq x \leq 2^{31}-1 = 2.147.483.647 \qquad (4.1.1)$$

Iz izložene defincije za pisanje celih brojeva sledi da

- ceo broj ne sadrži decimalnu tačku, i
- izmedju cifara celog broja ne može stajati medjuprostor.

Primeri

a) Dozvoljen oblik celog broja:

-386

65

0

2147483647

b) Nedozvoljen oblik celog broja:

35.0

35 000

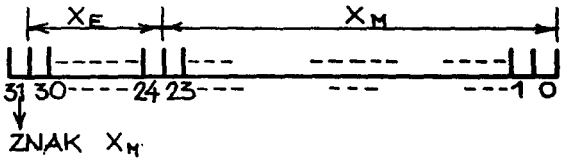
3147483647

4.1.2. Mešoviti brojevi

Mešovit broj se može pisati na dva načina, kao:

a) Niz dekadnih cifara, pri čemu se celobrojni i razlomljeni deo razdvajaju decimalnom tačkom. Ispred ovakvog niza može stajati znak + za pozitivne brojeve, a mora stajati znak - za negativne brojeve. Ovako zapisan mešovit broj mora imati najmanje jednu cifru, a najviše 7 dekadnih cifara.

b) Oblik kao pod a) iza kojeg se piše slovo E, a zatim se navodi ceo dvocifreni dekadni broj, koji predstavlja eksponent broja 10. Brojna vrednost ovako zapisane konstante jednaka je proizvodu brojeva ispred slova E i stepena broja 10, sa celobrojnim eksponentom navedenim iza slova E.



Sl. 4.1.2.

Mešoviti broj se registruje u memoriji računara u obliku pokretnog zareza (sl. 4.1.2.). Ovako registrovan broj x ima brojnu vrednost

$$x = x_M \cdot 16^{x_E} \qquad (4.1.2)$$

gde se eksponent x_E registruje kao ceo binarni broj, pa je

$$-64 \leq x_E \leq 63 \qquad (4.1.3)$$

a mantisa x_M registruje se u binarno kodiranom heksadekadnom sistemu, tako da je

$$0 \leq |x_M| \leq 1 - 16^{-6} \qquad (4.1.4)$$

Zamenom (4.1.3) i (4.1.4) u (4.1.2) lako se dobija da je brojni interval za broj x u pokretnom zarezu

$$0 \leq |x| \leq (1-16^{-6}) \cdot 16^{63} \qquad (4.1.5)$$

Najmanja vrednost mantise različita od nule jeste 16^{-1}, tako da svi brojevi manji po apsolutnoj vrednosti od 16^{-65} u računaru se registruju kao nule. Prema tome, broj $x \neq 0$, mora biti u intervalu

$$16^{-65} \leq |x| \leq 1-16^{-6}) \cdot 16^{63} \qquad (4.1.6)$$

ili u dekadnom brojnom sistemu

$$5,4 \cdot 10^{-79} \leq |x| \leq 7,2 \cdot 10^{75} \qquad (4.1.7)$$

Mešoviti brojevi, različiti od nule, zapisani u jednom od dozvoljenih oblika (a) ili (b) moraju po brojnoj vrednosti pripadati dozvoljenom intervalu (4.1.7). Brojevi ispod donje granice intervala registruju se kao nule, a brojevi iznad gornje granice prouzrokuju prekoračenje kapaciteta registra.

Primeri

a) Dozvoljeni oblici mešovitih brojeva:

```
  0.345
-22.0
482.57E-2
  -.12
 15.E15
 25.E0
```

b) Nedozvoljeni oblici mešovitih brojeva

2.000.596
-15.E136
25.E
1.876,00

4.2. Definicija realne promenljive

4.2.1. Ime promenljive

U matematici je uobičajeno da se promenljive označavaju jednim slovom azbuke. U FORTRAN-jeziku niz simbola označavaju promenljivu. Ovakav niz simbola zove se ime promenljive. Niz simbola koji čine ime promenljive mora ispunjavati sledeće uslove:

- prvi simbol mora biti veliko slovo engleske azbuke ili specijalni znak $,
- ostali simboli mogu biti: velika slova engleske azbuke, cifre dekadnog brojnog sistema ili specijalni znaci $
- broj simbola koji čine ime promenljive može biti od 1 do 6.

Primeri

a) Dozvoljena imena promenljivih:

A
AAA
A12
CDE15
$A
MASA
ZAGREB

b) Nedozvoljena imena promenljivih:

ABC1372
12A
A18.A

4.2.2. Vrsta promenljive po unutrašnjoj konvenciji

Vrsta promenljive odredjuje se prema tome kakva brojna vrednost se može dodeliti promenljivoj. Svaka promenljiva mora biti definisana po vrsti. To znači da promenljiva dobija ili celobrojne vrednosti ili mešovite brojne vrednosti (brojeve u pokretnom zarezu). Ako promenljiva uzima samo celobrojne vrednosti zove se celobrojna promenljiva, a ako uzima vrednosti mešovitih brojeva zove se realna promenljiva.

Vrsta promenljive, po unutrašnjoj konvenciji FORTRAN-jezika, definiše se na sledeći način:

- ako ime promenljive počinje slovom I, J, K, L, M ili N, to je celobrojna promenljiva,
- ako ime promenljive ne počinje jednim od navedenih slova, to je realna promenljiva.

Svakoj promenljivoj u FORTRAN-programu pre izvršenja programa na računaru, dodeljuje se jedan registar u kojem će se čuvati brojna vrednost promenljive. Ako je promenljiva celobrojna njena brojna vrednost će se registrovati u odgovarajućem registru kao ceo broj. Ako je promenljiva realna njena brojna vrednost će biti registrovana kao broj u pokretnom zarezu.

Primeri

a) Celobrojne promenljive po unutrašnjoj konvenciji

I
J
I19
IAB
MASA
NETO

b) Realne promenljive po unutrašnjoj konvenciji

A
B

CENA
BRUTO
C1846

4.3. Aritmetički izraz

Aritmetički izraz čine jedan argument ili više argumenata medju sobom razdvojenih znacima aritmetičkih operacija. Argument aritmetičkog izraza je konstanta ili promenljiva.

4.3.1. Aritmetičke operacije

Aritmetičke operacije su:

+ sabiranje,
- oduzimanje,
* množenje,
/ deljenje i
** stepenovanje.

Aritmetički izraz se piše kao niz, koji se sastoji od naizmeničnog smenjivanja argumenata i aritmetičkih operacija, pri čemu:

- niz počinje sa argumentom ili znakom minus (-), koji označava promenu znaka prvom argumentu, i
- niz se završava argumentom.

Vrednost aritmetičkog izraza izračunava se sleva na desno, pri čemu važi prioritet aritmetičkih operacija, prikazan u tabeli 4.3.1, gde je sa 1 označen najviši prioritet, a sa pz operacija promene znaka argumentu.

Tabela 4.3.1

Prioritet	Aritmetička operacija	Izvršava se
1	**,pz	sdesna na levo
2	*,/	sleva na desno
3	+,-	sleva na desno

Aritmetička operacija izmedju dva argumenta, koja su celi brojevi, daje kao rezultat ceo broj. Tako, ako se dele dva cela broja rezultat je samo celobrojni deo količnika.

Rezultat aritmetičke operacije izmedju argumenta od kojih je jedan realan, a drugi celobrojni, ili su oba realna, jeste realan.

Za operaciju stepenovanja treba imati u vidu kako se ona realizuje na računaru:

a) Ako je izložilac stepena ceo broj, tada se operacija stepenovanja svodi na množenje. Prema tome, stepen

$$a^4$$

se računa, kao

$$a \cdot a \cdot a \cdot a$$

b) Ako je izložilac stepena realan broj, tada se vrednost stepena računa logaritmovanjem i antilogaritmovanjem. Tako ako treba izračunati

$$a^{2,5}$$

to se izračunava kao

$$\text{antiln}(2,5 \text{ ln } a)$$

Iz ovoga sledi da se u operaciji stepenovanja, kada je izložilac realan, ne može pojaviti negativan broj kao osnova jer operacija logaritmovanja nije definisana za negativne brojeve.

c) Niz operacija stepenovanja izvršavaju se s desna na levo. Izraz

A**B**C

računa se tako što se najpre odredi (B ** C), a zatim A** (B**C). Ovo je različito od (A**B) **C.

d) Promena znaka ima isti prioritet kao i stepenovanje tako ako se napiše izraz

- A ** B

ovo će biti izračunato kao - (A ** B), a ne kao (-A)**B.

U tabeli 4.3.2. navedeni su primeri aritmetičkih izraza i njihovi ekvivalenti u matematičkoj notaciji.

Tabela 4.3.2.

ARITMETIČKI IZRAZI	
U FORTRANU	U MATEMATICI
A*B+C*D	A.B+C.D
A/B*C	$\frac{A}{B}$ C
-A*B**C	$(-A).B^{C}$
-A**2+B*C-5.6	$-A^{2}+B.C-5,6$
3.*X**3+X**2-20.	$3X^{3}+X^{2}-20$

4.3.2. Upotreba zagrada

Ako utvrdjeni prioritet aritmetičkih operacija ne odgovara aritmetičkom izrazu koji se želi zapisati, mogu se koristiti zagrade. Prem tome, zagrade treba koristiti, kao i u matematici, kada se želi promeniti prioritet operacija. Deo aritmetičkog izraza u okviru otvorene i zatvorene male zagrade dobija najviši prioritet. Ako postoji veći broj zagrada, unutrašnja zagrada je najvišeg prioriteta. Deo aritmetičkog izraza izmedju zagrada, odnosi se prema aritmetičkom izrazu u celini kao jedan argument.

U tabeli 4.3.3. navedeni su primeri aritmetičkih izraza sa zagradama i njihovi ekvivalenti u matematičkoj notaciji.

4.3.3. Vrsta aritmetičkog izraza

Izračunata brojna vrednost, koja se dobija kao rezultat aritmetičkog izraza po vrsti jeste:

- ceo broj, ako su svi argumenti aritmetičkog izraza celobrojne konstante ili celobrojne promenljive, odnosno

- realan, broj, ako je barem jedan argument aritmetičkog izraza realna konstanta ili realna promenljiva.

Tabela 4.3.3

ARITMETIČKI IZRAZI	
U FORTRANU	U MATEMATICI
A*(B+C)/D	$\frac{A(B+C)}{D}$
-(A+B)**2*D-4.	$-(A+B)^2.D-4$
X*(Y/(Y+X))	$X.\frac{Y}{Y+X}$
((-A)**2+A*(D/B))**2	$((-A)^2+A.\frac{D}{B})^2$
A/(B+C)**2-4./D+3.	$\frac{A}{(B+C)^2}-\frac{4}{D}+3$

U tabeli 4.3.4. dati su primeri aritmetičkih izraza sa oznakom vrste aritmetičkog izraza.

Tabela 4.3.4

ARITMETIČKI IZRAZ	VRSTA
I**4+3*MN1	ceo broj
(J/12)*K/3	ceo broj
(J/12)*K/3.	realan broj
A**3-M**2	realan broj
-I*(2.+K)/A	realan broj

4.4. Dodeljivanje brojne vrednosti promenljivoj

4.4.1. Aritmetička naredba

Opšti oblik aritmetičke naredbe je:

$$a = \psi \qquad (4.4.1)$$

gde je

a - ime promenljive,

ψ - aritmetički izraz.

Ova naredba ima sledeće dejstvo: vrednost aritmetičkog izraza ψ, pretvara se u vrstu brojnog podatka, saglasno vrsti promenljive a, i dodeljuje se promenljivoj a.

U aritmetičkom izrazu ψ ne može se pojaviti promenljiva kojoj nije dodeljena brojna vrednost pre izvršavanja naredbe (4.4.1).

Treba uočiti razliku izmedju znaka jednakosti u aritmetičkoj naredbi, i uobičajenog značenja u matematici. Znak jednakosti u aritmetičkoj naredbi opisuje proces koji se sastoji od:

- izračunavanja vrednosti aritmetičkog izraza, na desnoj strani znaka jednakosti, prema konkretnim vrednostima promenljivih,
- dovodjenja izračunate brojne vrednosti promenljive u celobrojni ili realni oblik, u saglasnosti sa vrstom promenljive na levoj strani znaka jednakosti, i
- dodeljivanja ovako dobijeńe brojne vrednosti, promenljivoj na levoj strani znaka jednakosti.

Sve navedene faze u izvršavanju aritmetičke naredbe dogadjaju se jedna za drugom. Prema tome, u fazi izračunavanja vrednosti aritmetičkog izraza može se koristiti brojna vrednost promenljive, kojoj će u zadnjoj fazi biti dodeljena izračunata brojna vrednost. Kako se brojna vrednost promenljive čuva u odredjenom memorijskom registru, to u ovom slučaju znači da sadržaj ovog registra može biti korišćen u fazi izračunavanja vrednosti aritmetičkog izraza, a zatim uništen upisom nove brojne vrednosti koja se dodeljuje promenljivoj. Tako se može pisati

$$A = A+B \qquad (4.4.2)$$

što znači: sabrati brojne vrednosti promenljivih A i B i dobijeni rezultati dodeliti kao novu brojnu vrednost promenljivoj A.

Primeri

1) Primeri aritmetičkih naredbi:

```
AB=C**2+AB+4.36

I=J+A**2-4

BRZINA=PUT/VREME

VREDN=CENA*KOLIC
```

2) Aritmetička naredba

```
A=I/3-4*B
```

za I = 10 i B = 3.5 dodeljuje promenljivoj A brojnu vrednost - 11.0.

4.4.2. Naredba ulaza

Aritmetička naredba dodeljuje brojnu vrednost promenljivoj, po izračunatoj vrednosti aritmetičkog izraza. Medjutim, neke promenljive u algoritmu dobijaju početne vrednosti prema konkretnom primeru koji se rešava. Ovakve veličine zovu se ulazne veličine. Tako, algoritam na sl.1.3.5. sadrži ulazne veličine x_1 i x_2, kojima moraju biti dodeljene brojne vrednosti na početku izvršavanja algoritma. U istom algoritmu promenljiva i dobija brojnu vrednost preko aritmetičke naredbe. Prema tome, postojanje promenljivih koje dobijaju brojne vrednosti na početku algoritma, omogućuje primenu algoritma za različite konkretne brojne vrednosti ovih promenljivih. Algoritam u kojem ne postoje promenljive, kojima se brojna vrednost može dodeliti kao veličina koja ulazi u algoritam, predstavlja niz izračunavanja koji pri svakom izvršavanju algoritma dovodi do istog rezultata. Jasno je da ovakav algoritam retko ima praktičnog smisla. Najčešće je potrebno sastaviti algoritam koji će izračunavati rezultate za različite polazne podatke. Polazni podaci nalaze se na spoljnim nosiocima informa-

macija i program koji ih koristi dodeljuje njihove brojne vrednosti odgovarajućim promenljivim. Naredba koja omogućuje ovakvo dodeljivanje brojnih vrednosti promenljivim zove se naredba ulaza. Ova naredba se piše u obliku

READ(i, j)lista (4.4.3)

gde je

READ - službena reč, koja označava da se radi o naredbi ulaza,

i - celobrojna konstanta bez znaka ili ime celobrojne promenljive, kojoj mora biti dodeljena brojna vrednost pre izvršavanje naredbe (4.4.3),

j - obeležje jedne naredbe FORMAT,

lista - spisak imena promenljivih, medju sobom razdvojenih zarezima, kojima se dodeljuju brojne vrednosti sa ulaza.

Ako je spoljni nosilac informacija kartica, tada veličina i ukazuje na čitač kartica sa kojeg će biti čitani brojni podaci. Kod računara IBM -360/44, kada se vrši ulaz preko čitača kartica, treba uzeti da je i = 5. Naredba ulaza je izvršna naredba i ima sledeće značenje: pročitati ulazne podatke sa ulaznog uredjaja i, pod kontrolom opisne naredbe j, i dodeliti pročitane brojne vrednosti promenljivim, navedenim u listi.

Prema tome, izvršna naredba sadrži informaciju o tome gde se nalaze ulazni podaci (i), i kojim promenljivim se dodeljuju (lista). Medjutim kako izgledaju brojne vrednosti na kartici to nije rečeno izvršnom naredbom ulaza. Brojni podatak na kartici nalazi se u obliku konstante, a konstante su sastavljene od niza simbola FORTRAN-jezika. Svaki simbol konstante buši se u jednu kolonu kartice. Više kolona, koje zauzima jedna konstanta na kartici, čine polje. Jedno ili više polja, koja se sa jedne kartice unose u memoriju računara, čine slog. Opisna naredba, kojom se opisuje izgled sloga, piše se u obliku

j FORMAT(slog) (4.4.4)

gde je

j - obeležje naredbe,

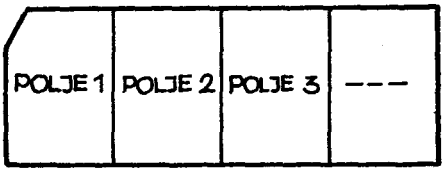
FORMAT - službena reč, koja ukazuje da se radi o naredbi za opis podataka, i

slog - niz opisa, medju sobom razdvojenih zarezima, kojima se opisuju pojedina polja ulaznog sloga.

Ulazni slog je sastavljen od više polja, a svako polje sadrži jednu konstantu. Za opis ulaznog sloga potrebno je opisati svako polje u okviru sloga. Opis polja zavisi od konstante koja se nalazi registrovana u polju. Neka 3 polja čine jedan ulazni slog. Tada se opis sloga sastoji od 3 opisa pojedinih polja, tj. naredba (4.4.4) dobija oblik

j FORMAT($opis_1$, $opis_2$, $opis_3$) (4.4.5)

Opisi pojedinih polja navode se u naredbi (4.4.5) sleva na desno,onako kako slede na kartici (sl. 4.4.1).



Sl. 4.4.1

Da bi se u nizu naredbi koje čine program omogućilo jednoznačno ukazivanje na odredjenu naredbu programa, uvodi se obeležje naredbe. Obeležje naredbe je jednocifren do petocifren ceo neoznačen dekadni broj.

Obeležje naredbe se piše sa leve strane naredbe. Svaka FORMAT-naredba mora imati obeležje (označeno sa j u 4.4.4 i 4.4.5), koje se obavezno navodi u naredbi ulaza (4.4.3), odnosno izlaza (4.5.1). Izvršne naredbe FORTRAN-programa mogu po potrebi imati obeležja. U jednoj programskoj jedinici ne može se jedno obeležje koristiti, kao obeležje, više od jedne FORTRAN-naredbe.

4.4.2.1. Opis celih brojeva

Ako polje sadrži celobrojnu konstantu, opisuje se na sledeći način

Ik (4.4.6)

gde je

I - simbol FORTRAN-jezika, koji označava da se radi o celom broju (Integer),

k - neoznačen ceo broj, koji ukazuje na broj kolona polja na kartici.

Ako je broj k takav, da polje sadrži veći broj kolona nego što to zahteva broj cifara brojnog podatka, tada odgovarajući broj kolona na levoj

strani polja ostaje nebušen. Tako, ako u polju sa opisom I10 koje sadrži 10 kolona, od 1. do 10. kolone, treba registrovati broj -386, to u kolonama od 1. do 6. neće biti ništa bušeno, a -386 će se bušiti sleva nadesno od 7. do 10. kolone.

4.4.2.2. Opis mešovitih brojeva

Ako polje na kartici sadrži mešoviti broj, opisuje se sa

$$Fk.d \qquad (4.4.7)$$

gde je

F - simbol FORTRAN-jezika, koji označava da se radi o mešovitom broju (Fixed point),

k - neoznačen ceo broj, koji ukazuje na broj kolona polja na kartici,

d - broj decimalnih mesta brojnog podatka.

Broj k mora biti tako odredjen da zadovoljava uslov

$$k \geq c + d + 2 \qquad (4.4.8)$$

gde je

c - broj cifara celobrojnog dela broja,

d - broj cifara razlomljenog dela broja,

2 - jedno mesto za znak broja, i jedno mesto za decimalnu tačku.

Decimalna tačka se može izostaviti na kartici, i pri tome će biti odredjena opisom (4.4.7). Tako u ovom slučaju relacija (4.4.8) dobija oblik

$$k \geq c + d + 1 \qquad (4.4.9)$$

Medjutim, ako postoji decimalna tačka na kartici, ali nije u saglasnosti sa opisom (4.4.7), tada će biti prihvaćeno mesto decimalne tačke na kartici, a ne u opisu (4.4.7). Tako, opis F8.0 će upisivati brojne podatke u memoriju računara sa brojem decimalnih mesta, zadatih decimalnom tačkom u odgovarajućem polju kartice.

Ako se u polju registruju samo pozitivni brojevi, može se izostaviti mesto za znak broja, a ako se izostavi i mesto za decimalnu tačku, to se relacija (4.4.9) svodi na

$$k \geq c + d \qquad (4.4.10)$$

Za registrovanje vrlo malih ili velikih brojeva oblik (4.4.7) je nepogodan, jer zahteva navodjenje svih cifara broja. U ovom slučaju pogodno je koristiti oblik

Ek.d (4.4.11)

gde je

E - simbol FORTRAN-jezika, koji ukazuje da se radi o mešovitom broju, zapisanom u eksponencijalnom obliku (Exponential form),

k - neoznačen ceo broj koji ukazuje na broj kolona polja na kartici,

d - broj decimalnih mesta brojnog podatka.

Konstanta zapisana u eksponencijalnom obliku registruje se u polju kartice, kao i ranije opisana mešovita konstanta samo što se iza decimalnih mesta broja navodi slovo E, a iza slova E izložilac broja 10. Tako se može pisati

17.83E-15

što odgovara decimalnom broju $17,83 \cdot 10^{-15}$. Broj zapisan u eksponencijalnom obliku zahteva polje sa

$$k \geq c + d + 6 \qquad (4.4.12)$$

gde je

c - broj cifara celobrojnog dela broja,

d - broj cifara razlomljenog dela broja,

6 - jedno mesto za znak broja, jedno mesto za decimalnu tačku, slovo E, jedno mesto za znak eksponenta i dva mesta za dekadne cifre eksponenta.

Znak eksponenta se može izostaviti, ako je eksponent pozitivan broj. Slovo E se može izostaviti, ali se pri tome obavezno mora pisati znak eksponenta. Tako se mogu registrovati sledeće konstante na kartici:

-2.147E20
-2.147+20
+8.14E-02
8.14-02

Mešoviti brojevi zapisani bez slova E i eksponenta mogu biti uneti pomoću opisa (4.4.11), pri čemu će se smatrati da je eksponent nula.

Tako se može pisati

20.156E0

20.156+0

20.156

Svi navedeni primeri će u memoriji računara biti registrovani u obliku pokretnog zareza, i predstavljaće brojnu vrednost 20,156.

4.4.2.3. Opis praznog polja

Vrlo često je potrebno neke kolone kartice preskočiti. Da bi se ovo omogućilo uveden je opis

nX (4.4.13)

gde je

X - simbol FORTRAN-jezika,koji označava da se radi o opisu praznog polja,

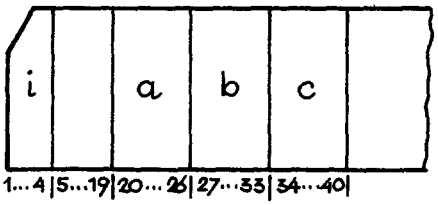
n - ceo neoznačen broj, koji ukazuje koliko kolona sadrži polje koje treba preskočiti.

Primer

Za zadate vrednosti promenljivih i , a , b i c izračunati vrednost

$y = (a^i + 32,4)(b-2c)$

Veličina i je ceo trocifren broj i nalazi se na kartici od 1. do 4. kolone. Veličine a, b i c sadrže 3 cela i 2 decimalna mesta i nalaze se na kartici od 20. do 40. kolone, pri čemu svaka zauzima polje od po 7 kolona. Izgled kartice je prikazan na sl. 4.4.2.



Sl. 4.4.2

Program na FORTRAN-jeziku ima sledeći izgled:

```
      READ(5,10) I,A,B,C
   10 FORMAT(I4,15X,F7.2,F7.2,F7.2)
      Y=(A**I+32.4)*(B-2.*C)
```

U naredbi FORMAT opisana su sva polja na kartici, sleva na desno, koja čine ulazni slog, uključujući i polje koje se preskače od 5. do 19. kolone. Promenljive I, A, B i C dobijaju brojne vrednosti sleva nadesno, kako su zapisane u listi naredbe READ. Konstanta 2, u aritmetičkoj naredbi zapisana je sa decimalnom tačkom, što znači da će u memoriji računara biti registrovana u obliku pokretnog zareza. Na ovaj način, operacija množenja 2.*C izvodi se izmedju argumenata u pokretnom zarezu. Ako bi konstanta bila zapisana bez decimalne tačke, tj. aritmetička naredba u obliku

```
Y=(A**I+32.4)*(B-2*C)
```

tada bi konstanta 2 bila registrovana u memoriji računara kao ceo broj. Ovo bi značilo da pre izvodjenja operacije množenja 2*C, ova konstanta mora biti prevedena u oblik pokretnog zareza, a potom izvršena aritmetička operacija množenja. I jedan i drugi oblik aritmetičke naredbe je korektan i dovodi do istog rezultata. Medjutim, prvi oblik (sa decimalnom tačkom) predstavlja bolji zapis, jer će takav aritmetički izraz biti brže izračunat pri izvršavanju programa na računaru.

4.5. Izdavanje brojne vrednosti promenljive

Brojna vrednost promenljive registrovana je u memoriji računara u binarnom brojnom sistemu, kao ceo broj ili kao broj u pokretnom zarezu. Binarni oblik broja je nepogodan za korišćenje od strane šireg broja ljudi koji su korisnici računara. Zato je potrebno broj prevesti iz binarnog u dekadni brojni sistem, i rezultate izdati u dekadnom brojnom sistemu. Potpuna informacija o izdavanju brojnih vrednosti promenljivih, na izlazni organ, sadrži sledeće:

- imena promenljivih čije se brojne vrednosti žele izdati, kao i
- oblik izdavanja brojnih vrednosti.

Prvi deo informacije, o imenima promenljivih čije se brojne vrednosti žele izdati, zadaje se izvršnom naredbom izlaza

WRITE (i, j)lista (4.5.1)

gde je

WRITE - službena reč, koja ukazuje da se radi o izdavanju brojnih vrednosti promenljivih;

i - celobrojna konstanta bez znaka ili celobrojna promenljiva, kojoj je dodeljena brojna vrednost pre izvršenja naredbe (4.5.1). Ova brojna vrednost odredjuje izlazni uredjaj na kojem se vrši izdavanje rezultata. Za računar IBM-360/44, kada se izlaz vrši na bušaču kartica to je broj 7, a na štampaču broj 6,

j - obeležje jedne naredbe FORMAT;

lista - spisak imena promenljivih, medju sobom razdvojenih zarezima, čije se brojne vrednosti izdaju.

Svakoj izvršnoj naredbi izlaza, kao što je naredba (4.5.1) pridružuje se jedna opisna FORMAT-naredba, koja sadrži informacije o obliku izdavanja brojnih vrednosti promenljivih. Oblik izdavanja zadaje se istim opisima koji su služili za opis podataka na kartici, samo što sada opisuju izgled štampanog dokumenta ili bušene kartice na izlazu. Jedan ili više podataka koji se prenose na izlazni organ čine izlazni slog. Dužina izlaznog sloga zavisi od nosioca informacija na kojem se vrši upis informacija na izlaznom organu računara. Ako se izlaz vrši na bušaču kartica, tada je maksimalna dužina izlaznog sloga 80 simbola, koji se mogu bušiti u 80 kolona jedne kartice. Ako se izlaz vrši na paralelnom štampaču, tada je dužina izlaznog sloga 120 tipografskih simbola (ovaj broj može biti i veći kod nekih tipova štampača), koji čine jedan red na štampanom dokumentu.

Opisna naredba, koja se pridružuje naredbi izlaza ima oblik

j FORMAT(slog) (4.5.2)

gde je

j - obeležje naredbe koje se navodi u izvršnoj naredbi izlaza (4.5.1),

FORMAT - službena reč, koja ukazuje na vrstu opisne naredbe,

slog - niz opisa koji definišu izlazni slog.

Izdavanje izlaznog sloga na paralelnom štampaču zahteva informaciju o vertikalnom pomeranju papira na štampaču. Ova informacija je sadržana u prvom simbolu izlaznog sloga. Kao prvi simbol izlaznog sloga može se pojaviti jedan od simbola +, b, 0 ili 1 pri čemu imaju sledeće značenje:

+ - bez pomeranja papira,

b - pomeranje papira za 1 novi red,

0 - pomeranje papira za 2 nova reda, i

1 - pomeranje papira na prvi red sledeće strane.

Jedan od navedena 4 simbola mora se nalaziti na početku izlaznog sloga. Ako ovaj simbol nije obezbedjen prvim opisom u izlaznom slogu, treba ga navesti izmedju apostrofa kao jedan od opisa izlaznog sloga.

Tako naredba

j FORMAT ('b', I3, 2X, E12.5) (4.5.3)

formira izlazni slog od 4 polja, pri čemu prvo polje sadrži jedan simbol blanko, drugo polje sadrži tri simbola za vrednost celobrojne promenljive, treće polje sadrži dva blanka i četvrto polje 12 simbola za brojnu vrednost realne promenljive. Prema tome, izlazni slog sadrži 17 simbola. Kada se ovako formiran izlazni slog pošalje na paralelni štampač, prvi simbol biće upotrebljen kao komandni simbol za vertikalno pomeranje papira na štampaču, i u ovom slučaju proizvešće novi red na štampanom dokumentu. Ostalih 16 simbola izlaznog sloga biće štampani od početka reda sleva na desno kako slede u izlaznom slogu.

4.5.1. Opis celih brojeva

Izdavanje brojne vrednosti celobrojne promenljive opisuje se sa

Ik (4.5.4)

gde je

I - simbol koji ukazuje da se radi o celim brojevima,

k - ceo neoznačen broj, koji odredjuje broj mesta koji će zauzeti brojna vrednost na izlaznom nosiocu informacija. To je broj tipografskih simbola na štampanom dokumentu, kada se izlaz vrši na štampaču, odnosno, broj kolona kartice, kada se izlaz vrši na bušaču kartica.

Ako se izlaz vrši na štampaču i pri tome brojna vrednost koja se štampa prevazilazi dužinu k, odredjenu na štampanom dokumentu tada se štampa k zvezdica (*) u predvidjenom polju.

Najveći ceo broj koji može biti registrovan u memorijskom registru sadrži 10 dekadnih cifara, a ako se uzme u obzir i 1 mesto za znak, to znači da format I11 uvek obezbedjuje korektno štampane vrednosti celobrojne promenljive. Brojna vrednost štampa se na desnoj strani predvidjenog polja za štampanje, a na levoj strani nevažeće nule štampaju se kao medjuprostori (blanko). Znak broja štampa se neposredno levo od prve važeće cifre pri čemu se štampa znak (-) za negativne brojeve, a znak + se ne štampa. Prema tome, predvidjanjem veće dužine polja od one koja je potrebna za štampanje brojne vrednosti može se obezbediti potreban broj medjuprostora (blanka) izmedju brojeva koji se štampaju u jednom redu.

Primer

Na kartici su zadata tri cela broja x, y i z. Broj x je bušen od 10. do 15., broj y od 20. do 24., a broj z od 52. do 54. kolone jedne kartice. Uneti brojeve sa kartice i štampati na paralelnom štampaču. U ovom slučaju potrebne su sledeće FORTRAN naredbe:

```
   READ(5,40)IX,IY,IZ
40 FORMAT(9X,I6,4X,I5,27X,I3)
   WRITE(6,41)IX,IY,IZ
41 FORMAT(' ',I6,I8,I6)
```

U naredbi FORMAT sa obeležjem 40, opisi 9X, 4X i 27X definišu prazna polja na kartici izmedju brojeva x, y i z. Naredba WRITE obezbedjuje štampanje brojnih vrednosti promenljivih IX, IY i IZ redom sleva nadesno po opisima u FORMAT-naredbi sa obeležjem 41. Tako će brojne vrednosti promenljivih IX i IZ biti štampane po opisu I6, a promenljive IY

po opisu I8. Kako brojna vrednost promenljive IY sadrži 5 simbola na kartici, a opis I8 odredjuje polje od 8 simbola, to će tri simbola sa leve strane polja biti neiskorišćena za prikazivanje broja i služiće zá razmak izmedju brojeva na štampanom dokumentu. Slično razmatranje važi i za promenljivu IZ. Štampani dokumenat će imati sledeći izgled

XXXXXXbbbXXXXXbbbXXX

gde je

X - cifra dekadnog brojnog sistema, medjuprostor (b) ili specijalni znak (-),

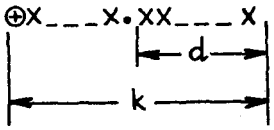
b - medjuprostor (blanko).

4.5.2. Opis mešovitih brojeva

Mešoviti broj registruje se u memoriji u obliku poketnog zareza. Ako se za izdavanje vrednosti realne promenljive koristi opis

Fk.d (4.5.5)

tada će brojna vrednost na izlazu biti u obliku:



gde je ⊕ mesto za znak broja, i to: simbol - za negativne brojeve i simbol b za pozitivne brojeve. Oblik (4.5.5) izdaje vrednosti promenljivih u vidu celobrojnog i razlomljenog dela. Celobrojni deo sadrži maksimum k - - d - 1 tipografsko mesto, ako je broj pozitivan, a k - d - 2 tipografska mesta, ako je broj negativan. Razlomljeni deo sadrži d dekadnih cifara. Prema tome, broj k je

$$k \geq c + d + 2 \qquad (4.5.6)$$

gde je

c - broj cifara celobrojnog dela broja,

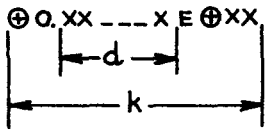
d - broj cifara razlomljenog dela broja,

2 - jedno mesto za znak broja (znak negativnog broja izdaje se kao simbol - , a znak pozitivnog kao simbol b), i jedno mesto za decimalnu tačku.

Ako se za izdavanje vrednosti realne promenljive koristi opis

Ek.d (4.5.7)

tada će brojna vrednost na izlazu biti u obliku



gde je

⊕ - mesto za znak broja, i to: simbol - za negativne brojeve, i simbol b za pozitivne brojeve,

X - cifra dekadnog brojnog sistema,

d - broj decimalnih mesta,

k - ukupan broj simbola.

Ukupan broj simbola za izdavanje brojne vrednosti sa opisom (4.5.7) je

$$k \geq d + 7 \qquad (4.5.8)$$

Ako u opisima (4.5.5) i (4.5.7) broj k definiše manji broj simbola od onog koji zahteva zapis brojne vrednosti, to će na predvidjenoj dužini k biti izdate zvezdice (*).

4.5.3. Opis praznog polja

Razmak izmedju brojeva, kao što smo već videli, može se ostvariti predvidjanjem većeg broja celih u opisima I, F ili E. Medjutim, opis

nX (4.5.9)

definiše na izlazu prazno polje od n medjuprostora, tako da se ovim opisom mogu definisati proizvoljni razmaci izmedju brojeva.

Primer

Na kartici se nalaze brojevi I, X i Y u sledećem rasporedu:

a) od 1. do 5. kolone ceo broj I (opis I5),

b) od 6. do 12. kolone mešoviti broj X (opis F7.2),

c) od 13. do 24. kolone mešoviti broj Y (opis E12.5).

Sastaviti program koji će uneti zadate brojeve sa kartice i štampati:

```
      READ(5,8) I,X,Y
    8 FORMAT(I5,F7.2,E12.5)
      WRITE(6,7) I,X,Y
    7 FORMAT(' ',I5,2X,F7.2,2X,E12.5)
```

Za I=18, X=-24.5 i Y=0.3E-20, štampani dokumenat ima izgled

```
bbb18bbb-24.50bbb0.30000E-20
|--->|-->|---->|-->|------->|
  I5   2X  F7.2   2X   E12.5
```

4.6. Proste linijske algoritamske strukture

4.6.1. Prekid rada po programu i fizički kraj programa

Dosadašnje izlaganje FORTRAN-jezika omogućuje zapis prostih linijskih algoritamskih struktura. Medjutim, za korektan zapis algoritama na FORTRAN-jeziku nedostaje mogućnost ukazivanja na zadnji algoritamski korak u algoritmu. Ovo se u FORTRAN-jeziku vrši naredbom

```
STOP                                    (4.6.1)
```

ili

```
STOP n                                  (4.6.2)
```

gde je

STOP - službena reč, koja označava kraj rada po programu,

n - jednocifreni do petocifreni ceo dekadni broj bez znaka.

U jednom programu može se nalaziti više naredbi STOP. Da bi se omogućio uvid kojom od više naredbi je završeno izvršavanje programa, to je uveden oblik (4.6.2), koji izdaje naredbu (4.6.2) na štampaču.

Program zapisan na FORTRAN-jeziku, prevodi se na mašinski jezik pre izvršavanja na računaru. Ovo prevodjenje vrši program za prevodjenje, u koji kao ulazni podaci ulaze naredbe FORTRAN-programa, a izlazne veličine su naredbe u mašinskom jeziku. Da bi program za prevodjenje dobio informaciju kada je završeno prevodjenje i zadnje naredbe FORTRAN-programa, uvodi se naredba

END (4.6.3)

koja se mora nalaziti na kraju svakog FORTRAN-programa.

4.6.2. Primeri algoritama sa prostim linijskim strukturama

Primer 1

Na jednoj kartici se nalaze brojne vrednosti promenljivih x_1, x_2, x_3, x_4 i x_5 u obliku F7.3. Izračunati rezultat y po formuli

$$y = \left[(x_1 + x_2)\cdot x_3 - x_4\right] \cdot x_5 \cdot x_1 \qquad (4.6.4)$$

Na izlazu štampati zadate brojeve x_i, i = 1, 2, ... 5, i rezultat y.

Na sl. 1.3.1. data je grafička shema algoritma, a FORTRAN-program ima sledeći izgled:

```
      READ(5,10) X1,X2,X3,X4,X5
   10 FORMAT(F7.3,F7.3,F7.3,F7.3,F7.3)
      Y=((X1+X2)*X3-X4)*X5*X1
      WRITE(6,11) X1,X2,X3,X4,X5,Y
   11 FORMAT(' ',F7.3,F9.3,F9.3,F9.3,F9.3,E19.7)
      STOP
      END
```

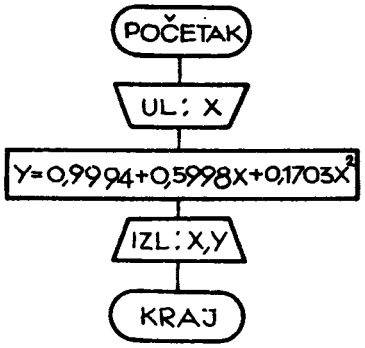
U naredbi FORMAT sa obeležjem 11 promenljiva X1 se štampa u istom obliku u kojem se nalazi na kartici. Ostale ulazne veličine se štampaju u obliku F9.3, čime su obezbedjena dva medjuprostora izmedju brojeva koji se štampaju. Rezultat Y se štampa u obliku E19.7, čime je obezbedjen razmak od 5 medjuprostora u odnosu na brojnu vrednost promenljive X5.

Primer 2

Sastaviti program koji za zadatu vrednost - $1 \leq x \leq 1$, izračunava $10^{x/4}$ po formuli

$$y = 10^{x/4} \approx 0,9994 + 0,5998\, x + 0,1703 \cdot x^2 \qquad (4.6.5)$$

Neka je vrednost argumenta x zadata od 1. do 6. kolone u obliku F6.3. Na izlazu štampati vrednost argumenta i vrednosti funkcije (4.6.5) u obliku E12.5. Na sl. 4.6.1. data je blok-shema algoritma.



Sl. 4.6.1

Program na FORTRAN-jeziku ima sledeći izgled:

```
      READ(5,10) X
   10 FORMAT(F6.3)
      Y=0.9994+0.5998*X+0.1703*X**2
      WRITE(6,15) X,Y
   15 FORMAT(' ',F6.3,2X,E12.5)
      STOP
      END
```

Navedeni primeri ilustruju programe sastavljene po zadatim algoritmima sa prostim linijskim strukturama. U toku jednog izvršavanja takvog programa, svaka naredba se izvrši jedanput. Naredbe se izvršavaju odozgo prema dole, kako slede u zapisanom nizu. Dolaskom na naredbu STOP prekida se dalji rad po programu.

Treba napomenuti da su navedeni primeri zapisanu u obliku programa koji predstavlja kompletan zapis algoritma na FORTRAN-jeziku. Svaka naredba ovakvog programa buši se u jednu karticu kako je to objašnjeno u odeljku 2.2.

4.7. Razgranate linijske algoritamske strukture

Niz naredbi koje čine prostu linijsku algoritamsku strukturu izvršavaju se jedna za drugom u zapisanom redosledu. Kod razgranatih linijskih algoritamskih struktura, mora postojati naredba kojom se redosled izvršavanja naredbi u programu može promeniti. Ovakve naredbe se zovu upravljačke naredbe. Postoje dve vrste upravljačkih naredbi: uslovne i bezuslovne. Uslovne upravljačke naredbe vrše prelazak na naredbu sa zadatim obeležjem, ako je navedeni uslov ispunjen, a bezuslovne upravljačke naredbe vrše uvek prelazak na naredbu sa zadatim obeležjem.

4.7.1. Uslovni prelazak po vrednosti aritmetičkog izraza

Vrednost aritmetičkog izraza ψ, može biti pozitivna, negativna ili nula. Naredba uslovnog prelaska po vrednosti aritmetičkog izraza ima oblik

$$IF(\psi)n_1, n_2, n_3 \qquad (4.7.1)$$

gde je

IF - službena reč, koja ukazuje da se radi o uslovnoj naredbi,

ψ - aritmetički izraz, a

n_i - obeležja izvršnih naredbi u FORTRAN-programu, i=1, 2, 3.

Naredba (4.7.1) ima sledeće značenje u zavisnosti od vrednosti aritmetičkog izraza:

a) $\psi < 0$, preći na naredbu sa obeležjem n_1,

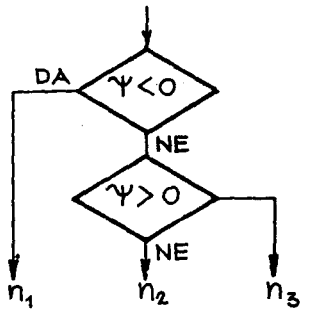
b) $\psi = 0$, preći na naredbu sa obeležjem n_2, i

c) $\psi > 0$, preći na naredbu sa obeležjem n_3.

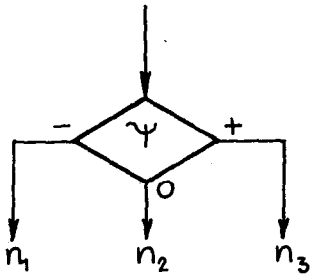
Kako vrednost aritmetičkog izraza mora biti manja, jednaka ili veća od nule, to znači da će se prelazak izvršiti uvek na jedno od obeležja n_1, n_2 ili n_3. Odavde sledi da naredba koja se nalazi neposredno ispod uslovne aritmetičke naredbe mora imati obeležje.

Na sl. 4.7.1. prikazani su grafički simboli uslovnih algoritamskih koraka koji odgovaraju naredbi (4.7.1). Pošto ova naredba omogućuje tri

izlaza u zavisnosti od vrednosti aritmetičkog izraza, to se u grafičkom prikazivanju može koristiti simbol prikazan na sl. 4. 7. 2 koji više odgovara mogućnostima naredbe (4. 7. 1).



Sl. 4. 7. 1



Sl. 4. 7. 2

4. 7. 2. Bezuslovni prelazak

Naredba IF omogućuje prelazak u zavisnosti od vrednosti aritmetičkog izraza. Vrlo često u programima treba bezuslovno promeniti redosled izvršavanja naredbi.

Ovo je omogućeno naredbom

GO TO n (4. 7. 2)

gde je

GO TO - službena reč, koja označava bezuslovni prelazak,

n - obeležje jedne izvršne FORTRAN-naredbe u programu.

4. 7. 3. Primeri razgranatih linijskih algoritamskih struktura

Primer 1

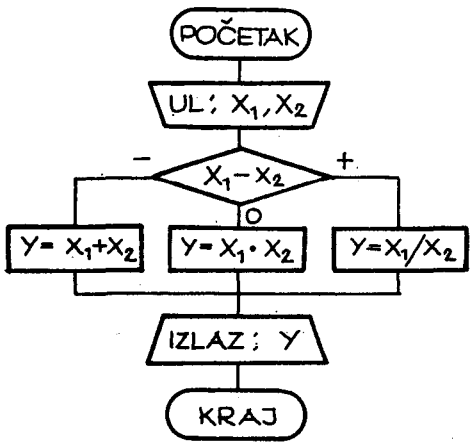
Za zadate vrednosti x_1 i x_2 izračunati y po formuli

$$y = \begin{cases} x_1 + x_2 & \text{ako je } x_1 < x_2 \\ x_1 \cdot x_2 & \text{ako je } x_1 = x_2 \\ x_1 / x_2 & \text{ako je } x_1 > x_2 \end{cases}$$

Brojevi x_1 i x_2 mogu imati dva cela i tri decimalna mesta. Prema tome, broj x_1 neka je bušen od 1. do 7. kolone, a x_2 od 8. do 14. kolone jedne kartice. Blok-shema algoritma prikazana je na sl. 1. 3. 5. Uvodjenjem grafičkog oblika sa sl. 4. 7. 2, algoritam se može prikazati kao na sl. 4. 7. 3.

FORTRAN-program sastavljen po algoritmu na sl. 4. 7. 3. ima izgled

```
      READ(5,100) X1,X2
  100 FORMAT(F7.3,F7.3)
      IF(X1-X2) 10,20,30
   20 Y=X1*X2
   40 WRITE(6,200) Y
  200 FORMAT(' ',E14.7)
      STOP
   10 Y=X1+X2
      GO TO 40
   30 Y=X1/X2
      GO TO 40
      END
```

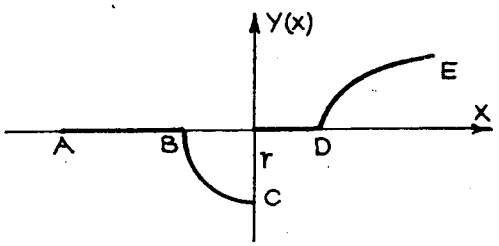


Sl. 4. 7. 3

Iz gornjeg primera se vidi da je korišćenjem naredbe GO TO izbegnuto ponavljanje naredbe WRITE, posle svakog izračunavanja promenljive Y, kao i naredbe STOP za prekid rada po programu.

Primer 2

Funkcija y(x) je zadata na sl. 4. 7. 4, što se može izraziti na



Sl. 4. 7. 4

sledeći način

$$y(x) = \begin{cases} 0 & \text{za} \quad -\infty < x \leq -r \\ -\sqrt{r^2 - x^2} & \text{za} \quad -r < x \leq 0 \\ 0 & \text{za} \quad 0 < x \leq r \\ \sqrt{x-r} & \text{za} \quad r < x < +\infty \end{cases}$$

Za $n \leq 999$ zadatih vrednosti argumenata $x_1, x_2, \ldots, x_n$, izračunati vrednosti funkcije $y(x)$.

a) Opis ulaznih podataka

Neka prva kartica sa ulaznim podacima sadrži od 1. do 3. kolone broj n, a od 4. do 12. broj r sa 4 decimalna mesta. Iza ove kartice dolazi n kartica i na svakoj od njih je bušena jedna vrednost argumenta x od 1. do 9. kolone, pri čemu argument x može imati 3 cela i 4 decimalna mesta.

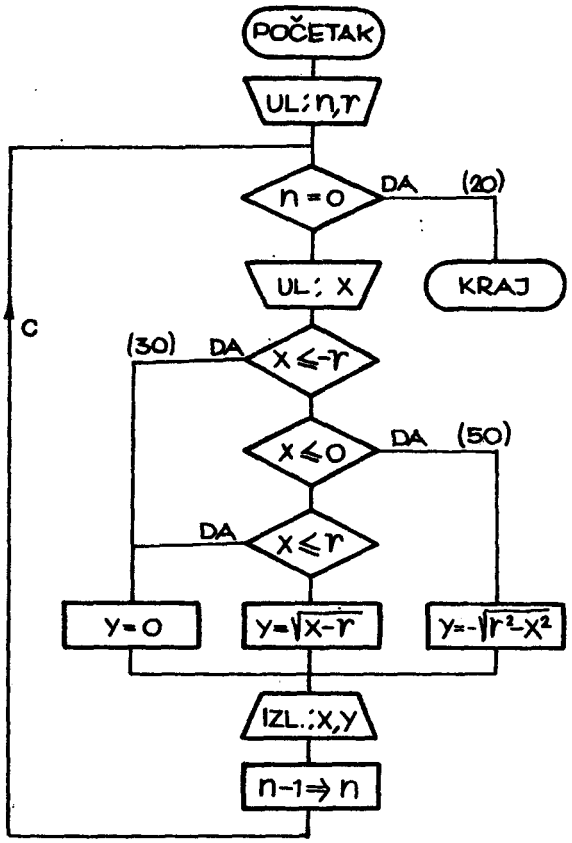
b) Raspored štampanja rezultata

Na izlazu formirati štampani dokumenat, koji će se sastojati od tabele:

x_1	$y(x_1)$
x_2	$y(x_2)$
.	.
.	.
.	.
x_n	$y(x_n)$

FORTRAN-program sastavljen po algoritmu na sl. 4. 7. 5 ima sledeći izgled:

```
      READ(5,100) N,R
  100 FORMAT(I3,F9.4)
   80 IF(N) 10,20,10
   10 READ(5,200) X
  200 FORMAT(F9.4)
      IF(X+R) 30,30,40
   40 IF(X) 50,50,60
   60 IF(X-R) 30,30,70
   70 Y=(X-R)**0.5
   90 WRITE(6,300) X,Y
  300 FORMAT(' ',F9.4,E20.7)
      N=N-1
      GO TO 80
   20 STOP
   30 Y=0
      GO TO 90
   50 Y=-(R**2-X**2)**0.5
      GO TO 90
      END
```



Sl. 4.7.5

Razgranata algoritamska struktura na sl. 4.7.5 nalazi se u okviru ciklusa, koji je označen na slici sa C. Izlazni kriterijum ovog ciklusa sadrži ispitivanje promenljive n, kojom se kontroliše broj izračunavanja funkcije y(x). Kada se izračuna vrednost funkcije i za zadnju zadatu vrednost argumenta, izlazi se iz ciklusa i prekida se dalji rad po programu.

Za n=4 i r=12, izlazni rezultati po ovom programu se štampaju u obliku tabele:

```
  8.5200        0.0
-15.0000        0.0
 -2.0000       -0.1183216E 02.
 37.0480        0.5004795E 01
```

gde je u prvoj koloni štampana vrednost argumenta, a u drugoj odgovarajuća vrednost funkcije.

4. 8. Dalje mogućnosti naredbe FORMAT

Opis ulaznog, odnosno izlaznog sloga u FORMAT-naredbi može u nekim primerima biti veoma glomazan. U ovom odeljku biće razmotrene sve mogućnosti naredbe FORMAT koje dozvoljavaju kraći zapis opisa ulaznih, odnosno izlaznih polja jednog sloga.

4. 8. 1. Ponavljanje jednog opisa

Vrlo često više uzastopnih polja imaju isti opis. Da bi se izbeglo uzastopno ponavljanje istog opisa u FORMAT-naredbi, mogu se opisi pisati u obliku

$$\begin{matrix} nIk \\ nFk.d \\ nEk.d \end{matrix} \qquad (4.8.1)$$

gde je n ceo neoznačen broj koji pokazuje koliko puta se ponavlja opis koji sledi. Tako, umesto niza opisa:

FORMAT(F7. 2, F7. 2, I4, I4, I4)

može se pisati:

FORMAT(2F7. 2, 3I4)

4. 8. 2. Ponavljanje više opisa

Ako se više opisa ponavlja, tada se ovi opisi mogu pisati izmedju otvorene i zatvorene male zagrade, tj.

$$n(\text{lista}) \qquad (4.8.2)$$

gde je

lista - spisak opisa medju sobom razdvojenih zarezima,

n - ceo neoznačen broj koji ukazuje na broj ponavljanja opisa navedenih u listi. Tako, niz opisa

j FORMAT(I2, F6. 4, I2, F6. 4, I2, F6. 4)

može se kraće pisati u obliku

j FORMAT(3 (I2, F6. 4))

4. 8. 3. Prelazak na novi slog

Ako promenljive u listi naredbe READ dobijaju brojne vrednosti iz više ulaznih slogova (sa više kartica), tada je potrebno u FORMAT-naredbi označiti kraj jednog ulaznog sloga i početak novog ulaznog sloga. Slična situacija nastaje kada se vrednosti promenljivih u listi naredbe WRITE žele izdati u više izlaznih slogova (novih redova). Ova informacija, o kraju jednog ulaznog, odnosno izlaznog, sloga i o početku sledećeg zadaje se simbolom kosa crta (/) u naredbi FORMAT. Tako naredba FORMAT može imati sledeći izgled:

j FORMAT($slog_1$ / $slog_2$) (4. 8. 3)

U opisnoj naredbi (4. 8. 3) kosa crta označava kraj ulaznog $sloga_1$, kada je ova naredba pridružena izvršnoj ulaznoj naredbi, ili kraj izlaznog $sloga_1$, kada je ova naredba pridružena izvršnoj izlaznoj naredbi.

Više kosih crta navedenih jedna za drugom imaju sledeće značenje:

a) Ako je n kosih crta navedeno na početku, ili na kraju naredbe FORMAT, tada će n ulaznih slogova biti preskočeno, odnosno izdato n praznih slogova na izlazu.

b) Ako se n uzastopnih kosih crta nalazi izmedju dva sloga, tada će na ulazu biti preskočen n-1 ulazni slog, odnosno na izlazu će biti izdat n-1 prazan slog.

Tako, naredba

FORMAT($slog_1$ /// $slog_2$) (4. 8. 4)

ima sledeće značenje:

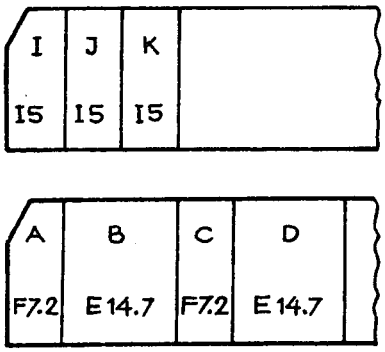
a) Na ulazu: sa prve kartice čita se $slog_1$, i prva kosa crta označava kraj ovog sloga. Druga i treća kosa crta označavaju prolazak druge i treće kartice bez čitanja, a zatim se čita $slog_2$ sa četvrte kartice.

b) Na izlazu: u prvom redu štampa se $slog_1$, a zatim prva kosa crta označava kraj ovog izlaznog sloga. Druga i treća kosa crta proizvode po

jedan novi red, a zatim se štampa $slog_2$ u četvrtom redu. Ovo objašnjenje je dato ne vodeći pri tome računa o prvom, komandnom, simbolu u izlaznom slogu. Medjutim, $slog_1$ i $slog_2$, moraju imati, kao prvi simbol, komandni simbol za vertikalno pomeranje papira na štampaču. Dejstvo ovog simbola je nezavisno od opisanog dejstva kosih crta u naredbi FORMAT.

Primer

Sastaviti program koji celobrojnim promenljivim I, J i K dodeljuje brojne vrednosti sa jedne kartice, a realnim promenljivim A, B, C i D dodeljuje brojne vrednosti sa druge kakartice. Dodeljene brojne vrednosti promenljivim štampati na paralelnom štampaču, tako da su bojne vrednosti promenljivih I, J i K u jednom, a promenljivih A, B, C i D u drugom redu štampanog dokumenta. Opisi pojedi - nih polja, i njihov raspored na karticama prikazani su na sl. 4. 8. 1.



Sl. 4. 8. 1

Izgled FORTRAN-programa prikazan je niže:

```
      READ(5,10)I,J,K,A,B,C,D
   10 FORMAT(3I5/2(F7.2,E14.7))
      WRITE(6,20)I,J,K,A,B,C,D
   20 FORMAT(' ',3I10/' ',2(F7.2,2X,E14.7,2X))
      STOP
      END
```

Tako za ulazne podatke

I=3867; J=-4005; K=11007

A=372,45; B=$-42,58\cdot 10^{17}$; C=-0,56; D=$4,13706\cdot 10^{-12}$

štampani dokument ima oblik

```
      3867      -4005      11007
 372.45  -0.4258000E 19    -0.56    0.4137060E-11
```

4.8.4. Veza izmedju opisa i liste

Posmatrajmo FORMAT-naredbu koja sleva na desno sadrži n opisa brojnih podataka. Ulazno-izlazna naredba, kojoj je pridružena ovakva FORMAT-naredba, neka sadrži u listi m imena promenljivih. Ovde su moguća tri slučaja:

1) Broj opisa u FORMAT-naredbi jednak je broju promenljivih u listi, tj. m = n. U ovom slučaju svakoj promenljivoj u listi odgovara jedan opis brojnog podatka u FORMAT-naredbi. Dodeljivanjem brojnih vrednosti svim promenljivim u listi, po opisima u FORMAT-naredbi, sleva nadesno, iskorišćeni su svi navedeni opisi u FORMAT-naredbi.

Tako, naredbe

```
      READ(5, 10) A1, M4, K2, JOT
10    FORMAT(F7.3, 3I4)
```

ispunjavaju uslov da je broj promenljivih (4) u listi, jedank broju opisa (4) u FORMAT-naredbi.

2) Broj opisa u FORMAT-naredbi je veći od broja promenljivih u listi, tj. n > m. U ovom slučaju svakoj promenljivoj u listi biće pridružen jedan opis u FORMAT-naredbi redom sleva nadesno, a ostatak opisa (n-m), u FORMAT-naredbi neće biti iskorišćen.

Ako se prethodno navedenoj READ-naredbi pridruži druga FORMAT--naredba, tako da naredbe imaju izgled:

```
      READ(5, 10)A1, M4, K2, JOT
10    FORMAT(F7, 3, 3I4, E12.5, I2)
```

tada će promenljivoj A1 biti pridružen opis F7.3, promenljivim M4, K2 i JOT opis I4, a opisi E12.5 i I2 neće se koristiti.

Prema tome, ako u FORMAT-naredbi postoji veći broj opisa nego što to zahteva lista naredbe READ, odnosno WRITE, tada će biti iskorišćeno samo onoliko opisa, sleva nadesno, koliko ima promenljivih u listi.

3) Broj opisa u FORMAT-naredbi je manji od broja promenljivih u listi, tj. n < m. Tada treba razlikovati dva slučaja:

a) Ako izmedju spoljne otvorene i zatvorene zagrade FORMAT-naredbe, koje se zovu zagrade nultog nivoa, ne postoje unutrašnje zagrade, tada će posle prolaska kroz sve opise, sleva nadesno, doći do prelaska na novi slog i ponovo će se koristiti isti opisi, od početka FORMAT-naredbe. Tako ako naredba sadrži 3 opisa, prvi slog se obrazuje od navedena 3 opisa, a sledeći slogovi se ponavljaju prema potrebi liste, ulazne odnosno izlazne, naredbe, tj.

$$\text{FORMAT } (\text{opis}_1, \text{opis}_2, \text{opis}_3) \qquad (4.8.5)$$

prvi slog
i sledeći slogovi

Primer

Neka su brojne vrednosti promenljivih A, B i C zadate na tri kartice sa opisom F7.2. Tada se dodeljivanje ovih brojnih vrednosti promenljivim može izvršiti pomoću naredbi:

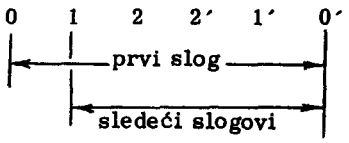
```
      READ(5, 20)A, B, C
20    FORMAT(F7.2)
```

Naredba 20 definiše ulazni slog od jednog polja sa opisom F7.2, i u početku dodeljuje brojnu vrednost promenljivoj A. Iza ovoga, pošto su iscrpeni svi opisi u FORMAT-naredbi, dolazi do formiranja novog ulaznog sloga, a to znači prelazak na čitanje sledeće kartice pod opisom F7.2, a brojna vrednost se dodeljuje promenljivoj B. Na isti način se prelazi na sledeću karticu i dodeljuje se brojna vrednost promenljivoj C.

b) Ako izmedju spoljne otvorene i zatvorene zagrade, FORMAT-naredbe, postoje unutrašnje zagrade koje mogu biti prvog i drugog nivoa, tada će posle prolaska kroz sve opise, sleva nadesno, doći do prelaska na sledeći slog i do ponavljanja opisa sleva nadesno, koji se nalaze izmedju zatvorene zagrade nultog nivoa i najbliže otvorene zagrade prvog nivoa.

Označimo li dekadnim ciframa 0, 1 i 2 otvorene, a sa 0′, 1′, i 2′ zatvorene zagrade, nultog, prvog i drugog nivoa, tada će biti

FORMAT (....(....(....)....)....) (4.8.6)



Primer

Na prvoj kartici ulaznih podataka nalaze se brojne vrednosti promenljivih N i E sa opisima I5 i E12.5. Iza ove kartice slede tri kartice na kojima se nalaze brojevi od prve do devete kolone, sa opisom F9.4 koje treba dodeliti promenljivim X1, X2 i X3. Naredbe koje obezbedjuju unošenje ovakvih podataka mogu se zapisati u ubliku

```
      READ(5,40)N,E,X1,X2,X3
40    FORMAT(I5,E12.5/(F9.4))
```

Navedena FORMAT-naredba definiše prvi ulazni slog sa opisima I5 i E12.5, i sledeće ulazne slogove sa opisom F9.4. Kraj prvog sloga je definisan kosom crtom, a ostali ulazni slogovi su definisani izmedju otvorene zagrade prvog nivoa i zatvorene zagrade nultog nivoa. Broj ovih slogova zavisi od broja promenljivih u listi READ naredbe. U navedenom primeru biće formirana tri ovakva sloga.

4.8.5. Tekst u FORMAT-naredbi

U dosadašnjem izlaganju videli smo kako se unose i izdaju brojni podaci. Medjutim, veliki broj brojnih podataka na izlazu čini nepreglednim štampani dokument. Zato je pogodno imati mogućnost tekstuelnog objašnjenja štampanih rezultata. Da bi se ovo omogućilo, u FORMAT-naredbi se može navoditi tekst. Tekst je sastavljen od niza simbola: od slova, cifara ili specijalnih znakova. Ovakav niz simbola zove se tekstuelna konstanta ili literal. Broj simbola literala zove se dužina literala.

Literal se može definisati na dva načina, kao

- niz simbola zapisanih izmedju apostrofa, ili
- pomoću posebnog H opisa.

Literal definisan izmedju apostrofa predstavlja niz simbola od kojih je prvi i zadnji simbol apostrof, tj.

literal' (4.8.7)

gde je

- simbol FORTRAN-jezika,

literal - tekst sastavljen od slova, cifara ili specijalnih znakova.

Tako može biti zapisan literal

VREDNOST FUNKCIJE'

U okviru niza simbola, koji čine literal u (4.8.7), ne može se nalaziti jedan specijalni znak apostrof. Medjutim, ako se želi navesti apostrof, u okviru literala, moraju se pisati dva apostrofa ('').

Drugi oblik navodjenja teksta u FORMAT-naredbi vrši se pomoću opisa

nHliteral (4.8.8)

gde je

n - ceo neoznačen broj, koji ukazuje na broj simbola literala,

H - simbol FORTRAN-jezika, koji ukazuje da se radi o opisu teksta, *)

literal - tekst sastavljen od slova, cifara i specijalnih znakova.

Tako se može pisati

17HVREDNOST FUNKCIJE

Kao što se vidi, navodjenje teksta izmedju apostrofa ne zahteva pisanje broja simbola koji čine tekst, kao što je to slučaj sa opisom (4.8.8). Medjutim, sa gledišta korišćenja oba opisa imaju isto značenje:

a) Ako je FORMAT-naredba, u kojoj se nalazi literal, pridružena naredbi izlaza (WRITE), tada će navedeni niz simbola činiti polje u izlaznom slogu na onom mestu na kojem se nalazi literal, sleva nadesno, u FORMAT-naredbi.

b) Ako je FORMAT-naredba, u kojoj se nalazi literal, pridružena naredbi ulaza (READ), tada će navedeni niz simbola, koji čini literal u FOR-

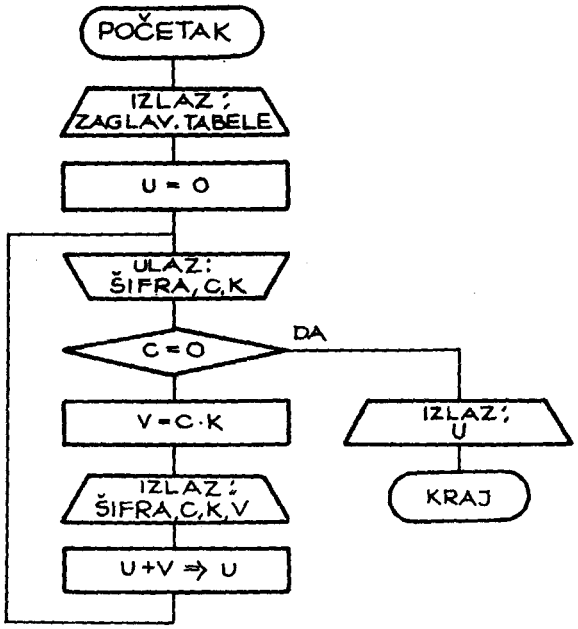
*) Herman Hollerit je pronalazač uredjaja za bušenje kartica, i po njegovom imenu je uzeto slovo H za opis teksta.

MAT-naredbi, biti zamenjen sadržajem odgovarajućeg polja u ulaznom slogu.

Ako FORMAT-naredba sadrži samo literale i opise praznih polja, bez opisa brojnih podataka, tada naredba ulaza, odnosno izlaza, kojoj se pridružuje ovakva FORMAT-naredba, ne sadrži listu.

Primer

Sastaviti program koji izračunava vrednost na osnovu zadate cene i količine. Na svakoj kartici nalaze se tri polja: prvo polje sadrži 5 kolona, i od 2. do 5. kolone buši se šifra robe, a prva kolona ostaje nebušena. Drugo polje sadrži cenu robe, i opisuje se sa opisom F10.2. Treće polje sadrži količinu robe i opisuje se sa F10.2. Broj ovakvih kartica može biti proizvoljan, a zadnja kartica u paketu, u polju za cenu, ima bušenu vrednost nula (to može biti i prazna kartica). Za svaku karticu izračunati vrednost, kao proizvod cene i količine, a na kraju obrade svih kartica štampati sumu svih pojedinačnih vrednosti.



Sl. 4.8.2

Na sl. 4. 8. 2 data je šema algoritma, gde su uvedene sledeće oznake:

C - cena,

K - količina,

V - vrednost,

U - ukupna vrednost.

FORTRAN-program ima sledeći izgled:

```
      WRITE(6,10)
   10 FORMAT('1SIFRA',4X,'CENA',3X,'KOLICINA',4X,'VREDNOST'/)
      UKUPNO=0
   20 READ(5,30) CENA,AKOL
   30 FORMAT(1X,'SIFR',2F10.2,F12.2)
      IF(CENA) 40,50,40
   40 VRED=CENA*AKOL
      WRITE(6,30) CENA,AKOL,VRED
      UKUPNO=UKUPNO+VRED
      GO TO 20
   50 WRITE(6,60) UKUPNO
   60 FORMAT(19X,18(1H-)/19X,'UKUPNO',F12.2/)
      STOP 1
      END
```

Za ulazne podatke

```
AC18      12.00      36.00
DE34     135.00      12.50
FA12      15.50       4.70
AK50       1.75     300.00
B821     200.00      45.50
            0.00
```

gde jedan štampani red odgovara sadržaju jedne kartice. Izlazni rezultati će biti u obliku

```
SIFRA      CENA     KOLICINA      VREDNOST

AC18      12.00        36.00        432.00
DE34     135.00        12.50       1687.50
FA12      15.50         4.70         72.85
AK50       1.75       300.00        525.00
B821     200.00        45.50       9100.00
                    ----------------------
                    UKUPNO       11817.35
```

4.9. Elementarne funkcije

U mnogim matematičkim i tehničkim zadacima zahteva se izračunavanje elementarnih funkcija za zadate vrednosti argumenata. Ovaj problem

je rešavan u matematici nekoliko stotina godina unazad, i sastojao se u iznalaženju formula po kojima se mogu izračunavati tablice za elementarne funkcije. To se najčešće vršilo razvijanjem funkcije u stepene redove ili primenom iteracionih formula. Poslednjih godina za izračunavanje elementranih funkcija široko se koriste ortogonalni polinomi, a posebno polinomi Čebiševa.

Razvoj elektronskih računskih mašina posebno je doveo do razvoja raznih algoritama, kojima se može izračunati vrednost elementarne funkcije sa zadatom tačnošću.

Neka je f(x) elemenatarna funkcija, čiju vrednost treba odrediti za zadatu vrednost argumenta. Neka je g(x) aritmetički izraz po kojem se vrši približno izračunavanje funkcije f(x), tako da je

$$f(x) \approx g(x) \qquad (4.9.1)$$

Na računaru se vrši izračunavanje funkcije f(x) primenom aproksimacione formule g(x). Apsolutna greška pri ovom izračunavanju je

$$E = |f(x) - g(x)| \qquad (4.9.2)$$

a relativna greška je odredjena sa

$$\varepsilon = \left| \frac{f(x) - g(x)}{f(x)} \right| \qquad (4.9.3)$$

Algoritmi za izračunavanje elementarnih funkcija nalaze se u memoriji računara, i po pozivu se koriste za izračunavanje. Poziv algoritma za izračunavanje elementarne funkcije u FORTRAN-jeziku vrši se sa

$$\text{funkcija}(\psi) \qquad (4.9.4)$$

gde je

funkcija - propisano ime elementarne funkcije,

ψ - aritmetički izraz, čija se vrednost uzima kao argument funkcije.

Funkcija zapisana u obliku (4.9.4) može se nalaziti kao argument u aritmetičkom izrazu. Izračunavanje funkcije predstavlja operaciju najvišeg prioriteta u aritmetičkom izrazu. Propisana imena funkcija predstavljaju uobičajene matematičke oznake ovih funkcija. Od toga se odstupa samo u slučajevima kad to ne odgovara uvedenim konvencijama FORTRAN-jezika.

Vremena izračunavanja elementarnih funkcija, navedena u daljem tekstu, odnose se na računar IBM-360/44 bez specijalnih registara, kojima se može povećati brzina rada računara.

4.9.1. Eksponencijalna funkcija

Eksponencijalna funkcija (e^{ψ}) piše se u obliku

$$EXP(\psi) \qquad (4.9.5)$$

gde izračunata vrednost argumenta ψ, mora biri realan broj, a vrednost funkcije biće takodje realan broj. Vrednost argumenta mora zadovoljavati uslov

$$\psi \leq 174,673 \qquad (4.9.6)$$

Vrednost argumenta koja ne zadovoljava uslov (4.9.6) dovodi do brojne vrednosti funkcije koja ne može biti registrovana u registru memorije (prelazi opseg realnih brojeva). Ako je

$$\psi < -180,218 \qquad (4.9.7)$$

onda se za vrednost funkcije uzima nula. Ako je

$$|\psi| < 0,373 \cdot 10^{-8} \qquad (4.9.8)$$

vrednost funkcije je 1. Relativna greška ovako izračunate funkcije iznosi

$$\varepsilon < 0,187 \cdot 10^{-8} \qquad (4.9.9)$$

Srednje vreme izračunavanja eksponecijalne funkcije iznosi oko 310 μsek.

4.9.2. Logaritamska funkcija

Logaritamska funkcija se piše u obliku

$$ALOG(\psi) \qquad (4.9.10)$$

ako se radi o prirodnom logaritmu ($\ln\psi$) ili

$$ALOG10(\psi) \qquad (4.9.11)$$

ako se radi o dekadnom logaritmu ($\log\psi$). Izračunate vrednosti argumenta i funkcije su realni brojevi. Kako naziv logaritamske funkcije, u uobičajenoj matematičkoj notaciji, počinje slovom L, a ovo slovo po unutrašnjoj

konvenciji FORTRAN-jezika ukazuje na celobrojne vrednosti, to je ispred uobičajene oznake funkcije, u FORTRAN-jeziku dodato slovo A.

Argument u logaritamskim funkcijama (4.9.10) i (4.9.11)mora ispunjavati uslov:

$$\psi > 0 \qquad (4.9.12)$$

Relativna greška pri izračunavanju funkcije je

$$\varepsilon < 0,372 \cdot 10^{-8} \qquad (4.9.13)$$

Srednje vreme izračunavanja logaritamske funkcije iznosi 230 μsek.

4.9.3. Kvadratni koren

Izračunavanje kvadratnog korena ($\sqrt{\psi}$) za zadatu vrednost argumenta, piše se sa

$$\text{SQRT}(\psi) \qquad (4.9.14)$$

gde su izračunate vrednosti argumenta i funkcije realni brojevi. Vrednost argumenta mora ispunjavati uslov

$$\psi \geq 0 \qquad (4.9.15)$$

Relativna greška pri izračunavanju kvadratnog korena iznosi

$$\varepsilon < 0,298 \cdot 10^{-7} \qquad (4.9.16)$$

Srednje vreme izračunavanja kvadratnog korena iznosi oko 140 μsek.

Ranije smo videli da se kvadratni koren mogao izračunati preko operacije stepenovanja, tj.

$$(\psi) ** 0.5 \qquad (4.9.17)$$

Međutim, ovde treba imati u vidu da se izračunavanje stepena (4.9.17) na računaru vrši korišćenjem logaritmovanja i antilogaritmovanja. Prema tome, oblik (4.9.17) je isto što i

$$\text{EXP}(0.5 * \text{ALOG}(\psi)) \qquad (4.9.18)$$

Kako se izrazom (4.9.18) poziva logaritamska funkcija, čije vreme izvršavanja iznosi 230 μsek, i eksponencijalna funkcija, čije vreme izvršavanja iznosi 310 μsek, to će izračunavanje kvadratnog korena trajati

oko 540 μsek. Odavde se vidi da je izračunavanje kvadratnog korena preko funkcije (4.9.14) 4 puta brže nego preko operacije stepenovanja (4.9.17).

4.9.4. Apsolutna vrednost

Apsolutna vrednost aritmetičkog izraza ($|\psi|$) piše se sa

$$ABS(\psi) \tag{4.9.19}$$

gde su izračunate vrednosti argumenta i funkcije realni brojevi.

Ako je izračunata vrednost argumenta ceo broj, apsolutna vrednost se piše u obliku

$$IABS(\psi) \tag{4.9.20}$$

pri čemu je i vrednost funkcije ceo broj. Za ovu funkciju se ne postavljaju ograničenja na vrednosti argumenta.

Srednje vreme dobijanja apsolutne vrednosti broja iznosi oko 4 μsek.

4.9.5. Trigonometrijske funkcije

Kod svih trigonometrijskih funkcija izračunate vrednosti argumenta i funkcije su realni brojevi. Argument trigonometrijske funkcije mora biti zadat u radijanima.

a) Trigonometrijska funkcija $\sin\psi$ piše se u obliku

$$SIN(\psi) \tag{4.9.21}$$

gde mora biti ispunjen uslov

$$|\psi| < 8,235 \cdot 10^{6} \tag{4.9.22}$$

Relativna greška, pri izračunavanju funkcije, je

$$\varepsilon < 0,372 \cdot 10^{-8} \tag{4.9.23}$$

a srednje vreme izvršavanja oko 200 μsek.

b) Trigonometrijska funkcija $\cos\psi$ se piše u obliku

$$COS(\psi) \tag{4.9.24}$$

gde mora biti ispunjen uslov (4.9.22). Relativna greška pri izračunavanju kosinusne funkcije je

$$\varepsilon < 0,298 \cdot 10^{-7} \tag{4.9.25}$$

a srednje vreme izvršavanja oko 200 μsek.

c) Trigonometrijska funkcija tgψ piše se u obliku

$$\mathrm{TAN}(\psi) \tag{4.9.26}$$

gde mora biti ispunjen uslov (4.9.22), pri čemu vrednost argumenta ne sme biti

$$\psi \approx \left(k + \frac{1}{2}\right)\pi \tag{4.9.27}$$

gde je k ceo broj.

Relativna greška pri izračunavanju tangesa je

$$\varepsilon \leq 1,74 \cdot 10^{-8} \tag{4.9.28}$$

a srednje vreme izvršavanja oko 220 μsek.

d) Trigonometrijska funkcija ctgψ piše se u obliku

$$\mathrm{COTAN}(\psi) \tag{4.9.29}$$

gde mora biti ispunjen uslov (4.9.22), pri čemu vrednost argumenta ne sme biti

$$\psi \approx k\pi \tag{4.9.30}$$

gde je k ceo broj.

Relativna greška pri izračunavanju kotangensa je

$$\varepsilon \leq 1,74 \cdot 10^{-8} \tag{4.9.31}$$

a srednje vreme izvršavanja oko 227 μsek.

4.9.6. Inverzne trigonometrijske funkcije

Kod svih inverznih trigonometrijskih funkcija izračunate vrednosti argumenta i funkcije su realni brojevi.

a) Inverzna sinusna funkcija (arcsin ψ) se piše u obliku

$$\mathrm{ARSIN}(\psi) \tag{4.9.32}$$

gde mora biti ispunjen uslov

$$|\psi| \leq 1 \tag{4.9.33}$$

Relativna greška pri izračunavanju funkcije je

$$\varepsilon \leq 0,372 \cdot 10^{-8} \tag{4.9.34}$$

a srednje vreme izvršavanja oko 310 μsek.

b) Inverzna kosinusna funkcija (arccos ψ) piše se u obliku

$$\text{ARCOS}(\psi) \qquad (4.9.35)$$

gde mora biti ispunjen uslov (4.9.33). Relativna greška je ista kao kod inverzne sinusne funkcije (4.9.34), a srednje vreme izvršavanja iznosi oko 325 μsek.

c) Inverzna funkcija tangensa (arctg ψ) piše se u obliku

$$\text{ATAN}(\psi) \qquad (4.9.36)$$

gde izračunata vrednost argumenta može biti ma koji realan broj. Relativna greška pri izračunavanju funkcije je

$$\varepsilon < 0,745 \cdot 10^{-8} \qquad (4.9.37)$$

a srednje vreme izvršavanja oko 165 μsek.

4.9.7. Hiperbolične funkcije

Kod svih hiperboličnih funkcija izračunate vrednosti argumenta i funkcije su realni brojevi.

a) Hiperbolična sinusna funkcija (sinh ψ) piše se u obliku

$$\text{SINH}(\psi) \qquad (4.9.38)$$

gde mora biti ispunjen uslov

$$|\psi| < 174,673 \qquad (4.9.39)$$

Relativna greška pri izračunavanju funkcije je

$$\varepsilon < 0,149 \cdot 10^{-7} \qquad (4.9.40)$$

a srednje vreme izvršavanja oko 460 μsek.

b) Hiperbolična kosinusna funkcija (cosh ψ) piše se u obliku

$$\text{COSH}(\psi) \qquad (4.9.41)$$

gde mora biti ispunjen uslov (4.9.39), a relativna greška je ista kao kod hiperbolične sinusne funkcije (4.9.40). Srednje vreme izvršavanja iznosi oko 485 μsek.

c) Hiperbolična tangensna funkcija (tgh ψ) piše se u obliku

$$\text{TANH}(\psi) \qquad (4.9.42)$$

gde izračunata vrednost argumenta može biti ma koji realan broj. Relativna greška pri izračunavanju funkcije je

$$\varepsilon < 0,745 \cdot 10^{-8} \qquad (4.9.43)$$

a srednje vreme izvršavanja zavisi od vrednosti argumenta i kreće se od 90 do 450 μsek.

Primer

Na sedam kartica je zadato 7 vrednosti za argument x, sa opisom F8.4. Izračunati vrednosti funkcija

$$y_1 = 1 - e^{-x}\sin 2x + \log(\cos^2 x)\cdot \text{tg}x$$

$$y_2 = \arcsin\left(\frac{x}{100}\right) + \ln|x| \cdot \text{arctg}x$$

$$y_3 = \sqrt{|1 - \text{tgh}x|} + \sinh x - 2\cosh x$$

i štampati u obliku tabele.

FORTRAN-program u ovom slučaju ima izgled:

```
      WRITE(6,100)
  100 FORMAT('1',4X,'X',14X,'Y1',15X,'Y2',15X,'Y3'/)
      I = 0
   10 READ(5,200) X
  200 FORMAT(F8.4)
      Y1 = 1.-EXP(-X)*SIN(2.*X)+ALOG10(COS(X)**2)*TAN(X)
      Y2 = ARSIN(X/100.)+ALOG(ABS(X))*ATAN(X)
      Y3 = SQRT(ABS(1.-TANH(X)))+SINH(X)-2.*COSH(X)
      WRITE(6,300) X, Y1, Y2, Y3
  300 FORMAT(' ',F8.4,2X,3E17.7)
      I = I+1
      IF(I-7) 10,20,10
   20 STOP
      END
```

Izlazni rezultati su štampani u obliku tabele

X	Y1	Y2	Y3
-75.3427	-0.5828479E 32	-0.7584949E 01	-0.7888761E 33
-34.2885	-0.3989898E 15	-0.5799388E 01	-0.1167875E 16
-28.0032	-0.7487951E 12	-0.5399271E 01	-0.2176315E 13
-2.3410	-0.9710955E 01	-0.1016113E 01	-0.1422783E 02
0.5684	0.3911758E 00	-0.2863056E 00	-0.1035359E 01
5.2028	0.2229445E 01	0.2329437E 01	-0.9089070E 02
17.3333	0.4728911E 02	0.4490717E 01	-0.1685488E 08

4.10. Naredbe promenljivog bezuslovnog prelaska

Uslovna naredba prelaska daje mogućnost grananja programa u tri različite grane, pri čemu je svaka od njih uslovljena vrednošću aritmetičkog izraza. Vrednost aritmetičkog izraza se ispituje da li je manja, jednaka ili veća od nule. Pored ovakve naredbe uslovnog prelaska, postoji i naredba po kojoj se prelazak vrši po brojnoj vrednosti promenljive. Ovu naredbu ćemo zvati naredba promenljivog bezuslovnog prelaska. Opšti oblik ove naredbe je

GO TO (lista),i (4.10.1)

gde je

GO TO - službena reč, i označava naredbu prelaska,

lista - spisak obeležja izvršnih naredbi u programu medju sobom razdvojenih zarezima,

i - ime celobojne promenljive.

Neka lista u (4.10.1) sadrži m obeležja, sleva nadesno označenih sa $n_1, n_2, \ldots, n_m$, dejstvo naredbe (4.10.1) je dledeće:

a) Ako je brojna vrednost promenljive $i = k$, a $k \in [1, m]$, onda će se izvršiti prelazak na naredbu sa obeležjem n_k.

b) Ako je brojna vrednost promenljive $i = p$, a $p \notin [1, m]$, tada će se preći na naredbu koja sledi neposredno iza naredbe (4.10.1) u programu.

Na sl. 4.10.1 prikazana su grananja koja se vrše naredbom (4.10.1). Kako je ovakav grafički prikaz nepotrebno složen uveden je odgovarajući prostiji grafički prikaz (sl. 4.10.2).

U FORTRAN-jeziku postoji i alternativni oblik naredbe promenljivog bezuslovnog prelaska, koji se piše

GO TO i, (lista) (4.10.2)

gde je

GO TO - službena reč, i označava naredbu prelaska,

i - ime celobrojne promenljive, a

lista - spisak obeležja izvršnih naredbi u programu medju sobom razdvojenih zarezima.

Ako lista u (4.10.2) sadrži m obeležja, sleva nadesno označenih sa $n_1, n_2, \ldots, n_m$ tada promenljiva i mora imati jednu od vrednosti zapisanih kao obeležja $n_1, n_2, \ldots, n_m$, u trenutku izvršavanja naredbe (4.10.2). Izvršavanje ove naredbe prouzrokuje prelazak na naredbu sa obeležjem dodeljenim promenljivoj i. Promenljivoj i može se dodeliti obeležje u ma kojem delu programa naredbom

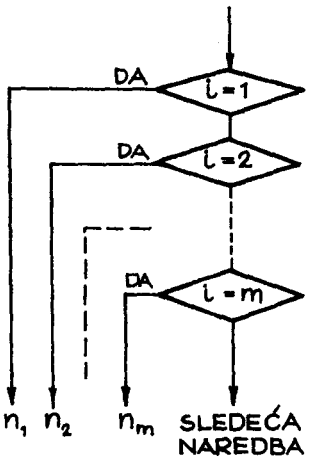
ASSIGN n TO i (4.10.3)

gde je

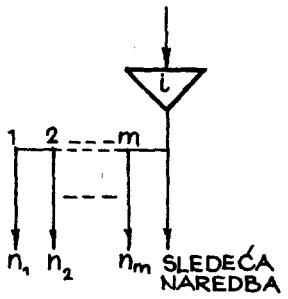
ASSIGN - službena reč, i označava dodeljivanje vrednosti obeležja promenljivoj,

n - obeležje izvršne FORTRAN-naredbe, a

i - ime celobrojne promenljive.



Sl. 4.10.1



Sl. 4.10.2

Dejstvo naredbe (4.10.3) je sledeće: obeležje $n \in \{n_1, n_2, \ldots, n_m\}$ dodeljuje se celobrojnoj promenljivoj i. Ovde je važno imati u vidu razliku izmedju obeležja i celog broja. Vrednost obeležja ne može biti dodeljena promenljivoj i ako se koristi aritmetička naredba

i = n (4.10.4)

Razlog za ovo je interno predstavljanje informacija u računaru. Obeležje kao informacija u računaru nije ceo broj nad kojim se mogu izvoditi arit-

metičke operacije, već adresa koja jednoznačno ukazuje na jednu naredbu. Zato je u FORTRAN-jeziku uvedena posebna naredba (4.10.3), kojom se promenljivoj može dodeliti vrednost obeležja. Medjutim, u naredbi (4.10.1) promenljivoj i može se dodeliti brojna vrednost sa ulaza ili aritmetičkom naredbom. Prema tome, naredba (4.10.1) jeste pogodnija za korišćenje u programima, nego naredba (4.10.2).

Primer

Na n kartica zadate su brojne vrednosti promenljivih k, x_1, x_2 i x_3, gde je $n \leq 99$. Brojna vrednost promenljive k bušena je u prvoj koloni kartice, a zatim u sledeća tri polja kartice brojne vrednosti promenljivih x_1, x_2 i x_3 sa opisom F7.4. Izračunati vrednost y_k na sledeći način

$$y_k = \begin{cases} 2x_1^2 + 3x_2^2 + 4x_3^2 & \text{ako je } k = 1 \quad (1) \\ (x_1 - x_2)^2 + (x_3 - x_1)^2 & \text{ako je } k = 2 \quad (2) \\ 4(x_1 - x_2)(x_2 - x_3) + x_1 x_2 x_3 & \text{ako je } k = 3 \quad (3) \\ (x_1 + x_2 + x_3)^3 - 8x_1 x_2 x_3^2 & \text{ako je } k = 4 \quad (4) \end{cases}$$

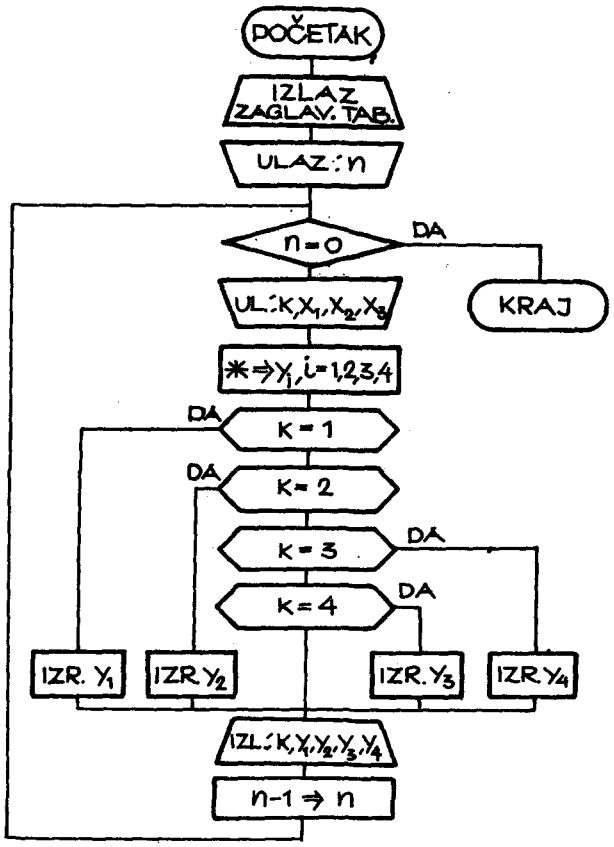
Ako $k \notin \{1, 2, 3, 4\}$ tada se ne vrši nikakvo izračunavanje. Rezultate štampati u obliku tabele u kojoj će se u prvoj koloni nalaziti broj k, a vrednost y_k štampati u (k+1)-oj koloni. U ostalim kolonama štampati zvezdice (*).

Ovaj zadatak se može rešiti na dva načina: bez korišćenja i sa korišćenjem naredbe promenljivog bezuslovnog prelaska.

a) Prvo rešenje: bez korišćenja naredbe promenljivog bezuslovnog prelaska.

Šema algoritma, u ovom slučaju, prikazana je na sl. 4.10.3.

U algoritmu se dodeljuje, pre izračunavanja funkcije, simbol (*) kao vrednost odgovarajuće promenljive. Zatim će ovaj simbol biti zamenjen vrednošću funkcije, u odgovarajućoj promenljivoj y_1, y_2, y_3 ili y_4, a u zavisnosti od toga po kojoj formuli će biti vršeno izračunavanje. Na ovaj način u algoritamskom koraku izlaza biće izdata brojna vrednost promenljive k i jedne funkcije Y_1, Y_2, Y_3 ili Y_4, a ostale funkcije će biti izdate kao simbol zvezdice.



Sl. 4. 10. 3

Ovakav oblik izalaza u FORTRAN-jeziku ostvaren je dodeljivanjem promeljivim Y1, Y2, Y3 i Y4 brojne vrednosti veće nego što je to predvidjeno opisom odgovarajućih promenljivih u FORMAT-naredbi koja je pridružena izlaznoj naredbi. Kako je promenljivim Y1, Y2, Y3 i Y4 pridružen opis F9. 3 u FORMAT- naredbi sa obeležjem 400, to znači da rezultat može imati najviše 5 celih mesta. Ako se promenljivim dodeli brojna vrednost koja ima veći broj celih mesta, na predvidjenom polju za štampanje vrednosti promeljive biće štampane zvezdice. Zato je promenljivim Y1,Y2, Y3 i Y4 dodeljena brojna vrednost $C = 10^8$, koja će pri izlazu dati simbole zvezdice za one promenljive kojima se za zadate vrednosti ulaza ne dodeljuje programom izračunata vrednost po jednoj od formula (1), (2), (3) ili

(4). FORTRAN-program sastavljen po algoritmu na sl. 4. 10. 3. ima sledeći izgled:

```
      WRITE(6,100)
  100 FORMAT(3H1 K,9X,'Y1',10X,'Y2',10X,'Y3',10X,'Y4'/)
      READ(5,200) N
  200 FORMAT(I2)
   10 IF(N) 20,20,30
   20 STOP
   30 READ(5,300) K, X1, X2, X3
  300 FORMAT(I2,3X,3(3X,F7.4))
      Y1 = .1E9
      Y2 = Y1
      Y3 = Y1
      Y4 = Y1
      IF(K-1) 40,11,40
   40 IF(K-2) 50,22,50
   50 IF(K-3) 60,33,60
   60 IF(K-4) 70,44,70
   44 Y4 = (X1+X2+X3)**3-8.*X1*X2*X3**2
      GO TO 70
   33 Y3 = 4.*(X1-X2)*(X2-X3)+X1*X2*X3
      GO TO 70
   22 Y2 = (X1-X2)**2+(X3-X1)**2
      GO TO 70
   11 Y1 = 2.*X1**2+3.*X2**2+4.*X3**2
   70 WRITE(6,400) K, Y1, Y2, Y3, Y4
  400 FORMAT(' ',I2,2X,4(3X,F9.3))
      N = N-1
      GO TO 10
      END
```

U programu je promenljivoj Y1 dodeljena brojna vrednost .1E9, tj. 10^8 (prva naredba ispod naredbe sa obeležjem 300). A zatim promenljivim Y2, Y3, i Y4 dodeljena je brojna vrednost promenljive Y1. Ovo je moglo biti zapisano i u obliku

```
Y1 = .1E9
Y2 = .1E9
Y3 = .1E9
Y4 = .1E9
```

Medjutim, zapis sproveden u programu kraći je i pruža manje šansi za greške pri bušenju kartica. O ovome treba voditi računa pri pisanju FORTRAN-programa, i gde je god to moguće treba što kraće zapisati svaku FORTRAN-naredbu.

Po navedenom programu izračunato je 6 vrednosti funkcija. Argumenti se unose pod opisom F7.4. Medjutim, u polju, na kartici, gde se

buši brojna vrednost argumenta decimalna tačka se može nalaziti u bilo kojoj koloni, i sve brojne vrednosti će biti korektno pročitane. Ovo je omogućeno time što ukoliko postoji neusaglašenost izmedju broja decimalnih mesta zadatih opisom i odredjenih decimalnom tačkom na kartici, biće važeći položaj decimalne tačke na kartici, a ne onaj zadat opisom. Ovo je ilustrovano i zadatim ulaznim podacima, gde se vidi da decimalna tačka nije u istoj koloni kartice:

```
6
2       4.2       -1.4       0.8
0       3.9       -5.0       4.4
4         1.0       1.25      -3.6
1        -0.4       1.05      -0.99
3          1.35      0.62     -4.27
4        -6.20       0.48       1.256
```

Izlazni rezultati, za navedene ulazne podatke, štampani su u obliku tabele:

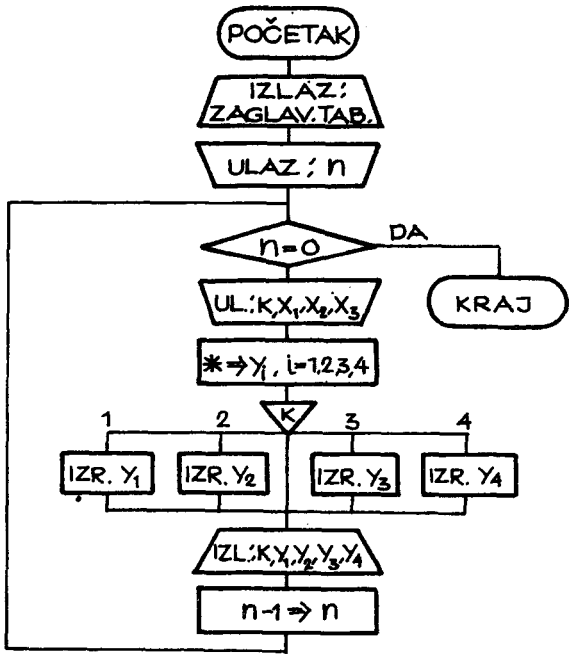
K	Y1	Y2	Y3	Y4
2	*********	42.920	*********	*********
0	*********	*********	*********	*********
4	*********	*********	*********	-132.060
1	7.548	*********	*********	*********
3	*********	*********	10.705	*********
4	*********	*********	*********	-51.398

Prva kartica ulaznih podataka sadrži broj n, koji u ovom slučaju ima vrednost 6. Druga kartica sadrži argumente za koje se vrednost funkcije računa po formuli (2). U pvroj koloni treće kartice nalazi se nula, što znači da neće doći do računanja ni po jednoj od formula (1), (2), (3) i (4). Ostale kartice sadrže argumente za koje se računaju vrednosti funkcija redom po formulama (4), (1), (3) i (4).

b) Drugo rešenje: sa korišćenjem naredbe promenljivog bezuslovnog prelaska.

Šema algoritma, u ovom slučaju, prikazana je na sl. 4. 10. 4. Algoritam je isti kao i na sl. 4. 10. 3, samo što je ispitivanje vrednosti promenljive k jednostavnije. FORTRAN-program zapisan po algoritmu na sl.4.10.4. ima sledeći izgled:

```
      WRITE(6,100)
  100 FORMAT(3H1 K,9X,'Y1',10X,'Y2',10X,'Y3',10X,'Y4'/)
      READ(5,200) N
  200 FORMAT(I2)
   10 IF(N) 20,20,30
   20 STOP 11
   30 READ(5,300) K, X1, X2, X3
  300 FORMAT(I2,3X,3(3X,F7.4))
      Y1 = +0.1E+09
      Y2 =  0.1E+9
      Y3 =   .1E09
      Y4 =   .1E9
      GO TO (11,22,33,44), K
      GO TO 70
   11 Y1 = 2.*X1**2+3.*X2**2+4.*X3**2
      GO TO 70
   22 Y2 = (X1-X2)**2+(X3-X1)**2
      GO TO 70
   33 Y3 = 4.*(X1-X2)*(X2-X3)+X1*X2*X3
      GO TO 70
   44 Y4 = (X1+X2+X3)**3-8.*X1*X2*X3**2
   70 WRITE(6,400) K, Y1, Y2, Y3, Y4
  400 FORMAT(' ',I2,2X,4(3X,F9.3))
      N = N-1
      GO TO 10
      END
```



Sl. 4. 10. 4

Neka objašnjenja programa:

1) Brojna vrednost 10^8 promenljivim Y1, Y2, Y3 i Y4 dodeljena je aritmetičkim naredbama (prve četiri naredbe iza naredbe sa obeležjem 300), u kojima je konstanta na desnoj strani zapisana u različitim oblicima. Ovo ilustruje ekvivalente oblike zapisa iste konstante.

2) Znaci blanko (medjuprostor) su bez značaja ispred i u okviru zapisa naredbe.

3) Iza naredbe STOP može stajati proizvoljan, maksimum petocifreni broj. Ovo je ilustrovano naredbom STOP sa obeležjem 20, iza koje stoji broj 11 (proizvoljno izabran).

Obe navedene varijante programa za iste ulazne podatke daju iste izlazne rezultate. Prema tome, ovi programi su medju sobom ekvivalentni, s tim što druga varijanta predstavlja kraći, pa prema tome i bolji zapis programa.

4.11. Dalje mogućnosti naredbe ulaza

4.11.1. Greške na ulazu

Naredba ulaza omogućuje dodeljivanje brojnih vrednosti promenljivim sa spoljnih nosioca informacija - kartica. Registrovanje podataka na spoljnim nosiocima informacija-karticama vrši se nezavisno od računara. U pripremi bušenih kartica može doći do grešaka koje mogu biti otkrivene pri čitanju kartica na čitaču kartica. Po otkrivenoj grešci na ulazu, može se upravljanje izvršavanjem programa preneti na deo programa u kojem se razrešava nastala greška na ulazu. U ovom slučaju naredba ulaza piše se u obliku

READ(i, j, ERR = n) lista (4.11.1)

U naredbi (4.11.1) dopisan je deo ERR = n, a ostali deo naredbe je ranije objašnjen. ERR = je službeni niz simbola, a n obeležje izvršne FORTRAN-naredbe. Skraćenica ERR je uzeta od engleske reči ERRor (greška). U slučaju otkrivanja greške na ulazu upravljanje se prenosi na FORTRAN-naredbu sa obeležjem n. Greške na ulazu mogu biti prouzrokovane iz dva razloga:

- bušenjem nevažećeg koda u koloni kartice, ili
- neregularnim položajem bušotina na kartici.

4.11.2. Kraj ulaznih podataka

Kraj ulaznih podataka može se kontrolisati programski na dva načina:

a) unošenjem broja ulaznih podataka, pre početka unošenja podataka čiji se kraj kontroliše. U ovom slučaju kraj ulaznih podataka konstatuje se prebrojavanjem unetih podataka (vidi primer na kraju odeljka 4.10).

b) U paketu ulaznih kartica sadržaj zadnje kartice može biti karakterističan. Kraj ulaznih podataka u ovom slučaju raspoznaje se identifikacijom karakterističnog sadržaja zadnje kartice (vidi primer na kraju odeljka 4.8.5).

Pored ovih načina raspoznavanja kraja ulaznih podataka, ovo se može ostvariti i naredbom ulaza

READ(i, j, END=n) lista (4.11.2)

gde je END = službeni niz simbola, a n obeležje jedne izvršne FORTRAN-naredbe u programu. Naredba (4.11.2), po unošenju svih ulaznih podataka, vrši prenošenje upravljanja na FORTRAN-naredbu sa obeležjem n.

END i ERR mogu se po želji pisati u READ-naredbi. Moraju se pisati iza obeležja FORMAT-naredbe j, a njihov redosled navodjenja je bez značaja. Tako se pored (4.11.1) i (4.11.2) mogu pisati i oblici

READ(i, j, END=n_1 , ERR=n_2) lista (4.11.3)

ili

READ(i, j, ERR=n_2 , END=n_1) lista (4.11.4)

Naredbe (4.11.3) i (4.11.4) su ekvivalentne.

4.12. Deklarisanje vrste promenljive

Unutrašnja konvencija FORTRAN-jezika razdvaja promenljive po vrsti, na celobrojne i realne, u zavisnosti od prvog slova imena promeljive. Medjutim, nije teško pretpostaviti da ime promenljive, koje odgovara pri-

rodi promenljive u problemu koji se rešava, može biti u suprotnosti sa unutrašnjom konvencijom o vrsti promenljive. Tako, MASA, kao ime promenljive, po unutrašnjoj konvenciji jeste celobrojna promenljiva,a po prirodi problema, predstavlja fizičku veličinu koja po pravilu nije ceo broj.

Da bi se omogučilo imenovanje promenljivih u suprotnosti sa unutrašnjom konvencijom, uvode se opisne naredbe kojima se može deklarisati vrsta promenljive po želji programera. Deklaracija vrste promenljive opisnim naredbama, može biti: eksplicitna i implicitna. I jedna i druga deklaracija je starija od unutrašnje konvencije FORTRAN-jezika.

Opisne naredbe za deklaraciju vrste promenljive pišu se na početku programa.

4.12.1. Eksplicitna deklaracija

Eksplicitna deklaracija vrste promenljive omogućuje deklarisanje izvesnih imena promenljivih kao celobrojnih, odnosno realnih promenljivih. Opšti oblik ove opisne naredbe je

vrsta lista (4.12.1)

gde je

vrsta - službena reč INTEGER ili REAL,

lista - spisak imena promenljivih, medju sobom razdvojenih zarezima, koje se deklarišu po vrsti.

Tako opisne naredbe

```
INTEGER A,GODINA,B12
REAL JOT,MASA,I6
```

deklarišu, u programu na čijem se početku nalaze, promenljive A, GODINA i B12 kao celobrojne, a JOT, MASA i I6 kao realne promenljive.

4.12.2. Implicitna deklaracija

Eksplicitna deklaracija služi za definisanje vrste promenljive po konkretnom imenu promenljive. Pored ovakve deklaracije, može se vrsta promenljive deklarisati po početnom slovu imena promenljive. Ovakva deklaracija se zove implicitna deklaracija vrste promenljive i zadaje se opisnom naredbom:

IMPLICIT lista (4.12.2)

gde je

IMPLICIT - službena reč, i ukazuje na implicitnu deklaraciju vrste promenljive,

lista - spisak sastavljen od elemenata medju sobom razdvojenih zarezima.

Elemenat liste u naredbi (4.12.2) je oblika

vrsta ($lista_1$) (4.12.3)

gde je

vrsta - službena reč INTEGER ili REAL,

$lista_1$ - spisak velikih slova engleske azbuke medju sobom razdvojenih zarezima.

Sve promenljive čija imena počinju slovima navedenim u $listi_1$, pripadaju po vrsti celobrojnim ili realnim promenljivim saglasno službenoj reči (INTEGER ili REAL), koja stoji na mesto reči vrsta ispred odgovarajuće liste.

Ako slova engleske azbuke u (4.12.3) slede u azbučnom redosledu, može se umesto navodjenja svih slova, navesti samo prvo i zadnje slovo razdvojeno povlakom, tako da elemenat liste može imati oblik

$a_1 - a_2$ (4.12.4)

gde su a_1 i a_2 velika slova engleske azbuke.

Primer

Opisna naredba

IMPLICIT REAL(I, L), INTEGER(A-F, Q)

deklariše, u programu na čijem se početku nalazi, promenljive čije ime počinje slovom I i L kao realne promenljive, a promenljive, čije ime počinje slovom A, B, C, D, E, F i Q kao celobrojne promenljive.

4.13. Tekstuelna objašnjenja u programu

4.13.1. Privremeni prekid rada i poruke operatoru

Naredba STOP prekida rad po programu bez mogućnosti nastavljanja rada po istom programu. Medjutim, ako postoji potreba da se izvrši izvesna manipulacija u toku rada programa, kao što je promena ulaznih kartica, papira na štampaču i sl. može se izvršiti privremen prekid rada po programu. Ovo obezbedjuje izvršna naredba

PAUSE (4.13.1)

gde službena reč PAUSE ima značenje privremenog prekida rada po programu. Nastavak rada po programu vrši operator odgovarajućom manipulacijom na komandnom pultu računara. U nastavku rada, po programu, izvršavanje programa počinje naredbom koja sledi iza naredbe PAUSE kojom je izvršen privremeni prekid rada po programu.

Ako postoji više naredbi za privremeni prekid rada po programu, one se mogu označiti brojevima. Tako se može pisati

PAUSE n (4.13.2)

gde je n neoznačen ceo broj koji ukazuje na naredbu kojom je izvršen privremeni prekid rada po programu.

Ako se želi izdati saopštenje kojim se objašnjava razlog privremenog prekida rada po programu, može se pisati

PAUSE literal (4.13.3)

gde je literal tekst koji će biti štampan kao poruka operatoru ili programeru.

Tako naredba

PAUSE PROMENITI PAPIR NA ŠTAMPAČU

vrši privremen prekid rada po programu, i štampa tekst koji ukazuje operatoru da treba izvršiti promenu papira na štampaču. Posle promene papira na štampaču operator zadaje nastavak rada po programu i izvršavanje programa se nastavlja naredbom koja sledi iza naredbe PAUSE.

4.13.2. Komentari u programu

Ako se želi, u programu, pisati tekst koji objašnjava program ili pojedine delove programa, radi lakšeg praćenja algoritma, može se koristiti pisanje komentara. Komentar počinje slovom C i može da sadrži slova, cifre i specijalne znake.

Kada se komentar buši na kartici, u prvoj koloni se buši slovo C, a od 2. do 80. kolone tekst komentara. Komentar se ne analizira od strane programa za prevodjenje, već u onom obliku kakav je bio na ulaznim karticama, izdaje se na štampani dokumenat na kojem se štampa FORTRAN-program pri prevodjenju na mašinski jezik (vidi glavu 2).

4.14. Primeri

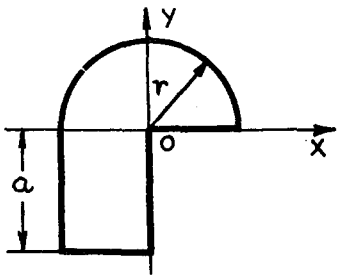
4.14.1. Izračunavanje težišta

Za više parova zadatih dimenzija r i a odrediti koordinate težišta površine na sl. 4.14.1. Veličine a i r su zadate sa dva cela i dva decimalna mesta. Rezultate štampati u obliku tebele:

IZRAČUNAVANJE TEŽIŠTA

R	A	X	Y
-	-	-	-
-	-	-	-
.	.	.	.
.	.	.	.
.	.	.	.
-	-	-	-

KRAJ PROGRAMA



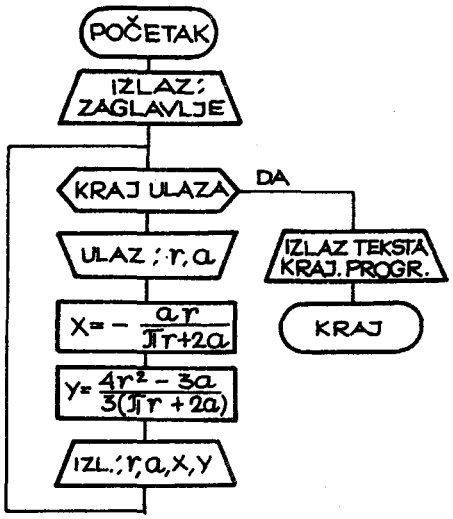
Sl. 4.14.1

Koordinate težišta na sl. 4.14.1, mogu se izračunati pomoću formula

$$x = -\frac{ar}{\pi r + 2a} \qquad (4.14.1)$$

$$y = \frac{4r^2 - 3a^2}{3(\pi r + 2a)} \qquad (4.14.2)$$

Šema algoritma prikazana je na sl. 4.14.2



Sl. 4. 14. 2

Program sastavljen po algoritmu na sl. 4. 14. 2 ima sledeći izgled

```
C     T E Z I S T E   R A V N E   F I G U R E
      WRITE(6,10)
   10 FORMAT('1',13X,'IZRACUNAVANJE TEZISTA'/
     *'0',7X,'R',9X,'A',9X,'X',9X,'Y'/)
   15 READ(5,20,END=40) R, A
   20 FORMAT(2F5.2)
      X = -A*R/(3.14*R+2.*A)
      Y = (4.*R**2-3.*A**2)/(3.*3.14*R+6.*A)
      WRITE(6,30) R, A, X, Y
   30 FORMAT(' ',4F10.2)
      GO TO 15
   40 WRITE(6,50)
   50 FORMAT('0','KRAJ PROGRAMA')
      STOP
      END
```

Za zadate ulazne podatke, izlazni rezultati se dobijaju u obliku tabele:

IZRACUNAVANJE TEZISTA

R	A	X	Y
30.00	40.00	-6.89	-2.30
30.00	30.00	-5.84	1.95
40.00	30.00	-6.47	6.65

KRAJ PROGRAMA

4.14.2. Statistički primer

Izvršeno je niz merenja. Svako merenje daje jedan podatak x_i, koji je bušen na kartici od 1. do 10. kolone. Sastaviti program koji će utvrditi broj merenja (n) i izračunati srednju vrednost

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i \qquad (4.14.3)$$

i standardno odstupanje

$$\sigma = \left(\frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right)^{\frac{1}{2}} \qquad (4.14.4)$$

kao i odstupanje pojedinih merenja od srednje vrednosti.

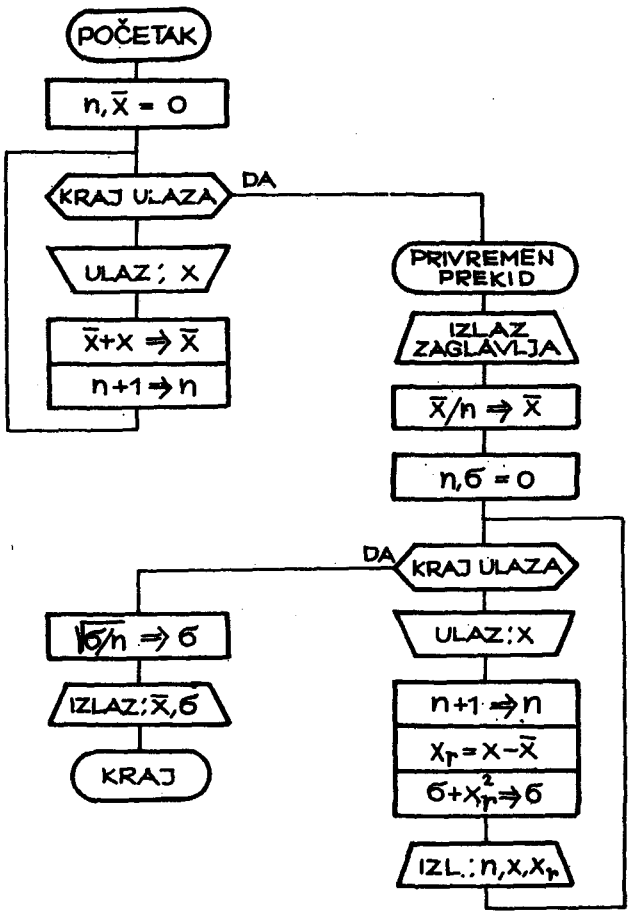
Izlazne rezultate štampati u obliku tabele:

N	X	X - NADX
1	-	-
2	-	-
.	.	.
.	.	.
.	.	.
.	.	.

SREDNJA VREDNOST = $\bar{x}$

STANDARDNO ODSTUPANJE = σ

Algoritam je prikazan na sl. 4.14.3. Kao što se vidi sa slike, algoritam se sastoji iz dva dela. U prvom delu se vrši unošenje ulaznih podataka u cilju izračunavanja srednje vrednosti, a u drugom delu se vrši ponovno unošenje ulaznih podataka u cilju izračunavanja odstupanja pojedinih merenja od srednje vrednosti i izračunavanja standardnog odstupanja.



Sl. 4. 14. 3

Program sastavljen po algoritmu na sl. 4. 14. 3 ima sledeći izgled:

```
C  PROGRAM ZA IZRACUNAVANJE
C  SREDNJE VREDNOSTI I STANDARDNOG ODSTUPANJA
C
      REAL NADX
      N = 0
      NADX = 0.
   30 READ(5,10,END=20) X
   10 FORMAT(F10.0)
      NADX = NADX+X
      N = N+1
      GO TO 30
   20 PAUSE 'POSTAVITI PONOVO ULAZNE PODATKE NA CITAC'
      WRITE(6,40)
```

```
   40 FORMAT(/////3X,'N',12X,'X',15X,'X-NADX'/)
      NADX = NADX/N
      N = 0
      SIGMA = 0.
   70 READ(5,10,END=50) X
      N = N+1
      XR = X-NADX
      SIGMA = SIGMA+XR*XR
      WRITE(6,60) N, X, XR
   60 FORMAT(' ',I3,5X,E14.7,5X,E14.7)
      GO TO 70
   50 SIGMA = SQRT(SIGMA/N)
      WRITE(6,80) NADX,SIGMA
   80 FORMAT(/' SREDNJA VREDNOST=',E14.7//
     *' STANDARDNO ODSTUPANJE=',E14.7)
      STOP
      END
```

Za zadate ulazne podatke x_i, izlazni rezultati se dobijaju u obliku tabele:

N	X	X-NADX
1	0.2190000E 03	-0.2031342E 01
2	0.2225000E 03	0.1468658E 01
3	0.2156500E 03	-0.5381348E 01
4	0.2182000E 03	-0.2831345E 01
5	0.2210000E 03	-0.3134155E-01
6	0.2243500E 03	0.3318649E 01
7	0.2265200E 03	0.5488647E 01

SREDNJA VREDNOST= 0.2210313E 03

STANDARDNO ODSTUPANJE= 0.3472305E 01

4.14.3. Izračunavanje korena transcendentne jednačine

Odrediti najmanji pozitivan koren transcendentne jednačine

$$\sin x = \frac{A}{x} \tag{4.14.5}$$

za zadate vrednosti parametra A iz intervala $[0,1;\ 1,0]$. Za izračunavanje korena primeniti iterativan postupak

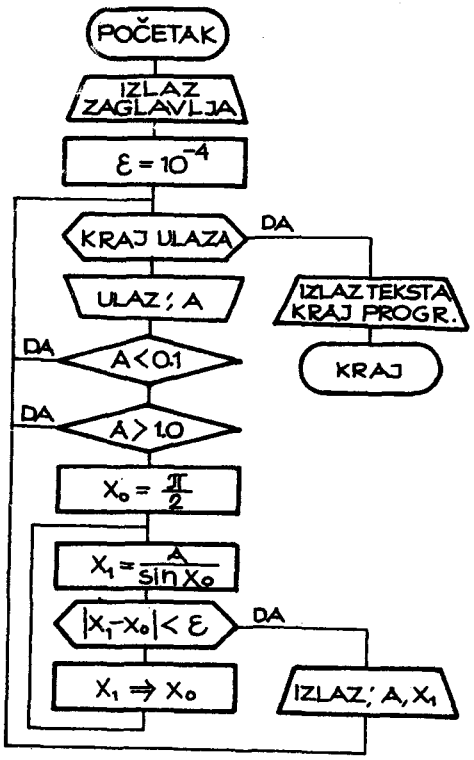
$$x_{i+1} = \frac{A}{\sin x_i}\ ,\ i = 0, 1, 2, \ldots \tag{4.14.6}$$

gde je $x_0 = \pi/2$. Iterativan postupak prekinuti kada bude ispunjen uslov

$$|x_{i+1} - x_i| < \varepsilon \tag{4.14.7}$$

gde je ε zadata tačnost.

Algoritam za izračunavanje korena jednačine (4.14.5) prikazan je na sl.4.14.4, gde je uzeto da je $\varepsilon = 10^{-4}$.



Sl. 4.14.4.

FORTRAN-program napisan po algoritmu na sl.4.14.4, ima sledeći izgled:

```
C  PRIMER   I T E R A C I O N O G   POSTUPKA
C
C
C  IZLAZ ZAGLAVLJA TABELE
C
      WRITE(6,100)
100   FORMAT('1IZRACUNAVANJE KORENA JEDNACINE '
     *' SIN(X) = A/X'///16X,'A',12X,'X'/)
C
C  ZADATA GRANICA APSOLUTNE GRESKE 'EPS'
C
      EPS=1.E-4
```

```
C
C  ULAZ PARAMETARA 'A'
C
10    READ(5,200,END=70) A
200   FORMAT(F5.2)
C
C  TESTIRANJE PARAMETRA 'A'
C
      IF(A-.1) 10,20,20
20    IF(A-1.) 30,30,10
C
C  POCETNA VREDNOST KORENA 'X'
C
30    X=3.141592*.5
C
C  ITERACIONI CIKLUS
C
40    XX = A/SIN(X)
      IF(ABS(XX-X)-EPS) 60,50,50
50    X = XX
      GO TO 40
C
C  IZLAZ PARAMETRA 'A' I KORENA 'X'
C
60    WRITE(6,300) A, X
300   FORMAT(6X,2(8X,F5.2))
      GO TO 10
C
C  ZAVRSETAK PROGRAMA
C
70    WRITE(6,400)
400   FORMAT(/' KRAJ PROGRAMA')
      STOP
      END
```

Za zadate vrednosti parametra A, izlazni rezultati se štampaju u obliku tabele:

```
IZRACUNAVANJE KORENA JEDNACINE  SIN(X) = A/X

          A          X

       0.10       0.32
       0.32       0.58
       0.68       0.88
       1.00       1.11

KRAJ PROGRAMA
```

U programu je promenljiva x_i , iz algoritma na sl. 4. 14. 4, označena sa XX.

5. PROMENLJIVE SA INDEKSIMA - NIZOVI

5.1. Definicija niza

U mnogim oblastima primene matematike, dolazi do potrebe izvodjenja operacija nad grupom brojnih podataka. Da se ne bi pojedinačno imenovali brojni podaci, ovakva grupa dobija zajedničko ime. Tako se uvodi pojam vektora, determinante i matrice. FORTRAN-jezik pruža mogućnosti lakog zapisa grupe brojnih podataka. Ovakva grupa brojnih podataka registruje se u memoriji tako što se svaki elemenat grupe registruje u jednom memorijskom registru. Neka su registri memorije adresirani redom sa $1, 2, 3, \ldots, n$, gde je n broj registara memorije. Neka grupa brojnih podataka, koja se želi registrovati, sadrži m elemenata (brojeva). Ako je prvi elemenat grupe registrovan u registru sa adresom r, tada će za registrovanje cele grupe biti upotrebljeni redom registri

$$r, \quad r+1, \quad r+2, \quad \ldots, \quad r+m-1 \qquad (5.1.1)$$

Sadržaji registara (5.1.1) jesu brojni podaci grupe, što znači da se svaka grupa brojnih podataka prikazuje u memoriji u obliku jednog niza brojeva, bez obzira na raspored brojeva u grupi.

Više podataka sa zajedničkim imenom, u FORTRAN-jeziku zove se niz. Prema tome, vektori, determinante i matrice u FORTRAN-jeziku pišu se kao nizovi.

Tako, kada se elementi matrice

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \qquad (5.1.2)$$

registruju u memoriji računara, tada se formira niz u obliku

$$a_{11}, a_{21}, a_{31}, a_{12}, a_{22}, a_{32}, a_{13}, a_{23}, a_{33}, a_{14}, a_{24}, a_{34} \qquad (5.1.3)$$

Ako je prvi elemenat a_{11} registrovan u registru čija je adresa 300, onda će za registrovanje matrice A, biti angažovani registri

$$300, \quad 301, \quad 302, \ \ldots, \ 311 \qquad (5.1.4)$$

čiji će sadržaji biti redom elementi niza (5.1.3).

5.1.1. Ime niza i indeksi

Ime niza definiše se na isti način kao i ime promenljive. Medjutim, ako jedno ime predstavlja ime promenljive, to se ne može koristiti isto ime i za ime niza.

Da bi smo označili pojedine elemente niza uvode se indeksi. Vrednost indeksa je ceo broj kojim se jednoznačno odredjuje jedan elemenat niza. Broj indeksa niza zove se dimenzija niza. Niz može biti jednodimenzionalan, dvodimenzionalan itd. do maksimum sedmodimenzionalan. Tako da je opšti oblik promenljive sa indeksima

$$\text{ime (lista)} \qquad (5.1.5)$$

gde je

ime - ime niza koje se definiše, kao i ime promenljive,

lista - spisak indeksa medju sobom razdvojenih zarezima. Broj indeksa može biti najmanje jedan, a najviše sedam.

Indeks može biti aritmetički izraz, čija se brojna vrednost izračunava, a zatim za odredjivanje elementa niza prevodi u celobrojnu konstantu. Vrednost ovako dobijenog celog broja mora biti veća od nule i manja ili jednaka najvećoj predvidjenoj vrednosti indeksa. Najveće predvidjene vrednosti indeksa niza moraju biti zadate na početku programa za svaki niz koji

se pojavljuje u programu. Ovo se zadaje opisnom naredbom

DIMENSION lista (5.1.6)

gde je

DIMENSION - službena reč, i odredjuje opisnu naredbu za definisanje maksimalnih vrednosti indeksa,

lista - spisak sastavljen od elemenata, medju sobom razdvojenih zarezima.

Elemenat liste (5.1.6) piše se u obliku

ime (lista) (5.1.7)

gde je

ime - ime niza,

lista - spisak celih neoznačenih brojeva, koji predstavljaju najveće moguće vrednosti indeksa, medju sobom razdvojenih zarezima.

Tako se može pisati

DIMENSION A(10,10), VEK(50)

što znači da će u memoriji računara biti rezervisano 100 registara za elemente matrice A i 50 registara za komponente vektora VEK.

5.1.2. Vrsta niza

Po vrsti nizovi se dele na celobrojne i realne. Ako su svi elementi niza celi brojevi, onda je niz celobrojan, a ako su elementi niza realni brojevi i niz je realan. Vrsta niza je odredjena unutrašnjom konvencijom FORTRAN-jezika, kao i vrsta promenljive, tj. ako ime niza počinje slovom I, J, K, L, M ili N, onda je niz celobrojan, a u suprotnom niz je realan.

Ako se želi nizu dodeliti ime koje je u suprotnosti sa unutrašnjom konvencijom FORTRAN-jezika, može se koristiti eksplicitna ili implicitna deklaracija vrste niza. Eksplicitna deklaracija vrste niza vrši se opisnim naredbama REAL ili INTEGER, objašnjenim u odeljku 4.12.1. Ime niza se navodi u jednoj od ove dve naredbe u zavisnosti od toga koje je vrste niz.

Ako se deklariše realan niz, navodi se ime niza kao elemenat liste naredbe REAL, ili ako se deklariše celobrojni niz, navodi se ime niza kao elemenat liste naredbe INTEGER. Opisnim naredbama REAL i INTEGER pored deklaracije vrste niza mogu se zadati i najveće moguće vrednosti indeksa odgovarajućeg niza. Tako u ovom slučaju elemenat u listi ovih naredbi može imati oblik (5.1.7). Ako su maksimalne vrednosti indeksa definisane u naredbi REAL ili INTEGER onda ovo ne treba posebno pisati u naredbi DIMENSION. Medjutim, za svaki niz koji se koristi u programu moraju se definisati maksimalne vrednosti indeksa, odakle sledi da se svako ime niza mora pojaviti ili u naredbi REAL ili INTEGER ili DIMENSION. Tako se može pisati

```
REAL M(5,12), J(20,20)                    (5.1.8)
INTEGER A(5,5), T(10)                     (5.1.9)
```

Naredba REAL deklariše matrice M i J kao realne nizove, a naredba INTEGER deklariše matricu A i vektor T kao celobrojne nizove.

Implicitna deklaracija vrste niza vrši se na isti način kao i implicitna deklaracija vrste promenljive (vidi odeljak 4.12.2). To se postiže opisnom naredbom IMPLICIT, kojom se vrsta niza može deklarisati po početnom slovu u imenu niza. Tako

```
IMPLICIT REAL (J,M), INTEGER (A,T)        (5.1.10)
```

deklariše sve promenljive i nizove čija imena počinju slovom J ili M kao realne promenljive, odnosno nizove, i sve promenljive i nizove čija imena počinju slovom A ili T kao celobrojne promenljive, odnosno nizove.

Treba uočiti razliku izmedju deklaracije (5.1.10) i (5.1.8) odnosno (5.1.9). Deklaracija (5.1.10) deklariše sve promenljive i nizove čija imena počinju slovom J ili M odnosno A ili T, dok deklaracija (5.1.8) i (5.1.9) deklariše samo nizove čija su imena M i J, odnosno A i T.

Eksplicitna deklaracija vrste promenljive i niza ima najviši prioritet, a zatim implicitna deklaracija vrste i na kraju unutrašnja konvencija FORTRAN-jezika.

5.2. Jednodimenzionalni nizovi

Elemenat jednodimenzionalnog niza ima opšti oblik

ime (i) (5.2.1)

gde je

ime - ime niza,

i - indeks niza.

Maksimalna vrednost indeksa niza može biti zadata u naredbi DIMENSION ili REAL ili INTEGER. Niz, u ovim naredbama, navodi se kao elemenat liste u obliku

ime (i_{max}) (5.2.2)

gde je i_{max}ceo neoznačen broj koji odredjuje maksimalnu vrednost indeksa, odnosno broj elemenata niza. Tako naredba

DIMENSION A(20), VEK (50) (5.2.3)

definiše 20 elemenata niza A:

A(1), A(2), A(3),..., A(20)

odnosno 50 elemenata niza VEK:

VEK(1), VEK(2),..., VEK(50).

Elemenat niza zove se promenljiva sa indeksom. Svaki elemenat niza registruje se u jednom memorijskom registru. Sve što je rečeno da važi za običnu promenljivu važi i za promenljivu sa indeksom.

5.2.1. Jednodimenzionalni nizovi u listi ulazno-izlaznih naredbi

Nizovi se pojavljuju kao elementi liste ulazne naredbe, kada se vrši dodeljivanje brojne vrednosti elementima niza sa ulaza. Ako se vrši izdavanje brojnih vrednosti pojedinih elemenata niza tada se nizovi pojavljuju

u listi izlazne naredbe. Elementi jednog niza mogu se pojaviti na više načina u listi ulazne, odnosno izlazne naredbe.

a) Ako elementi niza ne slede jedan za drugim po odredjenom zakonu, tada se mogu navoditi u listi na isti način kao imena promenljivih. Tako se može pisati

READ (5,10) A(4), A(2), A(8), A(15)

što znači da će elementima A(4), A(2), A(8) i A(15) niza A biti dodeljene brojne vrednosti sa ulaza.

b) Ako elementi niza koji se navode u listi slede u redosledu, počev od elementa m_1 do zaključno sa elementom m_2, tada se može pisati element liste, u ulaznoj, odnosno izlaznoj naredbi u obliku

$$(\text{ime}(i), i = m_1, m_2) \qquad (5.2.4)$$

gde je

ime - ime niza,

i - ime celobrojne promenljive,

m_1, m_2 - celi neoznačeni brojevi ili celobrojne promenljive.

Zapis (5.2.4) ima isti efekat kao da su elementi niza u listi, navedeni u redosledu

$$\text{ime}(m_1), \text{ime}(m_1+1), \ldots, \text{ime}(m_2) \qquad (5.2.5)$$

Ako se svi elementi niza žele navesti u listi, onda se oblik (5.2.4) svodi na

$$(\text{ime}(i), i = 1, i_{max}) \qquad (5.2.6)$$

Umesto oblika (5.2.6) može se pisati i samo ime niza u listi, tj.

$$\text{ime} \qquad (5.2.7)$$

Prema tome, ako u listi ulazne, odnosno izlazne naredbe stoji samo ime niza, to ima isti efekat kao da su navedeni elementi niza počevši od prvog do poslednjeg. Informacija o tome koliko niz elemenata sadržana

je u opisnim naredbama DIMENSION ili REAL ili INTEGER.

c) Ako elementi niza koji se navode u listi ne slede jedan iza drugog, može se u listi pisati oblik

$$(\text{ime}(i), i = m_1, m_2, m_3) \qquad (5.2.8)$$

gde sve oznake imaju isto značenje kao i u (5.2.4), a uvedena veličina m_3, može biti ceo neoznačen broj ili celobrojna promenljiva. Zapis (5.2.8) ima isti efekat kao da su elementi niza nabrojani u redosledu:

$$\text{ime}(m_1),\ \text{ime}(m_1+m_3),\ \text{ime}(m_1+2m_3),\ \ldots,\ \text{ime}(m_1+km_3) \qquad (5.2.9)$$

gde je

$$k = \left[\frac{m_2 - m_1}{m_3}\right] \qquad (5.2.10)$$

gde srednja zagrada označava celobrojni deo količnika.

Prema tome, oblik (5.2.8) sadrži k+1 elemenat niza, koji su navedeni u (5.2.9).

5.2.2. Elementi niza u aritmetičkoj naredbi

Elementi niza mogu se naći na levoj ili na desnoj strani znaka jednakosti u aritmetičkoj naredbi. Pojava elementa niza u aritmetičkoj naredbi ima isto značenje kao i pojava obične promenljive. Ako se elemenat niza pojavljuje na desnoj strani tada predstavlja argumenat aritmetičkog izraza, a ako se pojavljuje na levoj strani onda je to elemenat niza kojem se dodeljuje brojna vrednost izračunata aritmetičkom naredbom.

Primer

Na kartici je zadato 10 brojeva u formatu F8.3. Sastaviti program, koji će uneti brojeve sa kartice u memoriju računara, štampati njihove brojne vrednosti i izračunati i štampati njihov zbir.

Ovaj zadatak ćemo rešiti na dva načina: bez korišćenja nizova i sa korišćenjem nizova u programu.

a) Prvo rešenje: bez korišćenja nizova u programu

```
      READ(5,10) X1,X2,X3,X4,X5,X6,X7,X8,X9,X10
   10 FORMAT(10F8.3)
      Y=X1+X2+X3+X4+X5+X6+X7+X8+X9+X10
      WRITE(6,20) X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,Y
   20 FORMAT(' ',10F12.3//' Y=',F9.3)
      STOP
      END
```

b) Drugo rešenje: sa korišćenjem nizova u programu

```
      DIMENSION X(10)
      READ(5,10) X
   10 FORMAT(10F8.3)
      Y=0.
      I=1
   13 Y=Y+X(I)
      IF(I-10) 11,12,11
   11 I=I+1
      GO TO 13
   12 WRITE(6,20) X,Y
   20 FORMAT(' ',10F12.3//' Y=',F9.3)
      STOP
      END
```

Prvo rešenje zahteva uvodjenje promenljivih (X1, X2, ..., X10) kojima se dodeljuju brojne vrednosti zadate na kartici. Iste promenljive se moraju navesti u aritmetičkoj naredbi da bismo izračunali njihovu sumu. Medjutim, u drugom rešenju, gde su korišćeni nizovi uvodi se jedno ime niza (X), a promenom indeksa postiže se isti efekat kao i u prvom rešenju. Prvo rešenje je moglo biti primenjeno u slučaju 10 brojeva, medjutim, da se radilo o većem broju brojnih podataka, napr. 1000, ovo bi bilo nemoguće zapisati uvodjenjem 1000 imena promenljivih, dok se drugo rešenje bitno ne menja povećanjem broja brojnih podataka. Tako, ako bi se radilo o 1000 brojeva niz X bi bio definisan u DIMENSION-naredbi kao X(1000) i predvidjeno sumiranje bi se moglo izvršiti zamenom broja 10 sa 1000 u naredbi IF.

5.3. Ciklične algoritamske strukture

5.3.1. Naredba za opis programskog ciklusa

Ciklične algoritamske strukture vrlo se često javljaju pri sastavljanju algoritama za rešavanje različitih zadataka. Kao što je objašnjeno u odeljku 1.4., izlazni kriterijum iz ovakvih ciklusa može biti različite prirode. Najčešće, su ovi kriterijumi brojačkog karaktera, tako da se pomoću njih kontroliše broj ponavljanja ciklusa. Kada se ciklus izvrši zadati broj puta, vrši se izlaz iz ciklusa. Da bi se omogućilo lako pisanje ovakvih ciklusa u FORTRAN-jeziku postoji posebna naredba za njihovo definisanje, to je naredba oblika

$$\text{DO n i}=m_1, m_2, m_3 \qquad (5.3.1)$$

gde je

DO - službena reč FORTRAN-jezika,

n - obeležje jedne izvršne FORTRAN-naredbe, koja se nalazi iza naredbe (5.3.1),

i - ime celobrojne promenljive,

m_1, m_2 m_3- celi neoznačeni brojevi ili imena celobrojnih promenljivih.

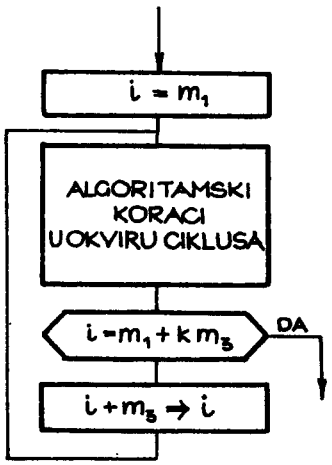
Naredba (5.3.1) ima sledeće značenje: naredbe koje se nalaze ispod naredbe (5.3.1) zaključno sa naredbom čije je obeležje n, čine programski ciklus, koji će se izvršiti k+1, puta gde je

$$k = \left[\frac{m_2 - m_1}{m_3}\right] \qquad (5.3.2)$$

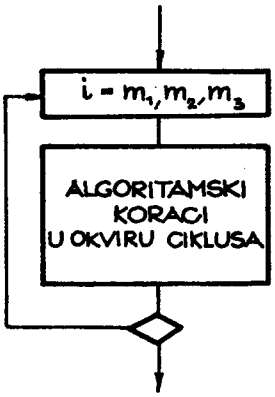
Promenljiva i pri prvom izvršavanju ciklusa ima vrednost m_1, a pri svakom sledećem izvršavanju prethodna vrednost promenljive i povećava se za m_3, tako da promenljiva i uzima redom vrednosti

$$i = m_1, \; m_1+m_3, \; m_1+2m_3, \ldots, \; m_1+km_3 \qquad (5.3.3)$$

Promenljiva i zove se indeks ciklusa, a m_1 je početna vrednost indeksa, m_2 gornja granica indeksa, a m_3 priraštaj indeksa. Na sl. 5.3.1 prikazana je šema ciklične algoritamske strukture koja se realizuje naredbom (5.3.1).



Sl. 5.3.1



Sl. 5.3.2

Šema na sl. 5.3.1 sadrži detalje koji se uvek ponavljaju kod ove vrste programskih ciklusa, kao što je postavljanje početne vrednosti indeksa ispitivanje izlaznog kriterijuma i povećanje indeksa za navedeni priraštaj. Da bi se izbeglo ovo ponavljanje uvedena je ekvivalentna šema na sl. 5.3.2. Iz šeme na sl. 5.3.1. vidi se da će se algoritamski koraci, koji čine ciklus, izvršiti najmanje jedanput, bez obzira na odnos izmedju veličina m_1 i m_2.

Ako je $m_3 = 1$ ne mora se navoditi u naredbi (5.3.1), pri čemu se ne piše ni zarez ispred m_3, pa se oblik (5.3.1) svodi na

$$\text{DO n i} = m_1, m_2 \qquad (5.3.4)$$

gde je značenje pojedinih simbola isto kao i u naredbi (5.3.1) s tim što je k uvek ceo broj i iznosi

$$k = m_2 - m_1 \qquad (5.3.5)$$

pa će algoritamski korak za ispitivanje izlaznog kriterijuma u ciklusu na sl. 5.3.1 biti

$$i = m_2 \qquad (5.3.6)$$

Prema tome, u slučaju naredbe ciklusa u obliku (5.3.4) prolazi kroz ciklus vrše se sa vrednostima indeksa

$$i = m_1,\ m_1+1,\ m_1+2,\ \ldots,\ m_2 \qquad (5.3.7)$$

Zadnja naredba u ciklusu (naredba sa obeležjem n) ne sme biti jedna od sledećih naredbi

GO TO
PAUSE
STOP
IF (po vrednosti aritmetičkog izraza)
DO

Primer

Ranije navedeni primer, na kraju odeljka 5.2.2., u kojem se vrši sabiranje 10 brojeva, kao elemenata niza X, može se rešiti primenom naredbe ciklusa. Rešenje u FORTRAN-jeziku ima sledeći izgled:

```
      DIMENSION X(10)
      READ(5,10) X
10    FORMAT(10F8.3)
      Y=0.
      DO 11 I=1,10
11    Y=Y+X(I)
      WRITE(6,20) Y
20    FORMAT(' Y=',F9.3)
      STOP
      END
```

U ovom slučaju programski ciklus sadrži jednu naredbu

$$Y=Y+X(I)$$

koja se izvršava 10 puta, za vrednosti promenljive $I=1,2,3,\ldots,10$. Kako je pre ulaska u ciklus promenljivoj Y dodeljena vrednost nula, a svaki prolazak kroz ciklus povećava prethodnu vrednost promenljive Y za odgovarajuću vrednost elementa niza X, to će po izlasku iz ciklusa vrednost promenljive Y biti suma zadatih 10 elemenata niza X.

5.3.2. Naredba bez dejstva

U FORTRAN-jeziku postoji mogućnost zapisa naredbe bez dejstva. Ova naredba se piše u obliku

CONTINUE (5.3.8)

gde je

CONTINUE - službena reč i označava naredbu bez dejstva.

Izvršavanje ove naredbe ne proizvodi nikakve promene u računaru, već samo prelazak na naredbu, koja treba da se izvrši iza ove naredbe. Ovde treba razlikovati dva slučaja:

1) Ako je naredba (5.3.8) zadnja naredba programskog ciklusa, tada će posle ove naredbe doći do ponavljanja ciklusa, ako to zahteva izlazni kriterijum, odnosno do prelaska na naredbu koja sledi iza naredbe (5.3.8), ako se izlazi iz ciklusa.

2) Ako naredba (5.3.8) nije zadnja naredba ciklusa, tada izvršavanje naredbe (5.3.8) predstavlja prelazak na naredbu koja sledi iza ove naredbe.

Naredba (5.3.8) koristi se u programiranju, najčešće u sledeća dva slučaja:

1) Ako bi zadnja naredba ciklusa trebalo da bude neka od nedozvoljenih naredbi, tada se kao zadnja naredba može koristiti naredba (5.3.8).

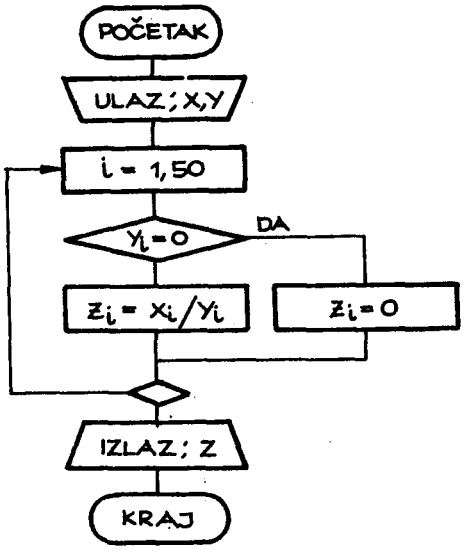
2) Ako se želi izbaciti iz programa naredba koja ima obeležje, tada, da se ne bi menjala obeležja, može se na mesto izbačene naredbe ubaciti naredba (5.3.8) sa istim obeležjem.

Primer

Zadata su dva niza brojeva x_i i y_i, $i = 1,2,\ldots,50$. Izračunati niz

$$z_i = \frac{x_i}{y_i}, \quad i = 1,2,\ldots,50$$

pri čemu ako je $y_k = 0$, uzeti da je $z_k = 0$, $k \in \{1,2,\ldots,50\}$. Elementi nizova x_i i y_i su brojevi sa maksimum 3 cela i 3 decimalna mesta. Neka se na jednoj kartici nalazi 10 ovakvih brojeva. Prema tome, ulazni podaci



Sl. 5.3.3.

nalaze se na 10 kartica i to na prvih 5 nalaze se elementi niza x, a na drugih 5 elementi niza y. Blok-šema algoritma prikazana je na sl. 5.3.3.

Program sastavljen po algoritmu na sl. 5.3.3. ima sledeći izgled:

```
      DIMENSION X(50),Y(50),Z(50)
      READ(5,50) X,Y
   50 FORMAT(10F8.3)
      DO 30 I=1,50
      IF(Y(I)) 10,20,10
   10 Z(I)=X(I)/Y(I)
   30 CONTINUE
      WRITE(6,60) (Z(I),I=1,50)
   60 FORMAT(' ',7X,'Z'//(' ',E14.7))
      STOP
   20 Z(I)=0.
      GO TO 30
      END
```

U ovom primeru naredba sa obeležjem 10 sadrži operaciju deljenja. Ova operacija se nalazi u ciklusu, jer je treba izvršiti 50 puta da bi smo formirali niz Z. Medjutim, u operaciji deljenja delilac ne sme biti jednak nuli. Zato se pre dolaska na operaciju deljenja vrši ispitivanje vrednosti delioca (elemenat niza Y), i ako je njegova vrednost različita od nule dola-

zi se na naredbu 10, a zatim na naredbu CONTINUE koja je zadnja naredba ciklusa. Ako je vrednost indeksa I < 50 vrši se ponavljanje ciklusa, a ako je I = 50 vrši se izlazak iz ciklusa, tj. prelazak na naredbu koja sledi iza naredbe CONTINUE. Ako je vrednost delioca (elemenat niza Y) jednaka nuli, vrši se prelazak na naredbu sa obeležjem 20, kojom se postavlja nula kao vrednost odgovarajućeg elementa niza Z. Posle ovoga vrši se prelazak na zadnju naredbu u ciklusu, čime se obezbedjuje normalno izvršavanje ciklusa.

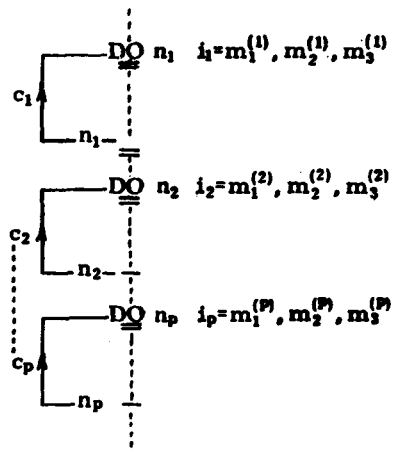
5.3.3. Odnos dva i više ciklusa

Već smo videli u prethodnom primeru da naredbe koje čine ciklus ne moraju biti zapisane izmedju DO naredbe i zadnje naredbe ciklusa. Prema tome, nekom naredbom uslovnog ili bezuslovnog prelaska može se privremeno izaći iz ciklusa i ponovo vratiti u ciklus. Ovaj povratak u ciklus može biti na ma koju naredbu u okviru ciklusa, uključujući i zadnju naredbu ciklusa. Medjutim, treba voditi računa da se vrednost promenljive koja predstavlja indeks ciklusa ne sme menjati naredbama u okviru ciklusa.

Posebno je važno pravilno koristiti cikluse kada ih ima veći broj u programu. Ovde ćemo razlikovati tri moguća slučaja u odnosu izmedju dva i više ciklusa:

1) Linijska kompozicija programskih ciklusa.

Za dva ili više ciklusa koji slede jedan iza drugog u programu, kaže se da čine linijsku kompoziciju ciklusa. U ovom slučaju odnos ciklusa je sledeći

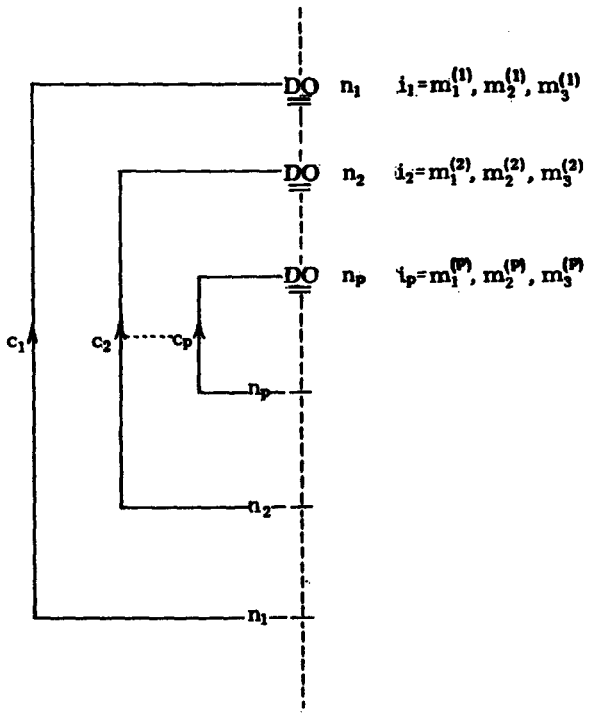


Za ovakvu kompoziciju od p programskih ciklusa C_1, $C_2, \ldots, C_p$ važi sledeće:

- broj ciklusa p u ovakvoj kompoziciji je neograničen,
- obeležja n_1, $n_2, \ldots, n_p$ su medju sobom različita,
- dva ili više indeksa iz skupa $\{i_1, i_2, \ldots, i_p\}$ mogu imati ista imena.

2) Koncentrična kompozicija programskih ciklusa.

Za dva ili više ciklusa, koji se nalaze jedan u okviru drugoga, kaže se da obrazuju koncentričnu kompoziciju programskih ciklusa. U ovom slučaju odnos ciklusa je sledeći:

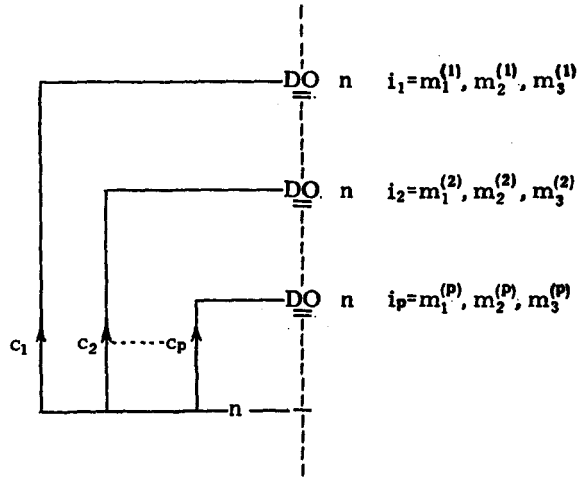


Za ovakvu kompoziciju od p programskih ciklusa C_1, $C_2, \ldots, C_p$ važi sledeće:

- neki od ciklusa mogu imati zajedničku zadnju naredbu, a to znači da neka od obeležja iz skupa $\{n_1, n_2, \ldots, n_p\}$ mogu biti jednaka,

- indeksi $i_1, i_2, \ldots, i_p$ moraju imati različita imena.

U koncentričnoj kompoziciji programskih ciklusa mogu biti jednaka obeležja zadnjih naredbi ciklusa, samo ako ne dovode do sečenja pojedinih ciklusa. Ako su sva obeležja medju sobom jednaka, tada koncentrična kompozicija dobija sledeći izgled:



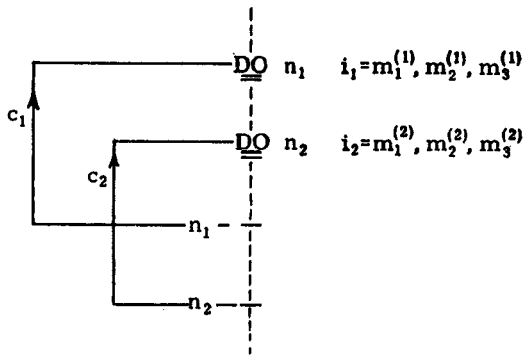
Ako su izlazni kriterijumi ciklusa $C_1, C_2, \ldots, C_p$, koji grade koncentričnu kompoziciju takvi da ulazak u ciklus C_k, znači prolazak kroz ciklus q_k puta, tada ulazak u koncentričnu kompoziciju ciklusa definiše sledeći broj prolazaka kroz pojedine cikluse:

- kroz ciklus C_1 prolazi se q_1 puta,
- kroz ciklus C_2 prolazi se $q_1 . q_2$ puta,
- kroz ciklus C_k prolazi se $q_1 . q_2 \ldots q_k$ puta, i
- kroz ciklus C_p prolazi se $q_1 . q_2 \ldots q_p$ puta.

Ovakav broj prolazaka kroz ciklus sledi iz činjenice što se za svaki prolazak kroz ciklus C_{k-1} ciklus C_k izvrši q_k puta.

3) Nedozvoljena kompozicija programskih ciklusa.

Dva ili više ciklusa medju sobom se ne smeju seći. Tako za dva ciklusa C_1 i C_2 nije dozvoljen sledeći odnos:



5.4. Dvodimenzionalni nizovi

Elemenat dvodimenzionalnog niza ima opšti oblik

$$\text{ime}(i_1, i_2) \qquad (5.4.1)$$

gde je

ime - ime niza,

i_1, i_2 - indeksi niza.

Maksimalna vrednost indeksa niza može biti zadata u naredbi DIMENSION ili REAL ili INTEGER. Dvodimenzionalni niz u ovim naredbama navodi se kao elemenat liste u obliku

$$\text{ime}(i_{1max}, i_{2max}) \qquad (5.4.2)$$

gde su i_{1max} i i_{2max} celi neoznačeni brojevi koji odredjuju maksimalne vrednosti indeksa.

Tako, naredba

DIMENSION A(10, 10), B(20, 8)

definiše 100 elemenata matrice A:

A (1, 1), A (1, 2), ..., A (1, 10)
A (2, 1), A (2, 2), ..., A (2, 10)
A (10, 1), A (10, 2), ..., A(10, 10)

i 160 elemenata matrice B:

B (1,1), B (1,2), ..., B (1,8)
B (2,1), B (2,2), ..., B (2,8)
B (20,1), B (20,2), ..., B (20,8)

5.4.1. Dvodimenzionalni nizovi u listi ulazno-izlaznih naredbi

Kao i elementi jednodimenzionalnih nizova (vidi 5.2.1), tako i elementi dvodimenzionalnih nizova mogu se pojaviti u listi ulazno-izlazne naredbe na više načina:

a) Ako se pojedinačni elementi nizova pojavljuju u listi tada se vrši navodjenje odgovarajućih elemenata. Tako se može pisati

WRITE (6,50) A (35,10), B (18,4), C (3,3)

što znači da treba izdati brojne vrednosti odgovarajućih elemenata matrica A, B i C.

b) Ako elementi matrice koji se navode u listi slede jedan za drugim, tada se elemenat liste može pisati u obliku

$$((\text{ime}(i_1, i_2),\ i_1 = m_1^{(1)},\ m_2^{(1)}),\ i_2 = m_1^{(2)},\ m_2^{(2)}) \qquad (5.4.3)$$

gde je

ime - ime niza,

i_1, i_2 - imena celobrojnih promenljivih,

$m_j^{(i)}$ - celi neoznačeni brojevi ili imena celobrojnih promenljivih, $i, j = 1, 2$.

Zapis (5.4.3) ima isti efekat kao da su elementi matrice navedeni u sledećem redosledu:

$$\begin{array}{l} \text{ime}(m_1^{(1)}, m_1^{(2)}),\ \text{ime}(m_1^{(1)}+1, m_1^{(2)}), \ldots, \text{ime}(m_2^{(1)}, m_1^{(2)}), \\ \text{ime}(m_1^{(1)}, m_1^{(2)}+1),\ \text{ime}(m_1^{(1)}+1, m_1^{(2)}+1), \ldots, \text{ime}(m_2^{(1)}, m_1^{(2)}+1), \\ \hline \\ \text{ime}(m_1^{(1)}, m_2^{(2)}),\ \text{ime}(m_1^{(1)}+1, m_2^{(2)}), \ldots, \text{ime}(m_2^{(1)}, m_2^{(2)}) \end{array}$$

Ako se u listi navode svi elementi dvodimenzionalnog niza, tada se oblik (5.4.3) svodi na

$$((\text{ime}(i_1, i_2),\ i_1 = 1,\ i_{1max}),\ i_2 = 1,\ i_{2max}) \qquad (5.4.4)$$

Umesto oblika (5.4.4) može se u listi navesti samo ime niza, tj. oblik

$$\text{ime} \qquad (5.4.5)$$

Ovde je važno uočiti da oblik (5.4.4), odnosno (5.4.5) obezbedjuje pojavljivanje elemenata niza u listi u redosledu kolona po kolona matrice. Medjutim, ako se želi redosled vrsta po vrsta matrice tada (5.4.4) treba zapisati u obliku

$$((\text{ime}(i_1, i_2),\ i_2 = 1,\ i_{2max}),\ i_1 = 1,\ i_{1max}) \qquad (5.4.6)$$

Medjutim, zapis (5.4.6) ne može se zameniti zapisom (5.4.5), jer (5.4.5) podrazumeva redosled (5.4.4). Važi opšte pravilo da se u zapisima (5.4.4), odnosno (5.4.6) brže menja indeks prvi sleva, a sporije indeks koji sledi. Tako, u (5.4.4) brže se menja indeks i_1, a sporije indeks i_2, tj. za $i_2 = 1$, indeks i_1 uzima sve vrednosti $i_1 = 1, 2, \ldots, i_{1max}$, a zatim dolazi do promene indeksa i_2, tj. $i_2 = 2$, pri čemu opet indeks i_1 uzima sve moguće vrednosti. Kod zapisa (5.4.6) indeks i_2 se brže menja, a indeks i_1 sporije.

c) Ako elementi niza, koji se navode u listi, ne slede jedan za drugim, može se u listi pisati oblik

$$((\text{ime}(i_1, i_2),\ i_1 = m_1^{(1)},\ m_2^{(1)},\ m_3^{(1)}),\ i_2 = m_1^{(2)}, m_2^{(2)}, m_3^{(2)}) \qquad (5.4.7)$$

gde su sve oznake iste kao u (5.4.3), a dopisane veličine $m_3^{(1)}$ i $m_3^{(2)}$ mogu biti celi neoznačeni brojevi ili celobrojne promenljive. Zapis (5.4.7) ima isti efekat kao da su elementi matrice navedeni u listi u sledećem redosledu:

$$\text{ime}(m_1^{(1)}, m_1^{(2)}), \text{ime}(m_1^{(1)} + m_3^{(1)}, m_1^{(2)}), \ldots, \text{ime}(m_1^{(1)} + k_1 m_3^{(1)}, m_1^{(2)}),$$
$$\text{ime}(m_1^{(1)}, m_1^{(2)} + m_3^{(2)}), \text{ime}(m_1^{(1)} + m_3^{(1)}, m_1^{(2)} + m_3^{(2)}), \ldots, \text{ime}(m_1^{(1)} +$$
$$+ k_1 m_3^{(1)}, m_1^{(2)} + m_3^{(2)}), \ldots, \text{ime}(m_1^{(1)}, m_1^{(2)} + k_2 m_3^{(2)}), \text{ime}(m_1^{(1)} + m_3^{(1)},$$
$$m_1^{(2)} + k_2 m_3^{(2)}), \ldots, \text{ime}(m_1^{(1)} + k_1 m_3^{(1)}, m_1^{(2)} + k_2 m_3^{(2)})$$

gde su

$$k_1 = \left[\frac{m_2^{(1)} - m_1^{(1)}}{m_3^{(1)}} \right] \qquad (5.4.8)$$

$$k_2 = \left[\frac{m_3^{(2)} \quad m_1^{(2)}}{m_3^{(2)}} \right] \qquad (5.4.8)$$

celobrojni delovi odgovarajućih količnika.

Tako, zapis

((A (I, J), I = 2,6,2), J = 4.8,3)

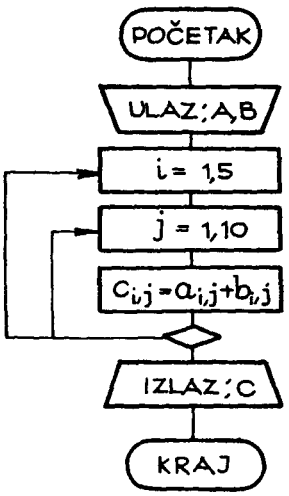
proizvodi sledeći redosled elemenata niza A u listi:

A (2,4), A (4,4), A (6,4), A (2,7), A (4,7), A (6,7)

Primer

Sastaviti program za izračunavanje zbira matrica

C = A + B



Sl. 5.4.1

gde su A i B matrice 5x10. Elementi jedne vrste matrice A, odnosno B, bušeni su na jednoj kartici sa opisom F8.3. Na izlazu štampati rezultujuću matricu C. Šema algoritma prikazana je na sl. 5.4.1, gde su sa A, B i C označene matrice, a sa $a_{i,j}$, $b_{i,j}$ i $c_{i,j}$ elementi odgovarajućih matrica. Program, na FORTRAN-jeziku, sastavljen po algoritmu na sl. 5.4.1 ima izgled koji je dat na sledećoj strani.

Kao što se vidi, program sadrži dva ciklusa jedan u okviru drugog. Spoljašnji ciklus se izvršava 5 puta, a unutrašnji 10 puta. Medjutim, za svaki prolazak kroz spoljašnji

```
      DIMENSION A(5,10),B(5,10),C(5,10)
      READ(5,10)((A(I,J),J=1,10),I=1,5),((B(I,J),J=1,10),I=1,5)
   10 FORMAT(10F8.3)
      DO 11 I=1,5
      DO 11 J=1,10
   11 C(I,J)=A(I,J)+B(I,J)
      WRITE(6,20)((C(I,J),J=1,10),I=1,5)
   20 FORMAT(' MATRICA C'//(' ',10F11.3))
      STOP
      END
```

ciklus unutrašnji ciklus se izvrši 10 puta, tako da naredba 11 (ova naredba je zadnja i zajednička naredba za oba ciklusa) izvrši se pri jednom izvršenju programa 50 puta.

5.4.2. Registrovanje dvodimenzionalnog niza u memoriji računara i veza sa jednodimenzionalnim nizom

Već je rečeno da se nizovi u memoriji računara registruju kolona po kolona u registrima memorije, čije adrese slede u prirodnom nizu brojeva. Posmatrajmo niz a(i,j), gde je a ime niza, i i j su indeksi niza, koji uzimaju sledeće vrednosti i = 1,2,...,n; j = 1,2,...,m. Uvedimo oznaku $a_{i,j}/k$ u kojoj $a_{i,j}$ predstavlja elemenat niza a(i,j), a k relativnu adresu registra u kome se registruje elemenat niza a(i,j). Relativna adresa registra definiše registar u kome se nalazi prvi elemenat niza sa 1, a ostale registre re-

Tabela 5.4.1

i \ j	1	2	---------	m
1	$a_{1,1/1}$	$a_{1,2/n+1}$	--------	$a_{1,m/(m-1)n+1}$
2	$a_{2,1/2}$	$a_{2,2/n+2}$	--------	$a_{2,m/(m-1)n+2}$
⋮	⋮	⋮	⋮	⋮
n	$a_{n,1/n}$	$a_{n,2/2n}$	--------	$a_{n,m/nm}$

dom 2, 3, 4, ..., n.m, gde je n.m relativna adresa zadnjeg elementa niza a(n, m). U tabeli 5.4.1 prikazan je raspored elemenata niza a(i, j) sa odgovarajućim relativnim adresama.

Iz tabele 5.4.1 sledi da se relativna adresa k, elementa sa indeksima i, j izračunava po formuli

$$k = i + n(j-1) \qquad (5.4.10)$$

Prema tome, dvodimenzionalni niz a(i, j) čiji indeksi uzimaju vrednosti i = 1, 2, ..., n; j = 1, 2, ..., m, može se posmatrati u memoriji računara kao jednodimenzionalni niz a(k) čiji indeks uzima vrednosti k = 1, 2, ..., n.m. Veza izmedju indeksa dvodimenzionalnog niza a(i, j) i jednodimenzionalnog niza a(k) data je relacijom (5.4.10).

Primer

Način registrovanja dvodimenzionalnih nizova i njihovu vezu sa jednodimenzionalnim nizovima prosledićemo na primeru sabiranja dvodimenzionalnih matrica. Ovde ćemo razlikovati dva slučaja:

1) Dimenzije matrica u programu su iste sa dimenzijama matrica definisanim opisnom naredbom DIMENSION.

a) Rešenje zadatka preko dvodimenzionalnih nizova.

Na ovaj način rešeno je sabiranje dvodimenzionalnih matrica u pri-

Tabela 5.4.2

i \ j	1	2	3	4	5	6	7	8	9	10
1	$a_{1,1}$ 1	$a_{1,2}$ 6	$a_{1,3}$ 11	$a_{1,4}$ 16	$a_{1,5}$ 21	$a_{1,6}$ 26	$a_{1,7}$ 31	$a_{1,8}$ 36	$a_{1,9}$ 41	$a_{1,10}$ 46
2	$a_{2,1}$ 2	$a_{2,2}$ 7	$a_{2,3}$ 12	$a_{2,4}$ 17	$a_{2,5}$ 22	$a_{2,6}$ 27	$a_{2,7}$ 32	$a_{2,8}$ 37	$a_{2,9}$ 42	$a_{2,10}$ 47
3	$a_{3,1}$ 3	$a_{3,2}$ 8	$a_{3,3}$ 13	$a_{3,4}$ 18	$a_{3,5}$ 23	$a_{3,6}$ 28	$a_{3,7}$ 33	$a_{3,8}$ 38	$a_{3,9}$ 43	$a_{3,10}$ 48
4	$a_{4,1}$ 4	$a_{4,2}$ 9	$a_{4,3}$ 14	$a_{4,4}$ 19	$a_{4,5}$ 24	$a_{4,6}$ 29	$a_{4,7}$ 34	$a_{4,8}$ 39	$a_{4,9}$ 44	$a_{4,10}$ 49
5	$a_{5,1}$ 5	$a_{5,2}$ 10	$a_{5,3}$ 15	$a_{5,4}$ 20	$a_{5,5}$ 25	$a_{5,6}$ 30	$a_{5,7}$ 35	$a_{5,8}$ 40	$a_{5,9}$ 45	$a_{5,10}$ 50

meru na kraju odeljka 5.4.1. U ovom slučaju tabela 5.4.1 za registrovanje elemenata matrice A ima oblik tabele 5.4.2. Na isti način se registruju i elementi matrica B i C.

Kako je u navedenom primeru pretpostavljeno da se elementi matrice A unose vrsta po vrsta, to znači da će elementi prve vrste biti registrovani u registrima čije su relativne adrese 1, 6, 11, 16, 21, 26, 31, 36, 41 i 46. Zatim se unose elementi druge vrste i registruju u registrima 2, 7, 12, itd. U istom programu sabiranje matrica je izvršeno preko dva programska ciklusa, tako da se sabiraju vrsta po vrsta matrica A i B i formira se matrica C.

b) Rešenje zadatka preko jednodimenzionalnih nizova.

Isti zadatak se može rešiti ako se koriste jednodimenzionalni nizovi. U ovom slučaju program na FORTRAN-jeziku ima sledeći izgled:

```
      DIMENSION A(50),B(50),C(50)
      READ(5,10) A,B
10    FORMAT(10F8.3)
      DO 11 I=1,50
11    C(I)=A(I)+B(I)
      WRITE(6,20) C
20    FORMAT(' MATRICA C'//(' ',10F11.3))
      STOP
      END
```

Tabela 5.4.3 prikazuje registrovanje elemenata matrice A, kada su uneti kao jednodimenzionalan niz, a pri tome su na karticama bili raspoređeni vrsta po vrsta.

Tabela 5.4.3

i \ j	1	2	3	4	5	6	7	8	9	10
1	$a_{1,1}$ 1	$a_{1,6}$ 6	$a_{2,1}$ 11	$a_{2,6}$ 16	$a_{3,1}$ 21	$a_{3,6}$ 26	$a_{4,1}$ 31	$a_{4,6}$ 36	$a_{5,1}$ 41	$a_{5,6}$ 46
2	$a_{1,2}$ 2	$a_{1,7}$ 7	$a_{2,2}$ 12	$a_{2,7}$ 17	$a_{3,2}$ 22	$a_{3,7}$ 27	$a_{4,2}$ 32	$a_{4,7}$ 37	$a_{5,2}$ 42	$a_{5,7}$ 47
3	$a_{1,3}$ 3	$a_{1,8}$ 8	$a_{2,3}$ 13	$a_{2,8}$ 18	$a_{3,3}$ 23	$a_{3,8}$ 28	$a_{4,3}$ 33	$a_{4,8}$ 38	$a_{5,3}$ 43	$a_{5,8}$ 48
4	$a_{1,4}$ 4	$a_{1,9}$ 9	$a_{2,4}$ 14	$a_{2,9}$ 19	$a_{3,4}$ 24	$a_{3,9}$ 29	$a_{4,4}$ 34	$a_{4,9}$ 39	$a_{5,4}$ 44	$a_{5,9}$ 49
5	$a_{1,5}$ 5	$a_{1,10}$ 10	$a_{2,5}$ 15	$a_{2,10}$ 20	$a_{3,5}$ 25	$a_{3,10}$ 30	$a_{4,5}$ 35	$a_{4,10}$ 40	$a_{5,5}$ 45	$a_{5,10}$ 50

Poredjenjem tabela 5.4.2 i 5.4.3 vidi se da su to dva sasvim različita načina registrovanja elemenata matrice A, a isto će biti i za matricu B. U tabeli 5.4.3 elementi prve vrste matrice nalaze se u registrima čije su relativne adrese 1,2,3,4,5,6,7,8,9,10, a zatim slede elementi druge vrste itd. Izračunavanje rezultujuće matrice C vrši se preko jednog programskog ciklusa, koji se izvršava 50 puta.

c) Rešenje zadatka pod a) i b) je sprovedeno pod pretpostavkom da su ulazni podaci bili u redosledu vrsta po vrsta matrica A i B. Medjutim, ako pretpostavimo da se na jednoj kartici nalaze elementi jedne kolone matrice A, odnosno B, tada će za registrovanje matrice A biti potrebno 10 kartica, a na svakoj će se nalaziti po jedna kolona matrice. Isto će biti i za matricu B.

Program za sabiranje matrica može se napisati u sledećem obliku:

```
      DIMENSION A(50),B(50),C(50)
      READ(5,10) A,B
   10 FORMAT(5F8.3)
      DO 11 I=1,50
   11 C(I)=A(I)+B(I)
      WRITE(6,20)
   20 FORMAT(' MATRICA C'/)
      DO 12 J=1,5
   12 WRITE(6,21)(C(I),I=J,50,5)
   21 FORMAT(' ',10F11.3)
      STOP
      END
```

Elementi matrica A i B unose se sa kartica kao elementi jednodimenzionalnih nizova. Izračunavanje matrice C takodje je izvršeno preko sabiranja elemenata jednodimenzionalnih nizova. Način registrovanja elemenata jednodimenzionalnog niza A identičan je kao i u slučaju dvodimenzionalnog niza i prikazan je u tabeli 5.4.2. Medjutim, kako rezultujuću matricu C želimo da štampamo u obliku vrsta po vrsta, to se u listi izlazne naredbe matrica C piše kao jednodimenzionalni niz, čiji se svaki peti elemenat štampa u jednom redu, pri čemu je prvi elemenat odredjen indeksom (J) koji se menja u DO ciklusu.

2) Dimenzije matrica u programu su manje od dimenzija matrica definisanih opisnom naredbom DIMENSION. U ovom slučaju opisnom naredbom definišu se najveće moguće dimenzije matrica, a dimenzije matrica

za koje se izvršava program zadaju se preko ulaznih podataka.

a) Rešenje zadatka preko dvodimenzionalnih nizova. Neka su matrice A i B tipa n x m, gde je $n \leq 5$, $m \leq 10$. Ulazni podaci neka su rasporedjeni u redosledu vrsta po vrsta, a na svakoj kartici se nalazi 10 elemenata jedne ili više vrsta. FORTRAN-program u ovom slučaju ima sledeći izgled:

```
   DIMENSION A(5,10),B(5,10),C(5,10)
   READ(5,10) N,M,((A(I,J),J=1,M),I=1,N),((B(I,J),J=1,M),I=1,N)
10 FORMAT(2I2/(10F8.3))
   DO 11 I=1,N
   DO 11 J=1,M
11 C(I,J)=A(I,J)+B(I,J)
   WRITE(6,20)
20 FORMAT(' MATRICA C'/)
   DO 12 I=1,N
12 WRITE(6,30) (C(I,J),J=1,M)
30 FORMAT(' ',10F11.3)
   STOP
   END
```

Prema tome, pre unošenja elemenata matrica A i B, unosi se broj vrsta n i broj kolona m matrica A i B, i ovi brojevi dodeljuju se promenljivim N i M. Način registrovanja matrice A, u slučaju matrice 3x4, prikazan je u tabeli 5.4.4. Kao što se vidi iz tabele, matrica A je registrovana u 12 registara čije su relativne adrese 1, 2, 3, 6, 7, 8, 11, 12, 13, 16, 17 i 18, a svih ostalih 38 registara je slobodno. Važno je uočiti da sumiranje ovako registrovanih matrica ne može da se izvrši preko jednodimenzionalnog niza koji ima 12 elemenata, jer adrese elemenata matrica A i B ne slede jedna za drugom.

b) Rešenje zadatka preko jednodimenzionalnih nizova.

Neka su matrice A i B tipa n x m, i njihovi elementi rasporedjeni na karticama vrsta po vrsta, a na svakoj kartici da se nalazi 10 elemenata jedne ili više vrsta. Tada program na FORTRAN-jeziku može biti zapisan u obliku:

```
      DIMENSION A(50),B(50),C(50)
      READ(5,10) N,M
   10 FORMAT(2I2)
      K=N*M
      READ(5,40) (A(I),I=1,K),(B(I),I=1,K)
   40 FORMAT(10F8.3)
      DO 11 I=1,K
   11 C(I)=A(I)+B(I)
      WRITE(6,20)
```

```
20 FORMAT(' MATRICA C'/)
   DO 12 I=1,K,M
   L=I+M-1
12 WRITE(6,30) (C(J),J=I,L)
30 FORMAT(' ',10F11.3)
   STOP
   END
```

Neka su matrice A i B tipa 3 x 4. Gornji program će izvršiti registrovanje elemenata matrice A u rasporedu prikazanom u tabeli 5.4.5.

Tabela 5.4.4

i \ j	1	2	3	4	5	6	7	8	9	10
1	$a_{1,1}$ 1	$a_{1,2}$ 6	$a_{1,3}$ 11	$a_{1,4}$ 16	21	26	31	36	41	46
2	$a_{2,1}$ 2	$a_{2,2}$ 7	$a_{2,3}$ 12	$a_{2,4}$ 17	22	27	32	37	42	47
3	$a_{3,1}$ 3	$a_{3,2}$ 8	$a_{3,3}$ 13	$a_{3,4}$ 18	23	28	33	38	43	48
4	4	9	14	19	24	29	34	39	44	49
5	5	10	15	20	25	30	35	40	45	50

Tabela 5.4.5

i \ j	1	2	3	4	5	6	7	8	9	10
1	$a_{1,1}$ 1	$a_{2,4}$	$a_{3,3}$	16	21	26	31	36	41	46
2	$a_{1,2}$ 2	$a_{2,3}$ 7	$a_{3,4}$ 12	17	22	27	32	37	42	47
3	$a_{1,3}$ 3	$a_{2,4}$ 8	13	18	23	28	33	38	43	48
4	$a_{1,4}$ 4	$a_{3,1}$ 9	14	19	24	29	34	39	44	49
5	$a_{2,1}$ 5	$a_{3,2}$ 10	15	20	25	30	35	40	45	50

Prema tome, elementi matrice A sada su rasporedjeni u registrima čije adrese slede jedna za drugom. I sabiranje matrica može se izvršiti ako se primenjuje jedan programski ciklus koji se izvršava n x m puta.

c) Ako se elementi matrica A i B registruju kolona po kolona na karticama, i to tako da se na kartici nalazi 10 brojeva tada će program imati sledeći izgled:

```
      DIMENSION A(50),B(50),C(50)
      READ(5,10) N,M
   10 FORMAT(2I2)
      K=N*M
      READ(5,40) (A(I),I=1,K),(B(I),I=1,K)
   40 FORMAT(10F8.3)
      DO 11 I=1,K
   11 C(I)=A(I)+B(I)
      WRITE(6,20)
   20 FORMAT(' MATRICA C'/)
      L=M-1
      DO 12 I=1,N
   12 WRITE(6,30) (C(J),J=I,K,L)
   30 FORMAT(' ',10F11.3)
      STOP
      END
```

U ovom slučaju elementi matrice A, tipa 3 x 4 biće registrovani na način prikazan u tabeli 5.4.6.

Tabela 5.4.6

i \ j	1	2	3	4	5	6	7	8	9	10
1	$a_{1,1}$ 1	$a_{3,2}$ 6	$a_{2,4}$ 11	16	21	26	31	36	41	46
2	$a_{2,1}$ 2	$a_{1,3}$ 7	$a_{3,4}$ 12	17	22	27	32	37	42	47
3	$a_{3,1}$ 3	$a_{2,3}$ 8	13	18	23	28	33	38	43	48
4	$a_{1,2}$ 4	$a_{3,3}$ 9	14	19	24	29	34	39	44	49
5	$a_{2,2}$ 5	$a_{1,4}$ 10	15	20	25	30	35	40	45	50

Kao što se vidi, tabele 5.4.5 i 5.4.6 razlikuju se po tome što u tabeli 5.4.5 elementi slede po vrstama, a u tabeli 5.4.6 po kolonama. Kako rezultujuću matricu treba štampati po vrstama, to se naredbe izlaza u odgovarajućim programima razlikuju.

Navedeni primeri ilustruju različite načine registrovanja dvodimenzionalnih nizova u memoriji računara. Važno je uočiti da unošenje eleme-

nata dvodimenzione matrice kolona po kolona, i unošenje iste matrice kao jednodimenzionog niza ima isti raspored registrovanja elemenata u memoriji, ako je rang matrice jednak maksimalnim vrednostima indeksa u programu. U svim drugim slučajevima ovo registrovanje je različito. Medjutim, obrada nad elementima matrica, predstavlja bolje programsko rešenje, ako se matrica tretira kao jednodimenzionalni niz. Tako u slučaju sabiranja matrica videli smo da tretiranje matrice kao dvodimenzionalnog niza zahteva dva programska ciklusa, a u slučaju jednodimenzionalnog niza zahteva jedan programski ciklus.

5.5. Višedimenzionalni nizovi

Elemenat višedimenzionog niza ima opšti oblik

ime(lista) (5.5.1)

gde je

lista - spisak, od najviše 7, indeksa niza medju sobom razdvojenih zarezima.

Prema tome, višedimenzioni niz može imati najviše 7 indeksa. Sve što je rečeno za dvodimenzionalne nizove važi i za višedimenzionalne nizove. Maksimalne vrednosti indeksa navode se u listi naredbe DIMENSION, pri čemu elemenat liste ima oblik (5.1.7). Vrsta niza može biti definisana jednom od opisnih naredbi za deklarisanje vrste.

Višedimenzionalni niz u listi ulazno-izlaznih naredbi navodi se u obliku:

$$(\ldots(\text{ime}(i_1, i_2, \ldots, i_k), i_1 = m_1^{(1)}, m_2^{(1)}, m_3^{(1)}),$$
$$i_2 = m_1^{(2)}, m_2^{(2)}, m_3^{(2)}), \ldots, i_k = m_1^{(k)}, m_2^{(k)}, m_3^{(k)}) \qquad (5.5.2)$$

Ako je priraštaj indeksa 1, može se priraštaj izostaviti i oblik (5.5.2) svodi se na

$$(\ldots(\text{ime}(i_1, i_2, \ldots, i_k)\ i_1 = m_1^{(1)}, m_2^{(1)}),$$
$$i_2 = m_1^{(2)}, m_2^{(2)}), \ldots, i_k = m_1^{(k)}, m_2^{(k)}) \qquad (5.5.3)$$

Ako se indeksi u (5.5.3) menjaju od 1 do maksimalne vrednosti, tj.

$$(\ldots(\text{ime}(i_1, i_2, \ldots, i_k), i_1=1, i_{1max}), \ldots, i_k=1, i_{kmax}) \quad (5.5.4)$$

tada se mogu izostaviti, i oblik (5.5.4) svodi se samo na ime niza

$$\text{ime} \quad (5.5.5)$$

Kao i kod dvodimenzionalnih nizova, tako i kod višedimenzionalnih nizova najbrže se menja prvi indeks sleva, a zatim sleva na desno sporije se menjaju, tako da krajnji desni indeks se menja najsporije

Tako, elemenat liste u obliku

(((A(I, J, K), I=2, 6, 2), J=1, 2), K=4, 12, 5)

ima isti efekat kao da su navedene indeksne promenljive u sledećem redosledu:

A (2, 1, 4), A (4, 1, 4), A (6, 1, 4), A (2, 2, 4), A (4, 2, 4), A (6, 2, 4),
A (2, 1, 9), A (4, 1, 9), A (6, 1, 9), A (2, 2, 9), A (4, 2, 9), A (6, 2, 9)

Višedimenzionalni niz sa k indeksa $i_1, i_2, \ldots, i_k$, može se posmatrati kao jednodimenzionalni niz sa indeksom j. Veza izmedju indeksa višedimenzionalnog i jednodimenzionalnog niza data je relacijom

$$j = i_1 + (i_2 - 1)i_{1max} + (i_3 - 1)i_{1max} \cdot i_{2max} + \ldots$$
$$\ldots + (i_k - 1)i_{1max} \cdot i_{2max} \cdots i_{(k-1)max} \quad (5.5.6)$$

Relacija (5.5.6) za zadate vrednosti indeksa višedimenzionalnog niza odredjuje indeks odgovarajućeg jednodimenzionalnog niza, odnosno relativnu adresu elementa višedimenzionalnog niza.

Primer

Na jednoj kartici, u kolonama 4, 25, 32 i 48 bušeni su jednocifreni brojevi od 1 do 5. Označimo ove brojeve redom sa i, j, k, l. Sastaviti program koji će u proizvoljnom broju kartica utvrditi broj pojavljivanja m, za-

ke od mogućih kombinacija i, j, k, l. Na izlazu štampati kombinacije i, j, k, l koje se pojavljuju u zadatom paketu kartica, kao i njihov broj pojavljivanja.

```
      DIMENSION M(5,5,5,5)
      DO 10 I=1,5
      DO 10 J=1,5
      DO 10 K=1,5
      DO 10 L=1,5
   10 M(I,J,K,L)=0
   40 READ(5,20,END=30) I,J,K,L
   20 FORMAT(3X,I1,20X,I1,6X,I1,15X,I1)
      M(I,J,K,L)=M(I,J,K,L)+1
      GO TO 40
   30 WRITE(6,90)
   90 FORMAT(' ',4X,'I',4X,'J',4X,'K',4X,'L',4X,'M'/)
      DO 50 I=1,5
      DO 50 J=1,5
      DO 50 K=1,5
      DO 50 L=1,5
      IF(M(I,J,K,L)) 60,50,60
   60 WRITE(6,80) I,J,K,L,M(I,J,K,L)
   50 CONTINUE
   80 FORMAT(' ',5I5)
      STOP
      END
```

Naredba sa obeležjem 10, koja se izvršava 625 puta, vrši postavljanje nule, kao brojne vrednosti svih elemenata četvorodimenzionalnog niza M. To postavljanje vrši se na taj način što se redom menjaju indeksi niza M. U obradi svaka kombinacija I, J, K, L definiše jedan elemenat matrice M, i pojava ove kombinacije povećava vrednost odgovarajućeg elementa matrice M za jedinicu. Na ovaj način izvršeno je prebrojavanje svih kombinacija na ulaznim karticama.

Štampanje je programirano tako da se u jednom redu štampa kombinacija i broj pojavljivanja. Kombinacije koje se ne pojavljuju, neće biti štampane. Za jedan primer ulaznih kartica izlazna tabela ima sledeći izgled:

I	J	K	L	M
1	1	3	3	4
2	2	5	4	1
3	3	1	5	2
4	2	1	5	1
4	2	5	1	3
4	5	1	1	1
5	1	2	4	1
5	1	3	3	1

5.6. Redosled elemenata dva niza ili više nizova u listi ulazno-izlaznih naredbi

Ako se elementi nizova navode kao indeksne promenljive sa konkretnim vrednostima indeksa, njihov redosled može biti proizvoljan, ali usaglašen sa ulaznim podacima, odnosno sa oblikom štampanja na izlazu. Medjutim, ako elementi jednog niza slede u odredjenom redosledu, onda se oni mogu skraćeno pisati na način kako je to objašnjeno u prethodnim odeljcima (vidi 5.2.1 i 5.4.1).

Često postoji potreba da elementi dva niza ili više nizova slede naizmenično jedan iza drugog. U ovom slučaju element liste može imati sledeći oblik:

$$(ime_1(i),\ ime_2(i),\ i = m_1,\ m_2,\ m_3) \qquad (5.6.1)$$

gde elementi niza ime_1 i ime_2 slede naizmenično jedan iza drugog. Zapis (5.6.1) proizvodi sledeći redosled elemenata.

$$ime_1(m_1),\ ime_2(m_1),\ ime_1(m_1+m_3),\ ime_2(m_1+m_3),\ \ldots\ \ldots$$
$$\ldots\ \ldots,\ ime_1(m_1+km_3),\ ime_2(m_1+km_3)$$

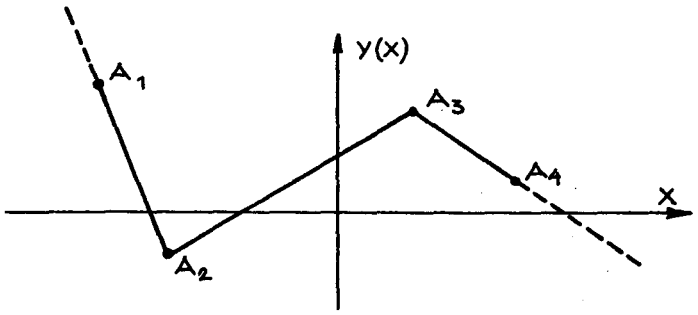
gde je k odredjeno sa (5.2.10).

Oblik (5.6.1) koji definiše dva jednodimenzionalna niza u naizmeničnom redosledu elemenata, može se proširiti na veći broj jednodimenzionalnih nizova.

Takodje, umesto jednodimenzionalnih nizova mogu se u naizmeničnom redosledu elemenata pisati višedimenzionalni nizovi.

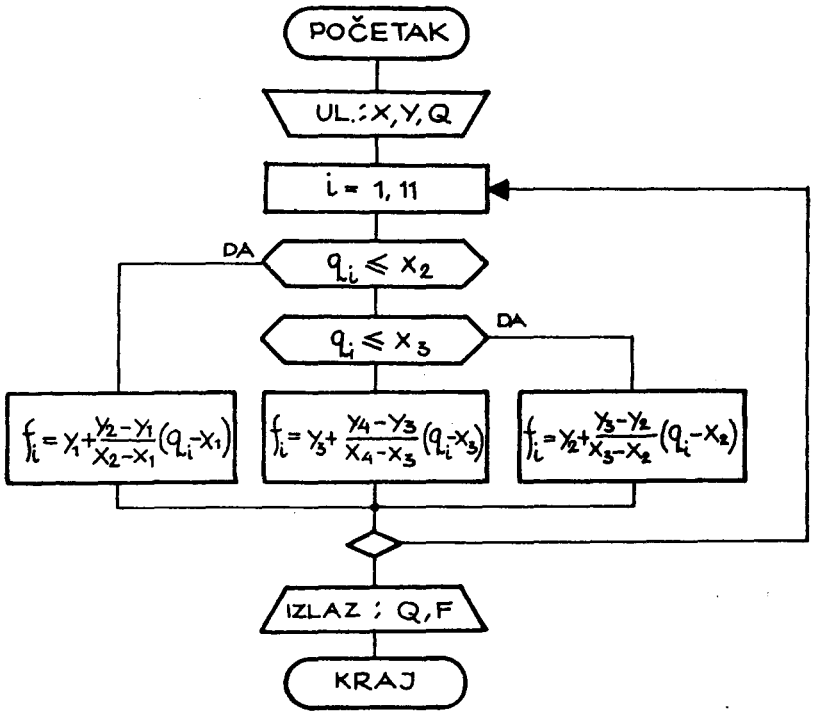
Primer

Koordinate tačaka A_i, $i = 1,2,3,4$ sa sl. 5.6.1, zadate su na jednoj kartici u sledećem redosledu $x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4$ i to tako da svaka koordinata zauzima 10 kolona, od kojih su tri namenjene za decimalna mesta. Iza ove kartice nalaze se dve kartice i to tako da se na prvoj nalazi 8 vrednosti, a na drugoj tri vrednosti za argumenat x, pri čemu argumenat može imati tri cela i tri decimalna mesta. Za zadatih 11 vrednosti argumenta x izračunati y(x), prema sl. 5.6.1, i štampati x i y u obliku tabele.



Sl. 5.6.1

Blok šema algoritma prikazana je na sl. 5.6.2. Sa X i Y označeni su nizovi čiji su elementi apscise, odnosno koordinate tačaka $A_i(x_i, y_i)$. U



Sl. 5.6.2

istom algoritmu Q označava niz čiji su elementi q_i, $i = 1,2,\ldots,11$ zadate vrednosti argumenta x. Vrednosti funkcije f_i, $i = 1,2,\ldots,11$ označene su nizom F. Kao što se vidi sa sl. 5.6.1. vrednosti funkcije izmedju tačaka odredjuju se linearnom interpolacijom, a vrednosti funkcije levo od tačke A_1 i desno od tačke A_4 odredjuju se linearnom ekstrapolacijom. Program na FORTRAN-jeziku sastavljen po algoritmu na sl. 5.6.2 ima sledeći izgled:

```
      DIMENSION X(4),Y(4),Q(11),F(11)
      READ(5,10) (X(I),Y(I),I=1,4),Q
   10 FORMAT(8F10.3)
      DO 11 I=1,11
      IF(Q(I)-X(2)) 12,12,13
   13 IF(Q(I)-X(3)) 14,14,15
   15 F(I)=Y(3)+(Y(4)-Y(3))/(X(4)-X(3))*(Q(I)-X(3))
   11 CONTINUE
      WRITE(6,16) (Q(I),F(I),I=1,11)
   16 FORMAT(' ',5X,'X',17X,'Y'//(' ',F10.3,E20.7))
      STOP
   12 F(I)=Y(1)+(Y(2)-Y(1))/(X(2)-X(1))*(Q(I)-X(1))
      GO TO 11
   14 F(I)=Y(2)+(Y(3)-Y(2))/(X(3)-X(2))*(Q(I)-X(2))
      GO TO 11
      END
```

U listi ulazno-izlaznih naredbi pojavljuju se nizovi čiji elementi slede u naizmeničnom redosledu.

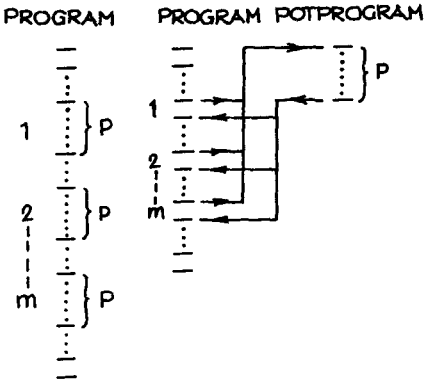
Za zadate koordinate tačaka A_i, tako da je $A_1(-80;45)$, $A_2(-40;-20)$, $A_3(10;100)$, $A_4(70;10)$ i 11 zadatih vrednosti argumenta x rezultati se dobijaju u obliku tabele:

X	Y
-10.000	0.5199998E 02
100.000	-0.3500000E 02
12.000	0.9700000E 02
60.000	0.2500000E 02
222.000	-0.2180000E 03
1.000	0.7839998E 02
13.000	0.9550000E 02
-4.000	0.6639998E 02
10.000	0.9999997E 02
20.000	0.8500000E 02
50.000	0.4000000E 02

6. POTPROGRAMI

6.1. Osnovni pojmovi

Više naredbi izdvojenih u posebnu programsku celinu grade potprogram. Potprogram najčešće predstavlja niz naredbi koje bi se pojavljivale na više mesta jednog programa i kojima bi se vršilo izračunavanje po istim formulama, ali za različite vrednosti argumenata. Izdvajanje ovih naredbi u posebnu programsku celinu omogućuje kraći zapis programa, a samim tim i njegovo lakše prenošenje na računar i manje angažovanje memorije računara. Tako ako se niz od n naredbi označen sa P (sl. 6. 1. 1.) u programu pojavljuje m puta, tada se ovaj niz može izdvojiti u posebnu programsku celinu - potprogram (sl. 6. 1. 2).



Sl. 6. 1. 1

Sl. 6. 1. 2

Program na sl. 6. 1. 1, pored ostalih naredbi, sadrži m. n naredbi, jer se niz P od n naredbi ponavlja m puta. Ako se niz P od n naredbi izdvoji u potprogram (sl. 6. 1. 2), tada se u programu na mestima gde se nalazio niz P vrši prelazak iz programa u potprogram. Po izvršenom potprogramu vrši se povratak u program i to neposredno na sledeću naredbu koja sledi iza mesta prelaza na potprogram. Na ovaj način umesto m. n naredbi piše se samo n naredbi koje čine potprogram. Prema tome, korišćenje potprograma u programiranju ima sledeća svojstva:

- pruža mogućnost kraćeg zapisa programa, a samim tim smanjuje

mogućnost greške u pripremi programa,

- smanjuje angažovanje memoriskog prostora,
- omogućuje lakše testiranje programa, jer se potprogrami kao posebne programske celine mogu odvojeno testirati, i
- isti potprogram može se koristiti u raznim programima.

Ovde treba napomenuti da korišćenje potprograma uopšte ne utiče na brže izvršavanje programa od strane računara. Jer, i ako je umesto m.n naredbi zapisano samo n naredbi, kada se izvršava program, izvršiće se m.n naredbi.

Iz svega što je do sada rečeno o potprogramima sledi da se za korišćenje potprograma u programiranju moraju poznavati sledeći elementi:

- način zapisa potprograma, tako da on čini posebnu programsku celinu,
- način prelaska iz programa u potprogram, i
- način povratka iz potprograma u program.

Izračunavanje elementarnih funkcija (odeljak 4.9) vrši se pomoću potprograma. Za izračunavanje elementarne funkcije, kao što je trigonometrijska funkcija sinus, potrebno je oko 100 naredbi na mašinskom jeziku. Svaka naredba angažuje jedan memorijski registar. Ako se izračunavanje sinusne funkcije vrši na 10 mesta u programu, i ako se ovo izračunavanje ne bi vršilo preko potprograma, to bi značilo da bi 10 puta po 100 naredbi bilo zapisano u programu na mašinskom jeziku, odnosno za ovo izračunavanje bilo bi angažovano 10.100=1000 registara u memoriji. Medjutim, ako se izračunavanje vrši preko potprograma, biće angažovano samo 100 registara u memoriji računara.

Korišćenje potprograma pruža mogućnost da se jedanput izradjen potprogram može dati na korišćenje širokom krugu programera, koji ga mogu lako koristiti u različitim programima. Na ovaj način formira se biblioteka potprograma u računskim centrima, u kojoj se nalaze kao gotovi potprogrami mnogi postupci iz numeričke matematike, statistike i drugih oblasti primene računara.

U FORTRAN-jeziku postoje tri vrste potprograma

- funkcijska naredba,
- funkcijski potprogram i
- opšti potprogram.

6.2. Funkcijska naredba

Funkcijska naredba omogućuje izdvajanje jednog aritmetičkog izraza kao potprograma. Opšti oblik pisanja funkcijske naredbe je

$$\text{ime(lista)} = \psi \qquad (6.2.1)$$

gde je

ime - naziv funkcijske naredbe i definiše se na isti način kao i ime promenljive,

lista - spisak fiktivnih argumenta medju sobom razdvojenih zarezima. Fiktivni argumenti mogu biti samo imena promenljivih.

ψ - aritmetički izraz, u kojem se kao argumenti mogu javiti: fiktivni argumenti, imena promenljivih iz programa, konstante, druga imena funkcijskih naredbi i funkcijskih potprograma (vidi 6.3).

Funkcijska naredba (6.2.1) poziva se na taj način što se kao argument aritmetičkog izraza u programu navodi ime funkcijske naredbe sa stvarnim argumentima izmedju zagrada, tj.

$$\text{ime (lista)} \qquad (6.2.2)$$

gde je

ime - naziv funkcijske naredbe,

lista - spisak stvarnih argumenata, medju sobom razvojenih zarezima.

Kada se izvršava program, svaki argument aritmetičkog izraza u programu, oblika (6.2.2) izračunava se tako što se u funkcijskoj naredbi odgovarajućeg imena (6.2.1) fiktivni argumenti redom zamenjuju stvarnim argumentima. Za ovako definisane vrednosti fiktivnih argumenata izračunava se vrednost aritmetičkog izraza ψ i tako dobijena brojna vrednost dodeljuje se imenu funkcijske naredbe.

Prema tome, funkcijska naredba predstavlja potprogram, sa proizvoljnim brojem ulaznih veličina. Izvestan broj ulaznih veličina se navode u listi (6.2.1) kao argumenti potprograma, a ostale ulazne veličine predstavljaju promenljive koje su definisane u programu, kojem je pridružena funkcijska naredba, a figurišu kao argumenti u aritmetičkom izrazu ψ

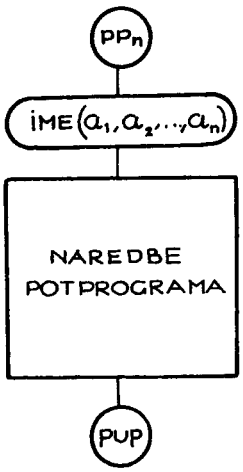
(6.2.1.). Funkcijska naredba mora imati najmanje jedan fiktivni argument. Izlazni rezultat potprograma je jedan broj koji se dodeljuje imenu funkcijske naredbe.

Na mestima stvarnih argumenata u (6.2.2) mogu se pisati aritmetički izrazi.

Fiktivni i stvarni argumenti moraju se slagati po broju, redu i vrsti. To znači ako fiktivnih argumenata ima n i stvarnih mora biti n, i pri tome stvarni argumenti u (6.2.2) zamenjuju fiktivne argumente u (6.2.1) redom sleva nadesno. Slaganje argumenata po vrsti podrazumeva da ako je fiktivni argument celobrojna, odnosno realna promenljiva, onda odgovarajuća veličina mora biti na mestu stvarnog argumenta.

Vrsta funkcijske naredbe deklariše se na isti način kao i vrsta promentljive

- unutrašnjom konvencijom po početnom slovu imena funkcijske naredbe,
- eksplicitnom deklaracijom pomoću opisne naredbe REAL ili INTEGER, ili
- implicitnom deklaracijom pomoću opisne naredbe IMPLICIT.



SL.6.2.1

Funkcijske naredbe se navode na početku programske jedinice pre prve izvršne naredbe programa.

Grafički potprogram se prikazuje pomoću:

- kružnog simbola, u koji se upisuje broj potprograma (PPn, n = 1,2,...), i označava početak potprograma (sl. 6.2.1),
- iza kružnog simbola sledi grafički simbol sa polukružnim bočnim stranama u koji se upisuju ime potprograma i fiktivni argumenti potprograma, koji se po broju, redu i vrsti moraju slagati sa stvarnim argumentima. Ime programa može se izostaviti i u tom slučaju algoritam potprograma se razlikuje od ostalih potprograma oznakom u kružnom simbolu na početku potprograma.

- povratak iz potprograma u program označava se kružnim simbolom u koji se upisuje skraćenica PUP (Povratak U Program).

Primer

Za zadate vrednosti x_i , y_i , i = 1,2,3 izračunati

$$z = \frac{F(x_1 , y_1)}{F(x_2 , y_2)} F(x_3 , y_3)$$

gde je

$$F(x, y) = 3x^2 + 8y + e^{3x^2+8y}$$

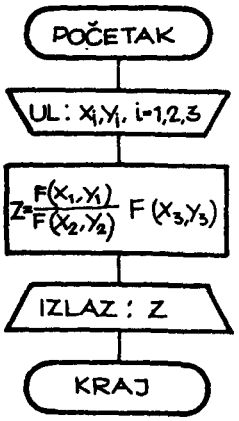
Algoritam za ovo izračunavanje prikazan je na sl. 6.2.2. U ovom algoritmu koristi se potprogram PP1 (sl. 6.2.3) za izračunavanje funkcije

$$F(x, y) = F_1(x, y) + e^{F_1(x, y)}$$

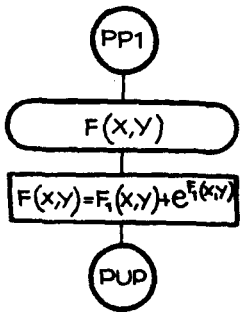
gde je

$$F_1(x, y) = 3x^2 + 8y$$

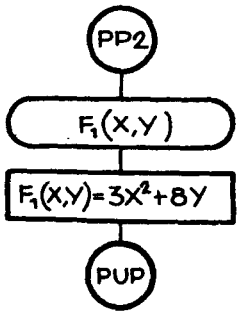
a vrednost funkcije $F_1(x, y)$ izračunava se pomoću potprograma PP2 (sl. 6.2.4) koji se poziva u potprogramu PP1.



Sl. 6.2.2



Sl. 6.2.3



Sl. 6.2.4

Program na FORTRAN-jeziku sastavljen po algoritmima na sl. 6.2.2, sl. 6.2.3 i sl. 6.2.4 ima sledeći izgled:

```
      DIMENSION X(3),Y(3)
      F1(X,Y)=3.*X*X+8.*Y
      F(X,Y)=F1(X,Y)+EXP(F1(X,Y))
      READ(5,70)(X(I),Y(I),I=1,3)
   70 FORMAT(6F6.2)
      Z=F(X(1),Y(1))/F(X(2),Y(2))*F(X(3),Y(3))
      WRITE(6,80) Z
   80 FORMAT(' ','Z=',E14.7)
      STOP
      END
```

U ovom programu je predvidjeno da se ulazni podaci nalaze na jednoj kartici u redosledu x_1, y_1, x_2, y_2, x_3, y_3, a svaki od njih se opisuje sa F6.2.

Za ulazne podatke

$x_1 = 1,00$

$y_1 = 0$

$x_2 = -0,14$

$y_2 = -1,05$

$x_3 = 0,35$

$y_3 = -0,75$

rezultat se dobija u obliku

```
Z= 0.1557935E 02
```

6.3. Funkcijski potprogram

Funkcijska naredba se može koristiti kao potprogram, ako se radi o jednom aritmetičkom izrazu. Medjutim, vrlo često potprogram sadrži veći broj naredbi i to ne samo aritmetičkih. Ovakav potprogram zove se funkcijski potprogram. Opšti oblik ovog potprograma je sledeći

```
FUNCTION ime (lista)
  —
  —                                                    (6.3.1)
  .
  .
  —
END
```

gde je

FUNCTION - službena reč, koja označava početak i tip potprograma,

ime - naziv potprograma koji se definiše na isti način kao ime promenljive,

lista - spisak fiktivnih argumenata potprograma, medju sobom razdvojenih zarezima,

END - službena reč, koja označava fizički kraj potprograma.

Fiktivni argumenti mogu biti imena promenljivih, imena nizova ili fiktivna imena drugih funkcijskih ili opštih potprograma. Kako se fiktivna imena drugih potprograma mogu koristiti kao fiktivni argumenti u listi (6.3.1) biće objašnjeno u odeljku 6.7.

Funkcijski potprogram se piše kao posebna programska celina. Službena reč FUNCTION označava početak ovog potprograma, a službena reč END kraj potprograma, i piše se uvek kao zadnja naredbá potprograma. Izmedju prve i zadnje naredbe potprograma može se nalaziti proizvoljan broj FORTRAN-naredbi, osim

- druge FUNCTION - naredbe,
- druge END - naredbe, ili
- SUBROUTINE - naredbe.

Vrsta fiktivnih argumenata odredjena je unutrašnjom konvencijom FOTRAN-jezika ili opisnim naredbama za eksplicitnu, odnosno implicitnu deklaraciju vrste, koje se pišu iza prve naredbe potprograma.

Prelazak iz programa u ovaj potprogram vrši se na isti način kao i kod funkcijske naredbe, navodjenjem imena potprograma, kao argumenta aritmetičkog izraza u obliku

ime(lista) (6.3.2)

gde je lista spisak stvarnih argumenata, medju sobom razdvojenih zarezima, kojima se zamenjuju redom fiktivni argumenti potprograma. Na mestima stvarnih argumenata mogu doći i aritmetički izrazi ili imena funkcijskih ili opštih potprograma. Stvarni i fiktivni argumenti moraju se slagati po broju, redu i vrsti. Funkcijski potprogram mora imati najmanje jedan argument. Izlazni rezultat funkcijskog potprograma je jedan broj koji se dodeljuje imenu potprograma. Prema tome, medju FORTRAN-naredbama ko-

je čine potprogram mora se nalaziti najmanje jedna aritmetička naredba, na čijoj levoj strani od znaka jednakosti stoji ime potprograma, a kojom se dodeljuje izlazni rezultat potprograma njegovom imenu.

Povratak iz potprograma u program vrši se posebnom FORTRAN-naredbom

RETURN (6.3.3)

koja se mora pojaviti najmanje jedanput izmedju prve i zadnje naredbe potprograma. Ako se u potprogramu koriste programski ciklusi onda naredba (6.3.3) ne sme biti zadnja naredba ciklusa, na sličan način kao što naredba STOP ne sme biti zadnja naredba ciklusa u programu.

U FORTRAN-jeziku postoji veliki broj funkcijskih potprograma, koji se u programu pozivaju propisanim imenom. Ovo su potprogrami opšteg karaktera, kao što je izračunavanje elementarnih funkcija i sl., koji se često koriste u raznim proračunima. U tabeli 6.3.1 dat je spisak ovih potprograma. U tabeli su uvedene sledeće oznake

$x, x_1, x_2, \ldots$ - artimetički izrazi,

R - realna veličina, koja se registruje u obliku pokretnog zareza,

C - celobrojna veličina, koja se registruje u obliku celog broja,

M - maksimalna vrednost celog broja (M=2 147 483 647),

P - maksimalna vrednost broja registrovanog u obliku pokretnog zareza ($P \approx 7,2.10^{75}$),

$[y]$ - celobrojni deo broja y.

Navedena relativna greška funkcije je najveća statistički dobijena relativna grška za razne vrednosti argumenta iz dozvoljenog intervala.

Funkcijski potprogrami navedeni u tabeli 6.3.1, pojavljuju se jedanput u programu posle prevodjenja sa FORTRAN-jezika na mašinski jezik. Svako mesto u FORTRAN-programu na kojem se pojavljuje ime funkcijskog potprograma znači prelaz na ovaj potprogram, a zatim, kada se izvrši ovaj potprogram, vrši se povratak iz potprograma u program.

U FORTRAN-jeziku postoje i funkcije koje se pišu na isti način kao i funkcijski potprogrami, a pojavljuju se u programu onoliko puta koliko

Tabela 6.3.1

Način pisanja		Argumenti		Funkcija		Opis
U Fortranu	U matematici	Vrsta	Ograničenje	Vrsta	Relativna greška je manja od	
EXP(x)	e^x	R	$x<174,673$	R	$4,69\cdot10^{-7}$	Eksponencijalna funkcija
ALOG(x)	ln(x)	R	$x>0$	R	$8,32\cdot10^{-7}$	Prirodni logaritam
ALOG10(x)	log(x)	R	$x>0$	R	$1,05\cdot10^{-6}$	Dekadni logaritam
SQRT(x)	$\sqrt{x}$	R	$x\geq0$	R	$8,70\cdot10^{-7}$	Kvadratni koren
SIN(x)	sin(x)	R	$\|x\|<8,235\cdot10^6$ (u radijanima)	R	$1,59\cdot10^{-6}$	Trigonometrijske funkcije
COS(x)	cos(x)	R	$\|x\|<8,235\cdot10^6$ (u radijanima)	R	$1,59\cdot10^{-6}$	
TAN(x)	tg(x)	R	$\|x\|<8,235\cdot10^6$ $\|x\|\neq(k+1/2)\pi$ $k=0,1,2,\ldots$	R	$6,58\cdot10^{-5}$	
COTAN(x)	ctg(x)	R	$\|x\|<8,235\cdot10^6$ $\|x\|\neq k\pi$ $k=0,1,2,\ldots$	R	$6,58\cdot10^{-5}$	
ARSIN(x)	arcsin(x)	R	$\|x\|\leq1$	R	$8,56\cdot10^{-7}$	Inverzne trigonometrijske funkcije
ARCOS(x)	arccos(x)	R	$\|x\|\leq1$	R	$1,80\cdot10^{-7}$	
ATAN(x)	arctg(x)	R	$\|x\|\leq P$	R	$9,75\cdot10^{-7}$	
ATAN2(x_1,x_2)	arctg(x_1/x_2)	R	$\|x_1\|,\|x_2\|\leq P$ osim $x_1=x_2=0$	R	$9,75\cdot10^{-7}$	
SINH(x)	sinh(x)	R	$\|x\|<174,673$	R	$1,20\cdot10^{-6}$	Hiperboličke funkcije
COSH(x)	cosh(x)	R	$\|x\|<174,673$	R	$0,149\cdot10^{-7}$	
TANH(x)	tgh(x)	R	$\|x\|\leq P$	R	$8,12\cdot10^{-7}$	
AMAX0(x_1,x_2,...)	max(x_1,x_2,...)	C	$\|x_1\|,\|x_2\|,\ldots\leq M$	R	-	Nalaženje najveće vrednosti
AMAX1(x_1,x_2,...)		R	$\|x_1\|,\|x_2\|,\ldots\leq P$	R	-	
MAX0(x_1,x_2,...)		C	$\|x_1\|,\|x_2\|,\ldots\leq M$	C	-	
MAX1(x_1,x_2,...)		R	$\|x_1\|,\|x_2\|,\ldots\leq P$	C	-	
AMIN0(x_1,x_2,...)	min(x_1,x_2,...)	C	$\|x_1\|,\|x_2\|,\ldots\leq M$	R	-	Nalaženje najmanje vrednosti
AMIN1(x_1,x_2,...)		R	$\|x_1\|,\|x_2\|,\ldots\leq P$	R	-	
MIN0(x_1,x_2,...)		C	$\|x_1\|,\|x_2\|,\ldots\leq M$	C	-	
MIN1(x_1,x_2,...)		R	$\|x_1\|,\|x_2\|,\ldots\leq P$	C	-	
ERF(x)	$\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$	R	$\|x\|\leq P$	R	$9,26\cdot10^{-7}$	Funkcija greške
ERFC(x)	$1-\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$	R	$\|x\|\leq P$	R	$9,26\cdot10^{-7}$	Komplement funkcije greške
GAMMA(x)	$\int_0^\infty e^{-t}t^{x-1}dt$	R	$0,13\cdot10^{-75}<x<57,5744$	R	$4,36\cdot10^{-5}$	Gama-funkcija
ALGAMA(x)	$\ln\int_0^\infty e^{-t}t^{x-1}dt$	R	$0<x<4,2913\cdot10^{73}$	R	$1,25\cdot10^{-6}$	Logaritam gama-funkcije

Tabela 6.3.2.

Način pisanja		Argumenti		Funkcija		Opis
U Fortranu	U matematici	Vrsta	Ograničenje	Vrsta	Rel.greška (ε)	
FLOAT(x)	-	C	$\|x\| \leq M$	R	-	Prevodjenje iz oblika celog broja u oblik pokretnog zareza i obratno
IFIX(x)	-	R	$\|x\| < M$	C	-	
ISIGN(x_1,x_2)	$\|x_1\| \operatorname{sign} x_2$	C	$\|x_1\| , \|x_2\| \leq M$	C	-	Algebarski znak od x_2 dodeljuje se $\|x_1\|$
SIGN(x_1,x_2)	$\|x_1\| \operatorname{sign} x_2$	R	$\|x_1\| , \|x_2\| \leq P$	R	-	
IDIM(x_1,x_2)	$x_1 - \min(x_1, x_2)$	C	$\|x_1\| , \|x_2\| \leq M$	C	-	Pozitivna razlika
DIM(x_1,x_2)	$x_1 - \min(x_1, x_2)$	R	$\|x_1\| , \|x_2\| \leq P$	R	-	
MOD(x_1,x_2)	$x_1 (\mathrm{mod} x_2)$	C	$\|x_1 (\mathrm{mod} x_2)\| \leq M$	C	-	Modularna aritmetika
AMOD(x_1,x_2)	$x_1 - \left[\frac{x_1}{x_2}\right] \times 2$	R	$\left\|x_1 - \left[\frac{x_1}{x_2}\right] \times 2\right\| < P$	R	-	
IABS(x)	$\|x\|$	C	$\|x\| \leq M$	C	-	Apsolutne vrednosti
ABS(x)	$\|x\|$	R	$\|x\| \leq P$	R	-	
INT(x)	$[x]$	R	$\|x\| \leq M$	C	-	Odbacivanje decimalnih mesta broja
AINT(x)	$[x]$	R	$\|x\| \leq P$	R	-	

puta su zapisani. Dakle, na svakom mestu gde se nalazi njihovo ime, u mašinskom programu, biće postavljen izvestan broj mašinskih naredbi koje realizuju odgovarajuću funkciju. Ove funkcije su prikazane u tabeli 6.3.2.

Primer

Sastaviti program koji izračunava vrednost funkcije

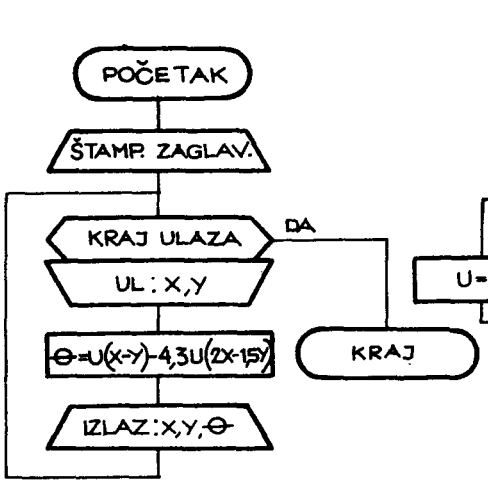
$$\theta(x,y) = U(x-y) - 4,3\ U\ (2x-1,5y) \qquad (6.3.4)$$

gde je

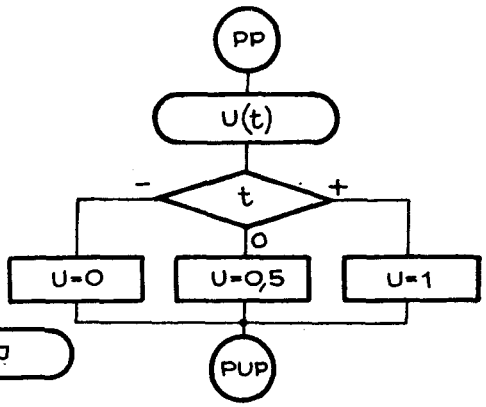
$$U(t) = \begin{cases} 0 & \text{za } t < 0 \\ 0,5 & \text{za } t = 0 \\ 1,5 & \text{za } t > 0 \end{cases} \qquad (6.3.5)$$

za proizvoljan broj parova (x,y). Svaki par brojeva (x,y) nalazi se na po jednoj kartici, i registruje se opisom polja F10.4.

Algoritam je prikazan na sl. 6.3.1, gde je pretpostavljeno da se izračunavanje funkcije (6.3.5) vrši pomoću funkcijskog potprograma (sl.6.3.2).



Sl. 6.3.1



Sl. 6.3.2

FORTRAN-program sastavljen po algoritmu na sl. 6.3.1 ima sledeći izgled:

```
      WRITE(6,10)
10    FORMAT(' ',5X,'X',14X,'Y',9X,'TETA'/)
14    READ(5,12,END=20) X,Y
12    FORMAT(2F10.4)
      TETA=UFUN(X-Y)-4.3*UFUN(2.*X-1.5*Y)
      WRITE(6,13) X,Y,TETA
13    FORMAT(' ',F10.4,5X,F10.4,5X,F4.1)
      GO TO 14
20    STOP
      END
```

Izračunavanje funkcije U(t) vrši se pomoću sledećeg funkcijskog potprograma

```
      FUNCTION UFUN(T)
      IF(T) 10,11,12
10    UFUN=0
      RETURN
11    UFUN=0.5
      RETURN
12    UFUN=1.0
      RETURN
      END
```

Za šest zadatih parova (x, y) na ulazu, izlazni rezultati se dobijaju u obliku tabele

X	Y	TETA
15.0750	-10.0000	-3.3
0.0033	480.0000	0.0
-150.0000	30.5000	0.0
-12.5500	-20.0000	-3.3
0.0	0.0	-1.6
50.0000	70.0000	0.0

6.3.1. Eksplicitna deklaracija vrste funkcijskog potprograma

Vrsta funkcijskog potprograma može se deklarisati unutrašnjom konvencijom FORTRAN-jezika ili implicitnom deklaracijom po prvom slovu imena potprograma. Pored ovih mogućnosti, funkcijski potprogram može se po vrsti deklarisati eksplicitno na sledeći način

```
vrsta FUNCTION ime(lista)                    (6.3.6)
  =
  !
  !
  !
END
```

gde namesto reči vrsta može doći službena reč INTEGER ili REAL u zavisnosti od toga koja se vrsta brojnog podatka dodeljuje imenu promenljive kao rezultat potprograma. Sve ostale reči u gore navedenoj konstrukciji imaju ranije opisano značenje (vidi 6.3.1).

6.4. Opšti potprogram

Funkcijska naredba i funkcijski potprogram kao izlaznu veličinu daju jednu vrednost koja se dodeljuje imenu odgovarajućeg potprograma. Medjutim, vrlo često se zahteva da potprogram može dati više vrednosti na izlazu. To je omogućeno opštim potprogramom, koji se piše kao posebna programska celina oblika

```
SUBROUTINE ime(lista)                        (6.4.1)
    =
    !
    !
    !
END
```

gde je

SUBROUTINE - službena reč i označava početak i tip potprograma,

ime - naziv potprograma i definiše se na isti način kao ime promenljive,

lista - spisak fiktivnih argumenata, medju sobom razdvojenih zarezima,

END - službena reč, koja označava fizički kraj potprograma.

Fiktivni argumenti opšteg potprograma mogu biti imena promenljivih, imena nizova, fiktivna imena drugih funkcijskih ili opštih potprograma (vidi odeljak 6.7).

Izmedju prve naredbe potprograma (SUBROUTINE) i zadnje naredbe (END) može se nalaziti prozvoljan broj FORTRAN-naredbi, osim:

- naredbe FUNCTION,

- druge END-naredbe, ili

- druge SUBROUTINE-naredbe.

Vrsta fiktivnih argumenata odredjena je unutrašnjom konvencijom FORTRAN-jezika ili opisnim naredbama za eksplicitnu, odnosno, implicitnu deklaraciju vrste, koje se pišu iza prve naredbe potprograma.

Kako opšti potprogram može dati više vrednosti na izlazu, to se od svih navedenih argumenata u listi neki pojavljuju kao ulazne veličine u potprogramu, a neki dobijaju brojne vrednosti u potprogramu i predstavljaju izlazne veličine. Prema tome, ime potprograma u ovom slučaju služi samo za raspoznavanje potprograma, a ne za dodeljivanje brojne vrednosti na izlazu iz potprograma. Opšti potprogram ima smisla i bez argumenta. Ako je potprogram bez argumenata, onda se piše u obliku

```
SUBROUTINE   ime                                    (6.4.2)
  =
  |
  |
  |
END
```

Prelazak iz programa u opšti potprogram vrši se posebnom naredbom

```
CALL  ime (lista)                                   (6.4.3)
```

gde je

CALL - službena reč i označava poziv opšteg potprograma,

ime - naziv opšteg potprograma koji se poziva,

lista - spisak stvarnih argumenata, medju sobom razdvojenih zarezima.

Stvarni argumenti mogu biti aritmetički izrazi, imena funkcijskih ili imena opštih potprograma.

Stvarni i fiktivni argumenti moraju se slagati po broju, redu i vrsti. Od navedenih stvarnih argumenata neki će biti ulazne veličine u opštem potprogramu, a neki će biti imena promenljivih ili nizova kojima se dodeljuju vrednosti u potprogramu i predstavljaju izlazne veličine potprograma.

Primer

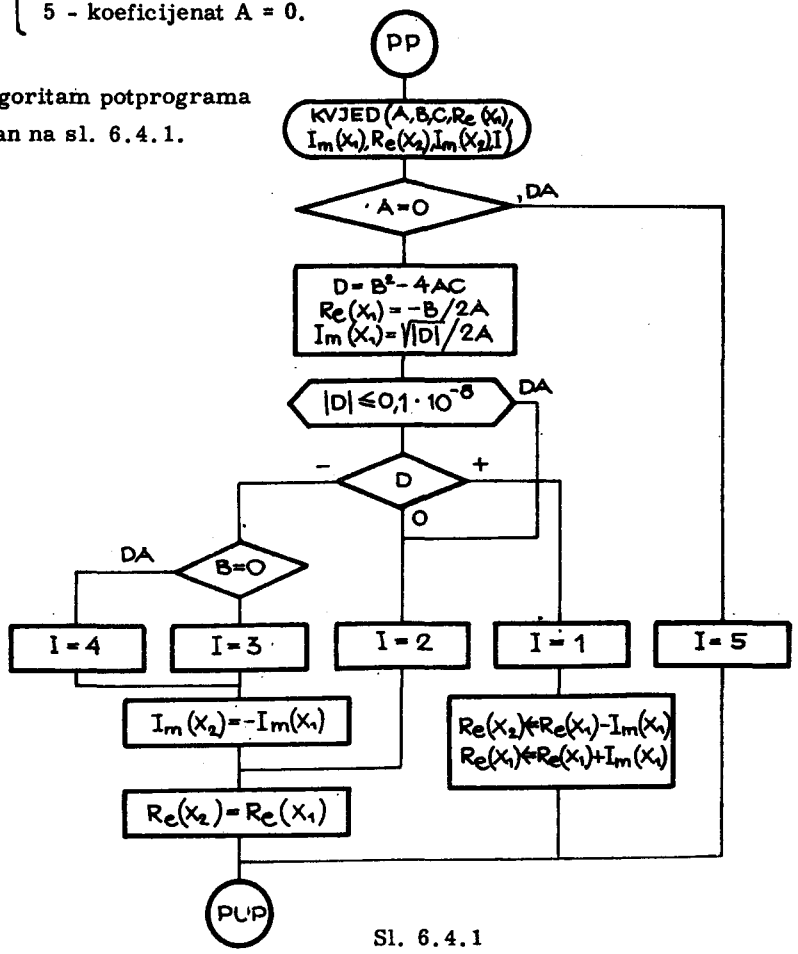
Sastavni potprogram za rešavanje kvadratne jednačine

$$Ax^2 + Bx + C = 0 \qquad (6.4.4)$$

Ulazne veličine u potprogramu su koeficijenti A, B i C, a izlazne veličine $Re(x_1)$, $I_m(x_1)$, $Re(x_2)$, $I_m(x_2)$, gde su x_1 i x_2 rešenja kvadratne jednačine (6.4.4.), i promenljiva I koja dobija u potprogramu različite vrednosti, u zavisnosti od karaktera rešenja kvadratne jednačine. Tako je

$$I = \begin{cases} 1 - \text{koreni realni i različiti,} \\ 2 - \text{koreni realni i jednaki,} \\ 3 - \text{koreni konjugovano kompleksni,} \\ 4 - \text{koreni imaginarni,} \\ 5 - \text{koeficijenat A = 0.} \end{cases}$$

Algoritam potprograma je prikazan na sl. 6.4.1.



Sl. 6.4.1

U FORTRAN-jeziku ovaj potprogram može se zapisati u vidu opšteg potprograma u obliku

```
      SUBROUTINE KVJED(A,B,C,REX1,IMX1,REX2,IMX2,I)
      REAL IMX1,IMX2
      IF(A) 20,21,20
   20 DISKR=B**2-4.*A*C
      REX1=-B/(2.*A)
      IMX1=SQRT(ABS(DISKR))/(2.*A)
      IF(ABS(DISKR)-1.0E-6) 11,11,22
   22 IF(DISKR) 10,11,12
   10 IF(B) 13,14,13
   13 I=3
   15 IMX2=-IMX1
   16 REX2=REX1
      RETURN
   14 I=4
      GO TO 15
   11 I=2
      GO TO 16
   12 I=1
      REX2=REX1-IMX1
      REX1=REX1+IMX1
      RETURN
   21 I=5
      RETURN
      END
```

U ovom potprogramu vrši se ispitivanje uslova

$$|D| < 1.0.10^{-6}$$

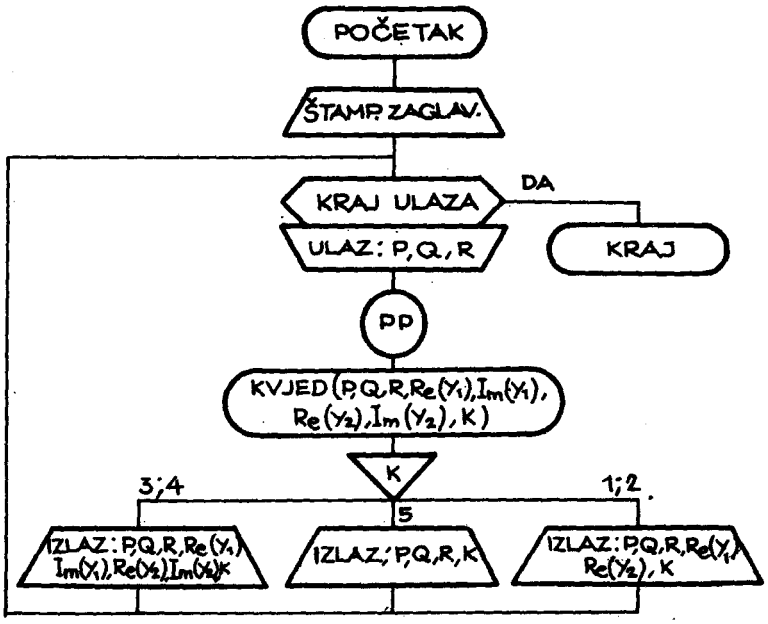
da bi se izbegao uticaj greške koja se javlja pri prevodjenju koeficijenata jednačine u interni kod računara (binarno kodirani heksadekadni brojni sistem), kao i uticaj greške usled računanja sa približnim brojevima.

Primeniti opšti potprogram za rešavanje kvadratne jednačine (KVJED) na rešavanje proizvoljnog broja kvadratnih jednačina, pri čemu su koeficijenti svake kvadratne jednačine zadati na jednoj kartici sa opisom F6.2.

Algoritam za ovo izračunavanje prikazan je na sl. 6.4.2, gde je pretpostavljeno da je opšti oblik kvadratne jednačine

$$Py^2 + Qy + R = 0 \qquad (6.4.5)$$

a promenljiva I svojom vrednošću odredjuje karakter rešenja, kako je to opisano u opštem potprogramu na sl. 6.4.1. FORTRAN-program sastav-



Sl. 6.4.2

ljen po algoritmu na sl. 6.4.2. ima sledeći izgled

```
      REAL IMY1,IMY2
      WRITE(6,10)
   10 FORMAT(' ',3X,'A',7X,'B',7X,'C',10X,'X1',14X,'X2',8X,
     *'I'/' ',24X,'REALAN  IMAGIN  REALAN  IMAGIN'/)
   13 READ(5,11,END=14) P,Q,R
   11 FORMAT(3F6.2)
      CALL KVJED(P,Q,R,REY1,IMY1,REY2,IMY2,I)
      GO TO (16,16,15,15,17),I
   15 WRITE(6,20) P,Q,R,REY1,IMY1,REY2,IMY2,I
   20 FORMAT(' ',7(F6.2,2X),I1)
      GO TO 13
   16 WRITE(6,30) P,Q,R,REY1,REY2,I
   30 FORMAT(' ',3(F6.2,2X),F6.2,10X,F6.2,10X,I1)
      GO TO 13
   17 WRITE(6,40) P,Q,R,I
   40 FORMAT(' ',3(F6.2,2X),32X,I1)
   14 STOP
      END
```

Za ulazne podatke, za rešavanje 4 kvadratne jednačine, po gornjem programu, dobijaju se rezultati u obliku tabele:

A	B	C	X1		X2		I
			REALAN	IMAGIN	REALAN	IMAGIN	
1.00	-8.00	-48.00	12.00		-4.00		1
1.00	-2.40	1.44	1.20		1.20		2
6.00	0.0	24.00	0.0	2.00	0.0	-2.00	4
1.00	-10.00	41.00	5.00	4.00	5.00	-4.00	3
0.0	-10.00	0.0					5

6.4.1. Promenljivi izlaz iz potprograma

Izlaz iz funkcijskog i opšteg potprograma vrši se preko naredbe RETURN. U slučaju funkcijskog potprograma naredba RETURN vrši povratak iz potprograma u program, i to u onaj aritmetički izraz programa, u kojem se kao argument pojavilo ime funkcijskog potprograma,po kojem se došlo iz programa u potprogram. U slučaju opšteg potprograma naredba RETURN vrši povratak iz potprograma u program i to na naredbu koja neposredno sledi iza naredbe CALL kojom je izvršen prelaz iz programa u potprogram.

Pored ovakvog povratka iz opšteg potprograma u program, može se vršiti i povratak na naredbu sa odredjenim obeležjem u programu. U ovom slučaju naredba povratka ima oblik

RETURN i (6.4.6)

gde je i ceo neoznačen broj veći od nule ili ime celobrojne promenljive. Brojna vrednost i ukazuje na obeležje naredbe, zadato medju stvarnim argumentima potprograma. Obeležja naredbi u programu na koja se može doći iz potprograma navode se kao stvarni argumenti potprograma u obliku

& n (6.4.7)

gde je

& - simbol koji ukazuje da je argument obeležje naredbe, a

n - obeležje jedne izvršne FORTRAN naredbe u programu.

Kako se fiktivni i stvarni argumenti moraju slagati po broju, redu i vrsti, to se na odgovarajućem mestu, u nizu fiktivnih argumenata piše *. Tako, naredba (6.4.6) vrši prelaz iz potprograma u program na naredbu

čije je obeležje n_i , gde je i redni broj obeležja u listi stvarnih argumenata potprograma.

U ranijem primeru, za rešavanje kvadratne jednačine, u programu se vrši prelaz na različite naredbe štampanja u zavisnosti od veličine I, kojoj se dodeljuje brojna vrednost u potprogramu (KVJED). Ako se koristi naredba (6.4.6), može se isti program napisati u obliku

```
      REAL IMY1,IMY2
      WRITE(6,10)
10    FORMAT(' ',3X,'A',7X,'B',7X,'C',10X,'X1',14X,'X2',8X,
     *'I'/' ',24X,'REALAN  IMAGIN  REALAN  IMAGIN'/)
13    READ(5,11,END=14) P,Q,R
11    FORMAT(3F6.2)
      CALL KVJED(P,Q,R,REY1,IMY1,REY2,IMY2,I,&16,&15,&17)
15    WRITE(6,20) P,Q,R,REY1,IMY1,REY2,IMY2,I
20    FORMAT(' ',7(F6.2,2X),I1)
      GO TO 13
16    WRITE(6,30) P,Q,R,REY1,REY2,I
30    FORMAT(' ',3(F6.2,2X),F6.2,10X,F6.2,10X,I1)
      GO TO 13
17    WRITE(6,40) P,Q,R,I
40    FORMAT(' ',3(F6.2,2X),32X,I1)
14    STOP
      END
```

Opšti potprogram u ovom slučaju ima sledeći izgled:

```
      SUBROUTINE KVJED(A,B,C,REX1,IMX1,REX2,IMX2,I,*,*,*)
      REAL IMX1,IMX2
      IF(A) 20,21,20
20    DISKR=B**2-4.*A*C
      REX1=-B/(2.*A)
      IMX1=SQRT(ABS(DISKR))/(2.*A)
      IF(ABS(DISKR)-1.0E-6) 11,11,22
22    IF(DISKR) 10,11,12
10    IF(B) 13,14,13
13    I=3
15    IMX2=-IMX1
      J=2
16    REX2=REX1
      RETURN J
14    I=4
      GO TO 15
11    I=2
      J=1
      GO TO 16
12    I=1
      REX2=REX1-IMX1
      REX1=REX1+IMX1
      RETURN 1
21    I=5
      RETURN 3
      END
```

6.5. Načini prenošenja argumenata iz programa u potprograme

Funkcijski i opšti potprogram predstavljaju posebne programske celine. Ovakvi potprogrami kada se jedanput napišu mogu se po potrebi koristiti u različitim programima. Sve promenljive i obeležja koja se javljaju u potprogramu nezavisni su od onih u programu. Tako se ista obeležja i imena promenljivih mogu pojaviti u potprogramu i programu. Medjutim, za ista imena promenljivih, u programu i potprogramu, angažuju se različiti registri u memoriji. U opštem potprogramu za rešavanje kvadratne jednačine, promenljivim A, B i C dodeljuju se brojne vrednosti stvarnih argumenata P, Q, R (vidi primer na kraju odeljka 6.4). Promenljiva I javlja se kao stvarni argument i kao fiktivni argument potprograma. Medjutim, brojna vrednost promenljive I u programu biće u jednom, a brojna vrednost promenljive u potprogramu biće u drugom memorijskom registru. Ovakav način prenošenja vrednosti argumenata zove se direktan prenos argumenata iz programa u potprogram.

Prema tome, u slučaju direktnog prenosa argumenata iz programa u potprogram, vrednosti argumenata se prenose iz registara memorije -u kojima se nalaze stvarni argumenti - u registre memorije angažovane za fiktivne argumente potprograma.

Pored ovog načina prenošenja argumenata iz programa u potprogram, može se koristiti indirektan prenos argumenata. U ovom slučaju, kao fiktivni argument pojavljuje se adresa registra u kojem se nalazi stvarni argument programa. Na ovaj način se vrednost argumenta uzima iz registra memorije u kojem se nalazi stvarni argument. Da bi se ukazalo na to da na mesto fiktivnog argumenta dolazi adresa, a ne vrednost stvarnog argumenta, fiktivni argument se piše izmedju kosih crta, tj.

$$/a/ \qquad (6.5.1)$$

gde je a ime fiktivnog argumenta.

Tako ako bismo u opštem potprogramu za rešavanje kvadratne jednačine, u prvoj naredbi, fiktivne argumente A, B i C napisali izmedju kosih crta, tj. potprogram u obliku

```
      SUBROUTINE KVJED(/A/,/B/,/C/,REX1,IMX1,REX2,IMX2,I,*,*,*)
      REAL IMX1,IMX2
      IF(A) 20,21,20
   20 DISKR=B**2-4.*A*C
      REX1=-B/(2.*A)
      IMX1=SQRT(ABS(DISKR))/(2.*A)
      IF(ABS(DISKR)-1.0E-6) 11,11,22
   22 IF(DISKR) 10,11,12
   10 IF(B) 13,14,13
   13 I=3
   15 IMX2=-IMX1
      J=2
   16 REX2=REX1
      RETURN J
   14 I=4
      GO TO 15
   11 I=2
      J=1
      GO TO 16
   12 I=1
      REX2=REX1-IMX1
      REX1=REX1+IMX1
      RETURN 1
   21 I=5
      RETURN 3
      END
```

tada se brojne vrednosti stvarnih argumenata P, Q i R neće dodeliti fiktivnim argumentima A, B i C, već će se u potprogramu namesto fiktivnih argumenata /A/, /B/ i /C/ čuvati adrese registara u kojima se nalaze brojne vrednosti promenljivih P, Q i R. Na ovaj način brojne vrednosti argumenata P, Q, R koriste se u potprogramu, indirektno iz registara memorije u kojima se one čuvaju u programu, a preko odgovarajućih adresa ovih registara.

6.6. Promenljivi ulazi u potprograme

U funkcijski i opšti potprogram dolazi se iz programa, na prvu izvršnu naredbu koja sledi iza naredbe FUNCTION ili SUBROUTIONE. Pored ovakvog prelaza na potprogram, kod ovih potprograma može se koristiti i promenljivi ulaz u potprogram. Pod pojmom "ulaz u potprogram" podrazumeva se mesto u programskom algoritmu na koje se vrši prelazak, kada se prelazi iz programa na potprogram. Promenljivi ulazi u potprogram označavaju se naredbom

ENTRY ime_i (lista) (6.6.1)

gde je

ENTRY - službena reč koja označava mesto ulaska u potprogram,

ime_i - naziv i-tog ulaza u potprogram, koji se definiše na isti način kao i ime potprograma,

lista - spisak fiktivnih argumenata i-tog ulaza u potprogram, medju sobom razdvojenih zarezima. Ovi fiktivni argumenti se definišu na isti način kao i fiktivni argumenti potprograma, ali mogu biti različiti po broju, redu i vrsti od fiktivnih argumenata potprograma.

Na ovaj način može biti definisan veći broj mesta u potprogramu, od kojih može početi izvršavanje potprograma, pored normalnog prelaza iz programa na potprogram od prve izvršne naredbe iza naredbe FUNCTION, odnosno SUBROUTINE. Na ma koje mesto potprograma, označeno kao mogući ulaz naredbom (6.6.1), može se doći u slučaju funkcijskog potprograma, navodjenjem argumenta aritmetičkog izraza u obliku

ime_i (lista) (6.6.2)

gde je

ime_i - naziv i-tog ulaska u potprogram,

lista - spisak stvarnih argumenata i-tog ulaza, medju sobom razdvojenih zarezima. Ovi stvarni argumenti se definišu na isti način kao i stvarni argumenti potprograma, samo što se po broju, redu i vrsti moraju slagati sa fiktivnim argumentima i-tog ulaza u potprogram.

U slučaju opšteg potprograma, prelaz na ulaz, definisan naredbom (6.6.1), vrši se naredbom

CALL ime_i (lista) (6.6.3)

gde je

CALL - službena reč koja označava pozivanje opšteg potprograma,

ime_i - naziv i-tog ulaska u potprogram,

lista - spisak stvarnih argumenata i-tog ulaza u potprogram, medju sobom razdvojenih zarezima. Ovi stvarni argumenti se definišu na isti način kao i stvarni argumenti potprograma, samo što se po broju, redu i vrsti moraju slagati sa fiktivnim argumentima i-tog ulaza u potprogram.

Naredbe (6.6.1) ne utiču na redosled izvršavanja naredbi potprograma. Ove naredbe se ne smeju nalaziti u okviru programskih ciklusa definisanih naredbama DO.

Primena naredbe (6.6.1), za definisanje više ulaza u potprogram, u programiranju je korisna kada se žele na različitim mestima programa definisati različiti argumenti, koji se ne slažu po broju, vrsti i redu sa argumentima definisanim u prvoj naredbi potprograma (FUNCTION, odnosno SUBROUTINE). Pored toga, ova naredba je korisna i kada se na različitim mestima programa definišu različite izlazne veličine potprograma.

Primer

Veličine A i B odredjuju Dekartove koordinate na sledeći način

$$X = \sqrt{A^2 + B^2} \qquad (6.6.4)$$

$$Y = A + B \qquad (6.6.5)$$

Sastaviti program koji će izračunavati Dekartove i polarne koordinate tačaka, na osnovu zadatih vrednosti A i B. Na sl. 6.6.1 prikazan je algoritam za rešavanje ovog zadatka.

Izračunavanje polarnih koordinata vrši se preko potprograma, čiji je algoritam prikazan na sl. 6.6.2. Kao što se vidi, postoje tri moguća ulaza u ovaj potprogram PP1, PP2 i PP3. Ulaz PP1 obezbedjuje izračunavanje

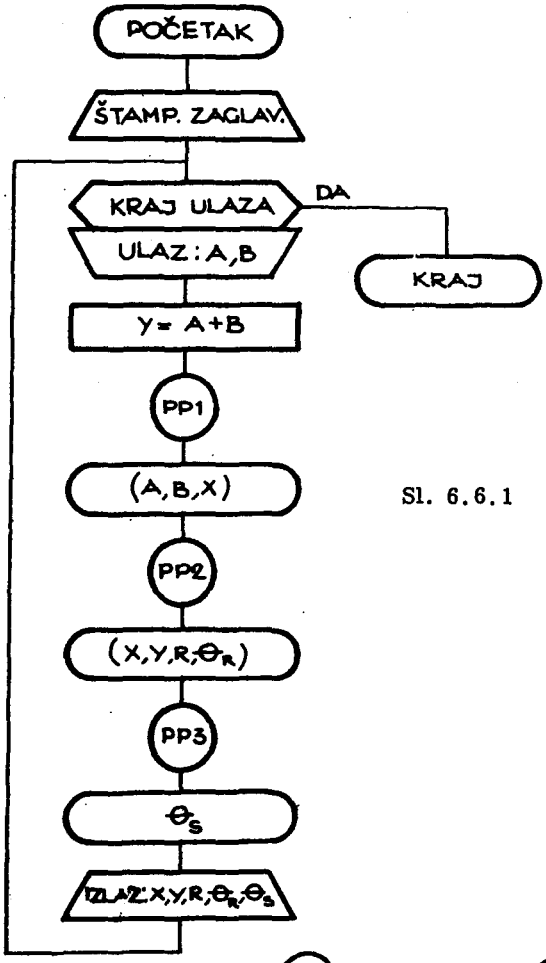
$$R = \sqrt{X^2 + Y^2} \qquad (6.6.6)$$

Ulaz PP2 obezbedjuje izračunavanje ugla u radijanima

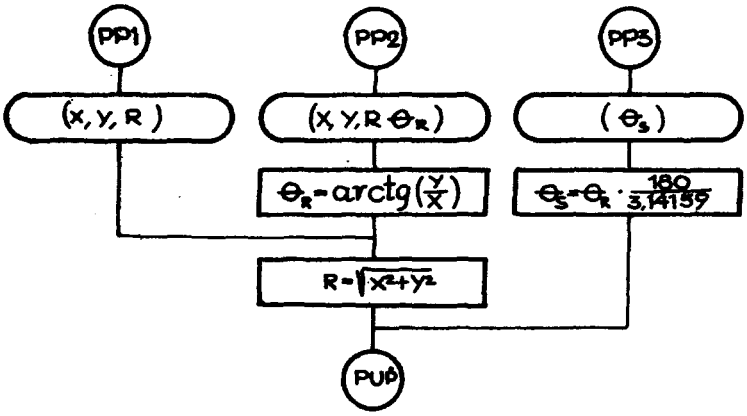
$$Q_r = \text{arc tg } \frac{X}{Y} \qquad (6.6.7)$$

i R po formuli (6.6.6), a ulaz PP3 obezbedjuje izračunavanje ugla u stepenima na osnovu zadatog ugla Q_r u radijanima.

$$Q_s = \frac{180}{3,14159} \cdot Q_r \qquad (6.6.8)$$


POČETAK
ŠTAMP. ZAGLAV.
KRAJ ULAZA
DA
KRAJ
ULAZ: A,B
Y= A+B
PP1
(A,B,X)
PP2
(X,Y,R,Θ_R)
PP3
Θ_S
IZLAZ: X,Y,R,Θ_R,Θ_S

Sl. 6.6.1


PP1
(X,Y,R)
PP2
(X,Y,R Θ_R)
Θ_R=arctg(Y/X)
PP3
(Θ_S)
Θ_S=Θ_R·180/3,14159
R=√(X²+Y²)
PUŠ

Sl. 6.6.2

Program sastavljen prema algoritmu na sl. 6.6.1 ima sledeći izgled:

```
      WRITE(6,10)
   10 FORMAT(' ',3X,'X',8X,'Y',8X,'R',10X,
     *'TETA'/' ',26X,'RADIJANA STEPENI'/)
   50 READ(5,20,END=40) A,B
   20 FORMAT(2F6.2)
      Y=A+B
      CALL KORR(A,B,X)
      CALL KORRT(X,Y,R,RADIJ)
      CALL KORTS(STEPEN)
      WRITE(6,30) X,Y,R,RADIJ,STEPEN
   30 FORMAT(' ',3(F7.2,2X),F6.3,2X,F7.2)
      GO TO 50
   40 STOP
      END
```

a potprogram, prema algoritmu na sl. 6.6.2, biće

```
SUBROUTINE KORRT(X,Y,R,TETA)
TETA=ATAN(Y/X)
ENTRY KORR(X,Y,R)
R=SQRT(X*X+Y*Y)
RETURN
ENTRY KORTS(TETAS)
TETAS=TETA*180./3.14159
RETURN
END
```

Potprogram nosi ime KORRT i ima 4 fiktivna argumenta, od kojih su X i Y ulazne vrličine, a R i TETA izlazne veličine. Pored toga, potprogram sadrži dva ulazna mesta, pri čemu ulazno mesto sa imenom KORR ima tri fiktivna argumenta, od kojih su prva dva ulazne veličine, a drugi izlazna veličina potprograma. Poredjenjem normalnog ulaza u potprogram, preko imena KORRT sa ulaznim mestom KORR vidimo da se u prvom računaju obe polarne koordinate θ i R, a u drugom samo koordinata R. Ulazno mesto sa imenom KORTS ima jedan fiktivni argument i to izlazni. Na ovo ulazno mesto ima smisla da se predje iz programa samo ako je pre toga potprogram pozivan preko imena KORRT, čime je promenljiva TETA dobila brojnu vrednost, pa je aritmetički izraz na desnoj strani aritmetičke naredbe

```
TETAS=TETA*180./3.14159
```

definisan i promenljiva TETAS može dobiti korektnu brojnu vrednost kada se na potprogram dodje preko ulaza KORTS.

Za ulazne podatke

A	B
13,20	-7,50
6,18	7,50
4,37	-6,07
11,87	-3,60

izlazni rezultati se dobijaju u obliku tabele

X	Y	R	TETA	
			RADIJANA	STEPENI
15.18	5.70	16.22	0.359	20.58
9.72	13.68	16.78	0.953	54.61
7.48	-1.70	7.67	-0.223	-12.81
12.40	8.27	14.91	0.588	33.69

6.7. Imena potprograma koji se javljaju kao argumenti drugih potprograma

Već je rečeno da se kao fiktivni, odnosno odgovarajući stvarni, argument funkcijskog i opšteg potprograma može pojaviti ime drugog potprograma. Medjutim, ako u programu stoji naredba

```
CALL FUN(MAT,A,D)                                    (6.7.1)
```

gde je MAT potprograma, a A i D imena promenljivih, tada program za prevodjenje sa FORTRAN-jezika na mašinski jezik ne raspolaže informacijom o tome da li je MAT ime potprograma ili ime promenljive. Prema tome, sva imena potprograma koja se javljaju kao stvarni argumenti u drugim potprogramima moraju biti deklarisana kao takva u programu. Ovo se vrši posebnom naredbom

EXTERNAL (lista) (6.7.2)

gde je

EXTERNAL - službena reč u FORTRAN-jeziku,

lista - spisak imena potprograma, medju sobom razdvojeni zarezima, koja se javljaju kao argumenti u potprogramima.

Naredba (6.7.2) piše se u programu pre prve izvršne naredbe programa.

Primer

Sastaviti potprogram za numeričko izračunavanje odredjenog integrala po Simpsonovom obrascu

$$Y = \int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + \ldots + 4f(x_{n-1}) + f(x_n)\right] \quad (6.7.3)$$

gde je

$$h = \frac{b - a}{n} \quad (6.7.4)$$

a n broj podintervala na koji se deli interval integracije $[a, b]$. U (6.7.3) vrednost apscise x_i odredjena je relacijom

$$x_i = a + ih, \qquad i = 0, 1, \ldots, n \quad (6.7.5)$$

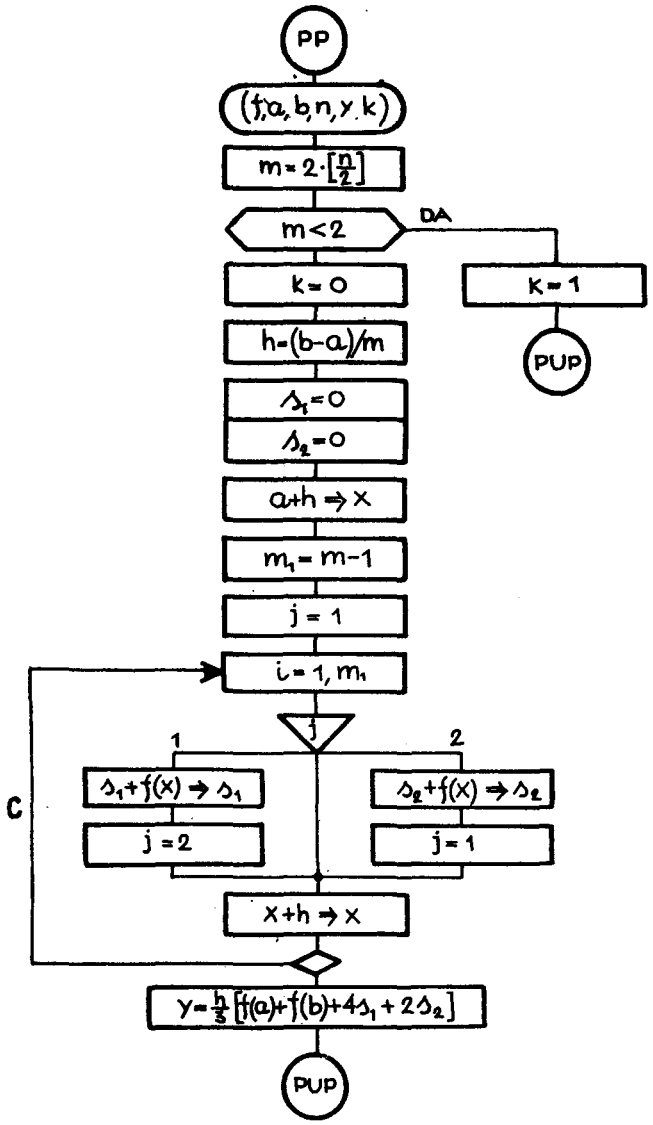
odakle sledi da je

$$f(x_0) = f(a)$$
$$f(x_n) = f(b) \quad (6.7.6)$$

Ulazne veličine za potprogram jesu ime potprograma za izračunavanje vrednosti funkcije $f(x_i)$, granice integracije a i b, kao i broj podintervala n. Izlazna veličina potprograma je vrednost integrala Y i broj k koji u potprogramu dobija sledeće vrednosti

$$k = \begin{cases} 0 - \text{ako je korektno izračunata vrednost integrala,} \\ 1 - \text{ako je ulazna veličina } n < 2, \text{ i tada se vrednost integrala ne izračunava.} \end{cases}$$

Algoritam sastavljen po gornjim zahtevima prikazan je na sl.6.7.1. Kako se Simpsonov obrazac (6.7.3) primenjuje za paran broj podintervala to se na početku algoritma, na osnovu zadatog broja n izračunava broj



Sl. 6.7.1

$$m = 2 \cdot \left[\frac{n}{2}\right] \qquad (6.7.7)$$

gde srednja zagrada označava celobrojni deo količnika. Promenljiva j, u okviru algoritamskog ciklusa C, definiše da li se vrši izračunavanje sume ordinata funkcije f(x), koja se kasnije množi sa 4 (suma s_1). Opšti potpro-

gram na FORTRAN-jeziku sastavljen prema algoritmu na sl. 6.7.1 ima sledeći izgled:

```
      SUBROUTINE SIMPSN(FUN,A,B,N,Y,K)
      M=2*(N/2)
      IF(M-2) 10,11,11
   10 K=1
      RETURN
   11 K=0
      H=(B-A)/M
      S1=0.
      S2=0.
      X=A+H
      M1=M-1
      J=1
      DO 12 I=1,M1
      GO TO (13,14),J
   13 S1=S1+FUN(X)
      J=2
   12 X=X+H
      Y=(H/3.)*(FUN(A)+FUN(B)+4.*S1+2.*S2)
      RETURN
   14 S2=S2+FUN(X)
      J=1
      GO TO 12
      END
```

Primeniti izložen potprogram na izračunavanje integrala

$$\int_0^1 \frac{dx}{1+x^2} \tag{6.7.8}$$

Prema tome, funkcija f(x), u ovom slučaju, ima oblik

$$f(x) = \frac{1}{1+x^2} \tag{6.7.9}$$

U ovom slučaju treba sastaviti program čiji će zadatak biti da pozove potprogram SIMPSN i da štampa vrednost izračunatog integrala. Medjutim, potprogram SIMPSN sadrži fiktivan argument FUN, koji predstavlja ime funkcije, pa prema tome i medju stvarnim argumentima mora se pojaviti ime potprograma po kojem se vrši izračunavanje vrednosti funkcije (6.7.9). Program u ovom slučaju ima sledeći izgled

```
      EXTERNAL F679
      READ(5,10) A,B,N
   10 FORMAT(2F6.2,I2)
      CALL SIMPSN(F679,A,B,N,VRED,I)
      IF(I) 11,12,11
   11 WRITE(6,20)
   20 FORMAT(' VREDNOST INTEGRALA NIJE IZRACUNATA JER'
     *' JE N MANJE OD 2')
      STOP
   12 WRITE(6,30) VRED
   30 FORMAT(' VREDNOST INTEGRALA JE',E15.7)
      STOP
      END
```

Naredba EXTERNAL ukazuje programu za prevodjenje sa FORTRAN-jezika na mašinski jezik da ime F679 u naredbi CALL nije ime promenljive, već ime potprograma. Pri prelasku iz programa u potprogram SIMPSN, fiktivno ime potprograma FUN zamenjuje se stvarnim imenom F679. To je ime funkcijskog potprograma koji se mora napisati odvojeno za svaku funkciju čiji se integral izračunava. Tako za funkciju datu sa (6.7.9) ovaj potprogram ima sledeći izgled:

```
FUNCTION F679(X)
F679=1./(1.+X*X)
RETURN
END
```

Ovaj funkcijski potprogram poziva se u opštem potprogramu SIMPSN, jer je prelazom iz programa na potprogram SIMPSN fiktivno ime potprograma FUN zamenjeno stvarnim imenom F679. Vrednost integrala (6.7.8) štampana je u obliku

```
VREDNOST INTEGRALA JE  0.7853980E 00
```

U programu je uzeto da je A = 0; B = 1,0 i N = 20.

6.8. Nizovi kao argumenti potprograma

Niz u potprogramu može biti argument potprograma ili ne. Ako niz nije argument potprograma, tada se maksimalne vrednosti njegovih indeksa moraju definisati u naredbi DIMENSION, kao i kod svih drugih programa. Za ovako definisan niz rezerviše se potreban prostor u memoriji, i ovaj prostor se koristi samo u okviru odgovarajućeg potprograma.

Medjutim, ako je niz argument potprograma, tada se ime niza mora pojaviti takodje u naredbi DIMENSION u potprogramu, ali se ne moraju navoditi maksimalne vrednosti indeksa, jer je za ovakav niz prostor u memoriji rezervisan u okviru programa, a ne potprograma. Prema tome, ime niza u naredbi DIMENSION, kada je niz argument potprograma, služi samo zato da ukaže da je odgovarajući argument niz, a ne zato da rezerviše memorijski prostor. Zato se za ovakve nizove u DIMENSION-naredbi najčešće navodi samo ime i početne vrednosti indeksa. Tako, ako se napiše

```
FUNCTION PP(A,N)
DIMENSION A(1),C(20)
```

to znači da potprogram PP ima za prvi fiktivni argument jednodimenzionalni niz A, čija će maksimalna vrednost indeksa biti zadata u programu, a konkretan broj elemenata niza A, koji se koristi u potprogramu, zadat je fiktivnim argumentom N. Naredba DIMENSION definiše u ovom slučaju da je argument A jednodimenzionalni niz, a niz C pošto nije argument potprograma biće definisan u potprogramu i za njega biće rezervisano 20 memorijskih registara.

Prema tome, elemenat liste DIMENSION-naredbe u potprogramu, kada niz nije argument potprograma ima oblik

$$\text{ime}(i_{max}) \qquad (6.8.1)$$

gde je

ime - naziv niza, a

i_{max} - maksimalna vrednost indeksa niza.

Medjutim, ako je niz argument potprograma, tada se može pisati oblik (6.8.1), ali će on imati isti efekat kao i

$$\text{ime}(1) \qquad (6.8.2)$$

Pošto broj elemenata niza može biti promenljiv kada je niz argument potprograma, to se dozvoljava i oblik

$$\text{ime}(n) \qquad (6.8.3)$$

gde je

ime - naziv niza, a

n - ime celobrojne promenljive, čijom brojnom vrednošću se definiše broj elemenata niza u potprogramu.

Oblik (6.8.3) je dozvoljen samo u potprogramima i to za nizove koji se javljaju kao argumenti potprograma. Za jednodimenzione nizove, koji su argumenti potprograma, oblici (6.8.1), (6.8.2) i (6.8.3) imaju isto značenje.

Medjutim, kada je u pitanju višedimenzioni niz, tada je opšti oblik (6.8.3) sledeći

ime (lista) (6.8.4)

gde je

ime - naziv niza, a

lista - spisak, od najviše 7, imena celobrojnih promenljivih, medju sobom razdvojenih zarezima. Brojne vrednosti ovih promenljivih definišu maksimalne vrednosti indeksa u potprogramu.

Oblik (6.8.4) omogućuje različiti raspored elemenata višedimenzionalnog niza. U memoriji računara višedimenzionalni niz se registruje kolona po kolona. Tako će elemenat ime (2,2) biti treći elemenat u nizu, ako je u pitanju niz ime (2,3)

ime(1,1)	ime(1,2)	ime(1,3)
ime(2,1)	ime(2,2)	ime(2,3)

odnosno peti elemenat ako je u pitanju niz ime (3,4):

ime(1,1)	ime(1,2)	ime(1,3)	ime(1,4)
ime(2,1)	ime(2,2)	ime(2,3)	ime(2,4)
ime(3,1)	ime(3,2)	ime(3,3)	ime(3,4)

Sledeći primer ilustruje, različite rasporede elemenata dvodimenzionalnog niza u potprogramu za isti dati dvodimenzionalni niz u programu.

Primer

Sastaviti program koji formira dvodimenzionalni niz

11	12	13	14	
21	22	23	24	(6.8.5)
31	32	33	34	

i potprogram koji štampa elemente dvodimenzionalnog niza (6.8.5) za različite oblike DIMENSION-naredbe u potprogramu.

Neka je ime dvodimenzionalnog niza (6.8.5) NIZ, tada će program imati sledeći izgled

```
      DIMENSION NIZ(3,4)
      DO 10 I=1,3
      DO 10 J=1,4
   10 NIZ(I,J)=10*I+J
      WRITE(6,20)
   20 FORMAT(' ',3X,'N',3X,'M',10X,
     *       'NIZOVI U POTPROGRAMU'/)
      DO 30 I=1,3
      DO 30 J=1,4
   30 CALL VARNIZ(NIZ,I,J)
      STOP
      END
```

gde je VARNIZ ime opšteg potprograma, čiji su argumenti: ime dvodimenzionalnog niza (NIZ), broj vrsta (I) i broj kolona (J) istog niza u potprogramu. Neka su broj vrsta i kolona, u potprogramu, definisani kao promenljive, tada potprogram ima sledeći izgled

```
      SUBROUTINE VARNIZ(A,N,M)
      DIMENSION A(N,M)
      WRITE(6,10) N,M,(A(1,J),J=1,M)
   10 FORMAT(' ',2I4,3X,4I5)
      IF(N-1) 20,30,20
   20 DO 40 I=2,N
   40 WRITE(6,50) (A(I,J),J=1,M)
   50 FORMAT(' ',11X,4I5)
   30 RETURN
      END
```

Dvodimenzionalni niz (6.8.5) u memoriji, registrovan je kolona po kolona. Prvi indeks niza u naredbi DIMENSION definiše broj vrsta dvodimenzionalnog niza, odnosno broj elemenata kolone. Prema tome, u potprogramu će biti korišćeni elementi niza (6.8.5), tako da svaka kolona sadrži N elemenata niza u programu. Rezultati gornjeg potprograma štampani su u sledećem rasporedu

```
N   M          NIZOVI U POTPROGRAMU

1   1      11
1   2      11   21
1   3      11   21   31
1   4      11   21   31   12
2   1      11
           21
2   2      11   31
           21   12
2   3      11   31   22
           21   12   32
2   4      11   31   22   13
           21   12   32   23
3   1      11
           21
           31
3   2      11   12
           21   22
           31   32
3   3      11   12   13
           21   22   23
           31   32   33
3   4      11   12   13   14
           21   22   23   24
           31   32   33   34
```

Kao što se vidi iz prikazanih rezultata, dvodimenzionalni niz sa dimenzijama A(2,2) u potprogramu koristiće sledeće elemente niza (6.8.5) iz programa

```
11   31
21   12
```

a ne

```
11   12
21   22
```

jer svaka kolona dvodimenzionalnog niza u potprogramu sadrži dva elementa, iz niza elemenata (6.8.5) u programu poredjanih kolona po kolona.

Medjutim, ako se u naredbi DIMENSION potprograma, niz zapiše u obliku A(1,1), tj. potprogram u obliku

```
      SUBROUTINE VARNIZ(A,N,M)
      DIMENSION A(1,1)
      WRITE(6,10) N,M,(A(1,J),J=1,M)
   10 FORMAT(' ',2I4,3X,4I5)
      IF(N-1) 20,30,20
   20 DO 40 I=2,N
```

```
40 WRITE(6,50) (A(I,J),J=1,M)
50 FORMAT(' ',11X,4I5)
30 RETURN
   END
```

tada će nizovi u potprogramu imati sledeći izgled

N	M	NIZOVI U POTPROGRAMU			
1	1	11			
1	2	11	21		
1	3	11	21	31	
1	4	11	21	31	12
2	1	11			
		21			
2	2	11	21		
		21	31		
2	3	11	21	31	
		21	31	12	
2	4	11	21	31	12
		21	31	12	22
3	1	11			
		21			
		31			
3	2	11	21		
		21	31		
		31	12		
3	3	11	21	31	
		21	31	12	
		31	12	22	
3	4	11	21	31	12
		21	31	12	22
		31	12	22	32

U ovom slučaju svaka kolona dvodimenzionalnog niza u programu počinje narednim elementom niza (6.8.5), kada se ovaj shvati kao jednodimenzionalni niz poredjan u redosledu kolona po kolona. Zato niz A(2,2) u potprogramu ima oblik

11 21

21 31

jer prva kolona počinje elementom 11; a druga sledećim elementom, pošto prvi indeks u naredbi DIMENSION niza A ima vrednost 1.

Ako se u potprogramu, u naredbi DIMENSION, niz A zapiše tako da prvi indeks ima istu vrednost kao i u programu, tako da potprogram ima oblik

```
      SUBROUTINE VARNIZ(A,N,M)
      DIMENSION A(3,1)
      WRITE(6,10) N,M,(A(1,J),J=1,M)
   10 FORMAT(' ',2I4,3X,4I5)
      IF(N-1) 20,30,20
   20 DO 40 I=2,N
   40 WRITE(6,50) (A(I,J),J=1,M)
   50 FORMAT(' ',11X,4I5)
   30 RETURN
      END
```

tada će nizovi u potprogramu imati sledeći redosled

N	M	NIZOVI U POTPROGRAMU			
1	1	11			
1	2	11	12		
1	3	11	12	13	
1	4	11	12	13	14
2	1	11			
		21			
2	2	11	12		
		21	22		
2	3	11	12	13	
		21	22	23	
2	4	11	12	13	14
		21	22	23	24
3	1	11			
		21			
		31			
3	2	11	12		
		21	22		
		31	32		
3	3	11	12	13	
		21	22	23	
		31	32	33	
3	4	11	12	13	14
		21	22	23	24
		31	32	33	34

U ovom slučaju isti indeksi u potprogramu i programu odredjuju iste elemente niza (6.8.5).

Vrlo često u potprogramima višedimenzionalni nizovi tretiraju se kao jednodimenzionalni nizovi. Tada se potprogram VARNIZ može napisati u obliku

```
      SUBROUTINE VARNIZ(A,N,M)
      DIMENSION A(1)
      NM=N*M
      WRITE(6,10) N,M,(A(I),I=1,NM)
   10 FORMAT(' ',2I4,3X,12I3)
      RETURN
      END
```

U ovom slučaju niz (6.8.5) u programu biće u potprogramu uzet kao jednodimenzionalni niz i to u redosledu kolona po kolona niza (6.8.5). Tako da će za razne vrednosti N i M biti definisani sledeći jednodimenzionalni nizovi u potprogramu

N	M	NIZOVI U POTPROGRAMU
1	1	11
1	2	11 21
1	3	11 21 31
1	4	11 21 31 12
2	1	11 21
2	2	11 21 31 12
2	3	11 21 31 12 22 32
2	4	11 21 31 12 22 32 13 23
3	1	11 21 31
3	2	11 21 31 12 22 32
3	3	11 21 31 12 22 32 13 23 33
3	4	11 21 31 12 22 32 13 23 33 14 24 34

U potprogramima koji se odnose na matrični račun, i nalaze se u biblioteci gotovih potprograma računskih centara, najčešće se matrice tretiraju kao jednodimenzionalni nizovi. U ovom slučaju je važno uočiti da će redosled elemenata matrica u potprogramu biti kolona po kolona matrice, kao jednodimenzionalni niz.

7. ALGORITMI SA LOGIČKIM KONSTANTAMA I PROMENLJIVIM

7.1. Operacije poredjenja

7.1.1. Definicije operacija poredjenja

U mnogim problemima tok algoritma zavisi od odnosa nekih brojnih veličina. U dosadašnjim izlaganjima ovakve odnose uvek smo svodili na ispitivanje vrednosti aritmetičkog izraza, a tok algoritma menjali u zavisnosti od toga da li je vrednost izraza bila manja, jednaka ili veća od nule.

U FORTRAN-jeziku postoji mogućnost direktnog poredjenja brojnih veličina. Opšti oblik ovakve operacije poredjenja je

$$a \odot b \qquad (7.1.1)$$

gde je

a, b - aritmetički izrazi,

$\odot$ - operacija poredjenja.

Operacija poredjenja u (7.1.1) može biti jedna od operacija navnih u tabeli 7.1.1.

Tabela 7.1.1.

Operacije poredjenja		
u FORTRANU	u matematici	Opis
.EQ.	$=$	jednako
.GT.	$>$	veće
.GE.	$\geqslant$	veće ili jednako
.LT.	$<$	manje
.LE.	$\leqslant$	manje ili jednako
.NE.	$\neq$	različito.

Navedene oznake operacija poredjenja u tabeli 7.1.1., treba shvatiti kao jedan nedeljivi simbol FORTRAN-jezika. Operacija poredjenja (7.1.1) uvek izražava jednu tvrdnju koja može biti istinita ili lažna.

Ako je tvrdnja-iskaz izražen sa (7.1.1) istinit, označićemo ga, kako je to uobičajeno u algebri logike, sa 1, a ako je lažan sa 0.

Tako se može, u FORTRAN-jeziku, pisati operacija poredjenja

$$J.GT.5 \qquad (7.1.2)$$

što u matematičkoj notaciji predstavlja relaciju

$$j>5 \qquad (7.1.3)$$

U matematici je uobičajeno da se funkcija (7.1.1) zove predikat, pa ćemo ovaj termin koristiti u daljem izlaganju. Karakteristika funkcije (7.1.1), odnosno predikata, jeste da argumenti funkcije uzimaju vrednosti iz potencijalno beskonačnog skupa veličina, a sama funkcija uzima vrednosti iz skupa $\{0, 1\}$. Tako, argument j predikata (7.1.3) uzima vrednosti iz skupa celih brojeva, a vrednost poredjenja $j>5$ može biti istinita ili lažna, a to znači da uzima jednu od vrednosti, uslovno označenih sa 1 ili 0.

7.1.2. Naredba prelaza po vrednosti poredjenja

Grananje u programima po vrednosti operacije poredjenja, može se izvršiti pomoću naredbe

$$IF(p)naredba \qquad (7.1.4)$$

gde je

IF - službena reč,

p - predikat (definisan sa 7.1.1.),

naredba - jedna izvršna FORTRAN-naredba, osim druge naredbe IF po vrednosti poredjenja, ili DO-naredbe.

Naredba (7.1.4) izvršava se različito u zavisnosti od vrednosti predikata p:

1) Ako je vrednost predikata p = 1, tada se izvršava naredba zapisana desno od zatvorene zagrade u (7.1.4), a zatim mogu nastati dva slučaja:

a) Ako naredba u (7.1.4) nije uslovna ili bezuslovna naredba prelaska, tada se izvršava naredba koja sledi iza naredbe (7.1.4),

b) Ako je naredba u (7.1.4) uslovna ili bezuslovna naredba prelaza, tada se izvršava naredba ukazana ovom naredbom prelaza.

2) Ako je vrednost predikata p = 0, tada se ne izvršava naredba zapisana desno od zatvorene zagrade u (7.1.4), već odmah prelazi na naredbu koja sledi iza naredbe (7.1.4).

Tako, naredba

```
IF(J.GT.5) A=B+C                    (7.1.5)
```

ima sledeće dejstvo:

- ako je J veće od 5, izvršiće se naredba A=B+C, a zatim naredba koja sledi iza naredbe (7.1.5),
- ako je J manje ili jednako 5, preskočiće se naredba A=B+C i izvršiti naredba koja sledi iza (7.1.4),

Medjutim, u slučaju naredbe

```
IF(J.GT.5) GO TO 100                (7.1.6)
```

ako je J veće od 5, izvršiće se naredba bezuslovnog prelaska na naredbu sa obeležjem 100, a ako je J manje ili jednako 5, preći će se na naredbu koja sledi iza naredbe (7.1.6).

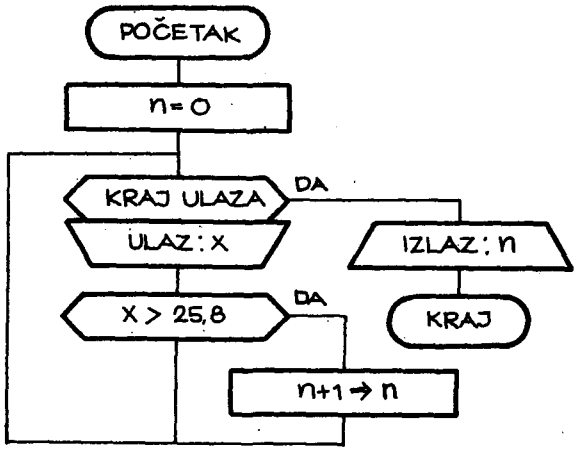
Primer

Zadat je niz brojeva x_i, i=1,2,... Svaki od brojeva x_i nalazi se na po jednoj kartici u polju od 1. do 10. kolone sa opisom F10.5. Odrediti koliko je od zadatih brojeva x_i veće od 25,8.

Algoritam za rešavanje ovog zadatka je prikazan na sl. 7.1.1.

Program sastavljen po algoritmu na sl. 7.1.1. ima sledeći izgled:

```
      N=0
   30 READ(5,100,END=20) X
  100 FORMAT(F10.5)
      IF(X.GT.25.8) N=N+1
      GO TO 30
   20 WRITE(6,40) N
   40 FORMAT(' N=',I4)
      STOP
      END
```



Sl. 7. 1. 1

7. 2. Logičke operacije

7. 2. 1. Logičke konstante i promenljive

U dvoznačnoj algebri logike (Bulovoj algebri) postoje dve konstante. Najčešće se ove konstante označavaju ciframa 0 i 1, pri čemu 0 predstavlja vrednost lažnog iskaza, a 1 vrednost istinitog iskaza. U FORTRAN-jeziku za logičke konstante koriste se simboli

.TRUE. (7. 2. 1)

za istinitost, i

.FALSE. (7. 2. 2)

za lažnost iskaza.

Logička konstanta se registruje u jednom memorijskom registru.

Imena logičkih promenljivih konstruišu se na isti način kao i imena brojnih promenljivih. Da se jedno ime promenljive u programu odnosi na logičku promenljivu, ukazuje se opisnim naredbama za deklaraciju vrste promenljive. Ova deklaracija se može izvršiti eksplicitno naredbom

LOGICAL lista (7. 2. 3)

gde je

LOGICAL - službena reč, a

lista - spisak imena promenljivih medju sobom razdvojenih zarezima, koje se deklarišu kao logičke promenljive.

Pored eksplicitne deklaracije logičkih promenljivih, može se izvršiti i implicitna deklaracija pomoću naredbe

IMPLICIT lista (7.2.4)

gde je

IMPLICIT - službena reč, a

lista - spisak elemenata medju sobom odvojenih zarezima.

Element liste u slučaju implicitne deklaracije logičkih promenljivih ima izgled

LOGICAL ($lista_i$) (7.2.5)

gde je

LOGICAL - službena reč, a

$lista_1$ - spisak velikih slova engleske azbuke, medju sobom razdvojena zarezima.

Implicitnom deklaracijom kao logičke promenljive deklarišu se sve promenljive u jednom programu, čija imena počinju jednim od navedenih slova u implicitnoj deklaraciji (7.2.4), gde je elemenata liste oblika (7.2.5).

Ako više uzastopnih slova engleske azbuke predstavljaju početna slova imena logičkih promenljivih, tada se elemenat liste u (7.2.4) može pisati u obliku

LOGICAL (x_1-x_2, x_3-x_4) (7.2.6)

Deklaracija (7.2.6) deklariše sve promenljive čija imena počinju od velikog slova x_1 do velikog slova x_2 engleske azbuke, odnosno od x_3 do x_4, kao logičke promenljive u programu.

Tako opisna naredba

```
LOGICAL AB1,L,BULL
```
(7.2.7)

deklariše promenljive AB1, L i BULL u programu kao logičke promenljive.

Opisna naredba

```
IMPLICIT LOGICAL(A-D,L)                    (7.2.8)
```

deklariše sve promenljive čija imena počinju slovima A, B, C, D i L kao logičke promenljive.

Više logičkih promenljivih sa zajedničkim imenom obrazuju niz. I sve što je rečeno za nizove u slučaju brojnih veličina važi i za nizove sa logičkim veličinama.

7.2.2. Definicije logičkih operacija

Logičke operacije definišu se nad argumentima koji mogu uzimati vrednosti iz skupa od dva elementa $\{0, 1\}$. Rezultati logičkih operacija uzimaju takodje vrednosti iz istog skupa elemenata $\{0, 1\}$. U algebri logike skup funkcija preko kojih se može izraziti proizvoljna funkcija algebre logike zove se pun sistem funkcija. Pun sistem funkcija grade različite funkcije algebre logike. U FORTRAN-jeziku su izabrane tri funkcije koje čine pun sistem funkcija, to su:

- negacija ili ne funkcija,
- konjukcija ili i funkcija, i
- disjunkcija ili ili funkcija.

Pomoću ove tri funkcije može se izraziti proizvoljna funkcija algebre logike. Sa druge strane, primena ovih funkcija najbliža je širem krugu ljudi, jer argumente povezuje na način koji je vrlo blizak uobičajenom načinu razmišljanja.

Funkcija negacije z je takva složena funkcija koja je istinita ako argument x nije istinit, odnosno lažna ako je argument x istinit. Ova funkcija se dobija primenom operacije negacije nad jednim argumentom

$$z = \bar{x} \qquad (7.2.9)$$

gde povlaka iznad x označava operaciju negacije, i čita se "ne x". U tabeli 7.2.1 data je definicija funkcije negacije.

x	$z = \bar{x}$
0	1
1	0

Tabela 7.2.1

Funkcija konjukcije z je takva složena funkcija algebre logike, koja je istinita samo ako su oba argumenta x i y, od kojih je sastavljena, istiniti. U svim drugim slučajevima ta funkcija je lažna. Ova funkcija se piše

$$z = x \wedge y \qquad (7.2.10)$$

gde simbol $\wedge$ označava operaciju konjukcije, i čita se "x i y". U tabeli 7.2.2 data je tabela istinitosti za funkciju (7.2.10).

Tabela 7.2.2.

x	y	$z = x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

Funkcija disjunkcije z je takva složena funkcija algebre logike, koja je lažna samo ako su oba argumenta x i y, od kojih je sastavljena, lažna. U svim drugim slučajevima ta funkcija je istinita. Ova funkcija se piše

$$z = x \vee y \qquad (7.2.11)$$

gde simbol $\vee$ označava operaciju disjunkcije, i čita se "x ili y". U tabeli 7.2.3 data je tabela istinitosti za funkciju (7.2.11).

Tabela 7.2.3.

x	y	$z = x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Definisane tri logičke operacije: negacija, konjukcija i disjunkcija predstavljaju logičke operacije u FORTRAN-jeziku. U tabeli 7.2.4 prikazani su simboli ovih operacija u FORTRANU.

Tabela 7.2.4

Logičke operacije		
u FORTRANU	u matematici	Opis
.NOT.	-	negacija
.AND.	$\wedge$	konjukcija
.OR.	$\vee$	disjunkcija

7.2.3. Logički izraz

Logički izraz je sastavljen od logičkih konstanti, logičkih promenljivih sa indeksom ili bez njega, i predikata medju sobom povezanih logičkim operacijama. Vrednost logičkog izraza odredjuje se izvršavanjem logičkih operacija sleva nadesno, pri čemu važi sledeći prioritet: najpre se izračunava vrednost predikata, a zatim redom logičkih operacija negacije, konjukcije i na kraju disjunkcije. Ako se želi drugačiji redosled u prioritetima operacija, to se može postići uvodjenjem zagrada. Deo logičkog izraza zapisan izmedju otvorene i zatvorene male zagrade ima najviši prioritet.

Dve logičke opereracije u logičkom izrazu mogu biti jedna do druge, samo ako je druga od njih operacija negacije.

Primer logičkog izraza u FORTRANU

B.AND.C.LT.3.6 (7.2.12)

Ovde je prvi argument i operacije logička promenljiva B, a drugi predikat C.LT.3.6. Promenljiva C jeste brojna promenljiva, i ako je $C < 3,6$, i B istinito, logički izraz (.7.2.12) je istinit; u svim drugim slučajevima on je lažan.

7.2.4. Dodeljivanje vrednosti logičkim promenljivim

7.2.4.1. Dodeljivanje vrednosti sa ulaza

Imena logičkih promenljivih, pojedini elementi logičkih nizova ili logički nizovi navode se u listi ulazne naredbe po istim pravilima kao i u slučaju brojnih veličina. Medjutim, u odgovarajućoj FORMAT-naredbi opisuje se polje u ulaznom slogu koje sadrži logičku konstantu, sa

nLw (7.2.13)

gde je

- n - ceo neoznačen broj koji ukazuje koliko puta se primenjuje opis L,
- L - simbol FORTRAN-jezika kojim se ukazuje da odgovarajuće polje u ulaznom slogu sadrži logičku konstantu,
- w - ceo neoznačen broj kojim se definiše širina polja u ulaznom slogu, odnosno broj kolona na kartici kada se radi o ulazu sa čitača kartica.

Polje na kartici, koje sadrži logičku konstantu može imati jednu kolonu ili više njih. Medjutim, sa gledišta registrovanja logičke konstante od značaja je samo prva kolona. U prvoj koloni polja mora se nalaziti bušen kod slova T, za konstantu.TRUE., odnosno kod slova F za konstantu .FALSE.

Ako su promenljive A i B deklarisane u programu kao logičke promenljive, tada naredbe

```
   READ(5,10) A,B
10 FORMAT(2L5)
```

dodeljuju vrednosti promenljivim A i B sa jedne kartice, pri čemu u kolonama 1. i 6. mora biti bušen kod slova T ili F u zavisnosti od toga koja konkretna vrednost se želi dodeliti promenljvim A i B. Sadržaj ostalih kolona, od 2. do 4. prvog polja i od 7. do 10. drugog polja, na kartici je bez značaja.

7.2.4.2. Logička naredba

Izračunata vrednost logičkog izraza može se dodeliti logičkoj promenljivoj, pomoću naredbe

$$a = \varphi \qquad (7.2.14)$$

gde je

a - ime logičke promenljive, sa indeksom ili bez njega,

= - simbol FORTRAN-jezika,

φ - logički izraz.

Tako se može pisati

```
DOM=A.AND.(X.OR.Y)
```  (7.2.15)

gde su A, X, Y i DOM logičke promenljive. Izračunavanje logičkog izraza na desnoj strani znaka jednakosti vrši se sleva nadesno. Kako je X.OR.Y zapisano izmedju zagrada, to će se najpre izračunati vrednost ove disjunkcije; označimo ovo sa z, a zatim vrednost konjukcije A.AND.z. Ovako dobijena vrednost logičkih izraza biće dodeljena logičkoj promenljivoj DOM.

7.2.5. Izdavanje vrednosti logičkih promenljivih

Imena logičkih promenljivih, pojedini elementi logičkih nizova ili logički nizovi navode se u listi izlazne naredbe po istim pravilima kao u slučaju brojnih veličina. Medjutim, u odgovarajućoj FORMAT-naredbi navodi se opis (7.2.13), samo što w označava broj simbola u polju izlaznog sloga. Ako se izdaje logička konstanta .TRUE., tada će krajnji desni simbol polja u izlaznom slogu biti slovo T, a ako se izdaje logička konstanta .FALSE., tada će krajnji desni simbol u polju biti slovo F. Svi ostali simboli u polju izlaznog sloga biće znaci blanko.

Tako naredba izalza

```
      WRITE(6,20) A,B
20    FORMAT(' ',2L5)                                    (7.2.16)
```

formira dva polja od po 5 simbola. Prva četiri simbola svakog polja biće znaci blanko, a u zadnjim pozicijama polja biće slovo T, odnosno F u zavisnosti od vrednosti logičkih promenljivih A i B.

Primer

Odrediti vrednost logičkih funkcija

$$\begin{aligned} F_1 &= x_1 \wedge x_2 \vee x_3 \\ F_2 &= \bar{x}_1 \vee x_2 \vee \bar{x}_3 \\ F_3 &= x_1 \wedge \bar{x}_2 \wedge \bar{x}_3 \end{aligned} \qquad (7.2.17)$$

gde su x_2 i x_3 logičke promenljive, a x_1 predikat

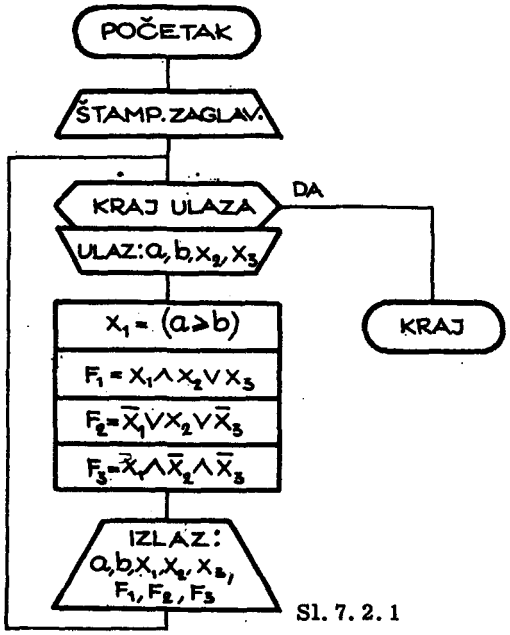
$$x_1 = (a \geq b) \qquad (7.2.18)$$

tj.

$$x_1 = \begin{cases} 0 & \text{ako je } a < b \\ 1 & \text{ako je } a \geq b \end{cases} \qquad (7.2.19)$$

Veličine koje ulaze u algoritam jesu brojevi a i b, i logičke konstante koje predstavljaju vrednosti argumenata x_2 i x_3.

Algoritam za izračunavanje funkcija (7.2.17) prikazan je na sl.7.2.1.



Sl. 7. 2. 1

Program na FORTRAN-jeziku zapisan po algoritmu na sl. 7. 2. 1. ima sledeći izgled

```
      IMPLICIT LOGICAL(X,F)
      DIMENSION X(3),F(3)
      WRITE(6,200)
  200 FORMAT(' ',3X,'A',7X,'B',5X,
     *       'X1    X2    X3    F1    F2    F3'/)
  600 READ(5,300,END=500) A,B,X(2),X(3)
  300 FORMAT(2F6.2,2L1)
      X(1)=A.GE.B
      F(1)=X(1).AND.X(2).OR.X(3)
      F(2)=.NOT.X(1).OR.X(2).OR..NOT.X(3)
      F(3)=X(1).AND..NOT.X(2).AND..NOT.X(3)
      WRITE(6,400) A,B,(X(I),I=1,3),(F(I),I=1,3)
  400 FORMAT(' ',F6.2,2X,F6.2,6L5)
      GO TO 600
  500 STOP
      END
```

Za zadate veličine a, b, x_2 i x_3 rezultati se dobijaju u obliku tabele

A	B	X1	X2	X3	F1	F2	F3
1[illegible]5[illegible]	13.12	F	F	F	F	T	F
40.10	60.00	F	F	T	T	T	F
1.50	22.00	F	T	F	F	T	F
4.80	-1.00	T	F	F	F	T	T
-12.00	-12.00	T	T	T	T	T	F
-12.00	0.0	F	F	F	F	T	F

7.2.6. Naredba prelaza po vrednosti logičkog izraza

U odeljku 7.1.2. videli smo naredbu prelaza po vrednosti predikata. Medjutim, opštiji oblik ove naredbe biće

IF(φ) naredba (7.2.17)

gde sve oznake imaju isto značenje, kao i u slučaju naredbe (7.1.4), samo što se izmedju zagrada može pisati ma kakav logički izraz φ. Dejstvo naredbe (7.2.17) je isto kao i u slučaju naredbe (7.1.4), samo sve ono što se odnosi na predikat p sada se odnosi na logički izraz φ. Zapravo naredba (7.1.4) je poseban slučaj naredbe (7.2.17) kada je logički izraz sastavljen od jednog predikata.

Tako se može pisati

IF(A. AND. B. OR. C. GT. 28.5) GO TO 100

U ovom slučaju promenljive A i B moraju biti deklarisane kao logičke promenljive, a promenljiva C mora biti brojna promenljiva. Izračunavanje vrednosti logičkog izraza u gornjoj naredbi odvija se sledećim redosledom:

a) Odredjuje se vrednost konjukcije A. AND. B koju ćemo označiti sa p_1.

b) Odredjuje se vrednost predikata C. GT. 28.5 koju ćemo označiti sa p_2.

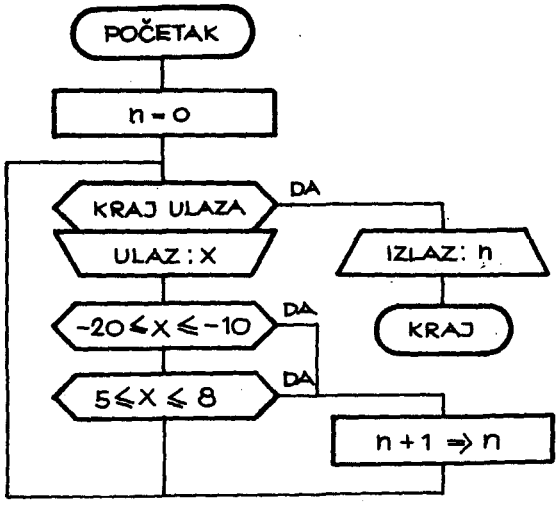
c) Na kraju se odredjuje vrednost logičkog izraza p_1. OR. p_2 koju ćemo označiti sa p.

Dejstvo gornje naredbe je sledeće: ako je vrednost logičkog izraza p istinita prelazi se na naredbu GO TO 100, u suprotnom na naredbu koja sledi iza opisane IF naredbe.

Primer

Zadat je niz brojeva x_i , i=1, 2, ... Svaki od brojeva x_i nalazi se na po jednoj kartici u polju od 1. do 10. kolone kartice sa opisom F10.5. Odrediti koliko od zadatih brojeva x_i leži u intervalu [-20; -10] ili [5, 1; 8, 5] .

Algoritam za rešavanje ovog zadatka je prikazan na sl. 7. 2. 2.



Sl. 7. 2. 2

Program sastavljen po algoritmu na sl. 7. 2. 2. ima sledeći izgled

```
      N=0
   30 READ(5,100,END=20) X
  100 FORMAT(F10.5)
      IF(X.GE.-20.AND.X.LE.-10.OR.
     *X.GE.5.1.AND.X.LE.8.5) N=N+1
      GO TO 30
   20 WRITE(6,40) N
   40 FORMAT(' N=',I8)
      STOP
      END
```

8. ALGORITMI SA REALNIM KONSTANTAMA I PROMENLJIVIM DVOSTRUKE TAČNOSTI

Pri razmatranju algoritama sa realnim konstantama i promenljivim, u glavi 4, podrazumevala se tzv. obična tačnost, tj. registrovanje mešovitog broja u jednom memorijskom registru. Ovakav način registrovanja dozvoljavao je da mešoviti broj sadrži najviše 7 važećih dekadnih cifara.

U mnogim, posebno tehničkim primenama računara, ova tačnost je dovoljna. Medjutim, u nekim naučnim problemima, kao i u izuzetnim tehničkim izračunavanjima, može se zahtevati predstavljanje mešovitih brojeva sa većom tačnošću. Najčešće se uzima povećan broj cifara mešovitih konstanti dva ili više puta veći od obične tačnosti. Kod računara IBM--360/44 registrovanje brojeva u dvostrukoj tačnosti obuhvata 16 važećih dekadnih cifara broja.

8.1. Definicija mešovite konstante dvostruke tačnosti

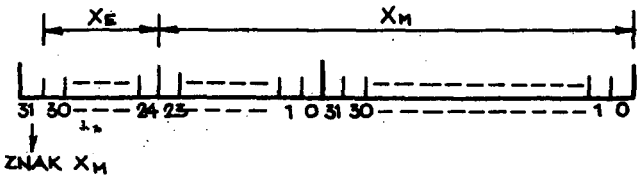
Mešovita konstanta dvostruke tačnosti piše se u jednom od sledeća dva oblika, kao

a) Niz dekadnih cifara, pri čemu se celobrojni i razlomljeni deo razdvajaju decimalnom tačkom. Ispred ovakvog niza može stajati znak + za pozitivne brojeve, a mora stajati znak - za negativne brojeve. Ovako zapisan mešoviti broj može imati najmanje 8, a najviše 16 dekadnih cifara.

b) Oblik kao pod a), sa najmanje jednom cifrom, a najviše 16 dekadnih cifara, iza kojeg se piše slovo D, a zatim se navodi ceo dvocifreni dekadni broj, koji predstavlja eksponent broja 10. Brojna vrednost ovako za-

pisane konstante jednaka je proizvodu brojeva ispred slova D i stepena broja 10, sa celobrojnim eksponentom navedenim iza slova D.

Mešoviti broj dvostruke tačnosti registruje se u memoriji u obliku pokretnog zareza, tako da se eksponent i deo mantise veće težine registruje u jednom registru, kao u slučaju obične tačnosti, a deo manje težine mantise u susednom memorijskom registru (sl. 8. 1. 1). Ovako registrovan



Sl. 8. 1. 1

broj x ima brojnu vrednost

$$x = x_M \cdot 16^{x_E} \qquad (8.1.1)$$

gde za x_E važi relacije (4. 1. 3), a za x_M važi relacija (4. 1. 4), s tim što se u ovom slučaju u mantisi nalazi 14 binarno kodiranih heksadekadnih cifra, što odgovara približno 16 dekadnih cifara.

Primeri

a) Dozvoljeni oblici mešovitih konstanti dvostruke tačnosti

384.157D4
-14.12D-2
15684.32907

b) Nedozvoljeni oblici mešovitih konstanti dvostruke tačnosti

12.004
-5.07D150

8. 2. Definicija realne promenljive dvostruke tačnosti

Ime realna promenljive dvostruke tačnosti definiše se na isti način kao i ime realne promenljive obične tačnosti. Medjutim, da bi se imena ovih promenljivih razlikovala u programu, sva imena promenljivih dvostruke tačnosti moraju biti eksplicitno deklarisana jednom opisnom naredbom. Ova opisna naredba se piše

DOUBLE PRECISION lista (8.2.1)

gde je

DOUBLE PRECISION - službena reč, koja označava opisnu naredbu za deklarisanje realnih promenljivih dvos - truke tačnosti,

lista - spisak imena promenljivih i nizova medju sobom razdvojenih zarezima, koji se u programu deklarišu kao promenljive i nizovi dvos - truke tačnosti.

Ako se ime niza deklariše kao niz čiji su elementi dvostruke tačnosti, tada se u zagradi iza imena niza navodi dimenzija niza na isti način kao u naredbi DIMENSION. Kada je ovakav niz naveden u listi opisne naredbe (8.2.1), ne navodi se i u listi DIMENSION-naredbe. Tako se može pisati

```
DOUBLE PRECISION K,TEM,C(20)
```

što znači da će u programu promenljive K i TEM, kao i niz C koji ima 20 elemenata, biti dvostruke tačnosti. Za svaku promenljivu, sa indeksom ili bez indeksa dvostruke tačnosti, u memoriji računara biće angažovana dva registra.

Treba uočiti da opisnom naredbom (8.2.1) eksplicitno se deklarišu imena promenljivih navedena u listi kao realne promenljive dvostruke tačnosti. Tako u gornjem primeru promenljiva K, koja je po unutrašnjoj konvenciji celoborjna, deklariše se kao realna i to dvostruke tačnosti.

Opisnom naredbom (8.2.1) deklarišu se i imena funkcijskih naredbi i potprograma ako se želi rezultat potprograma u dvostrukoj tačnosti.

Opisna naredba (8.2.1) se piše na početku programa, pre prve izvršne naredbe programa. Redosled pisanja opisnih naredbi na početku programa je sledeći:

1. Naredbe za eksplicitnu deklaraciju vrste promenljivih (REAL, INTEGER, DOUBLE PRECISION, LOGICAL),
2. Naredba za implicitnu deklaraciju vrste promenljive (IMPLICIT),
3. Naredba za navodjenje imena potprograma koji se javljaju kao argumenti drugih potprograma (EXTERNAL),

4. Naredba za definiciju dimenzija nizova (DIMENSION),
5. Funkcijske naredbe, i
6. Ostale naredbe programa.

8.3. Dodeljivanje brojnih vrednosti realnim promenljivim dvostruke tačnosti

8.3.1. Aritmetička naredba

Na isti način na koji se običnim promenljivim dodeljuje brojna vrednost aritmetičkom naredbom, može se i promenljivoj dvostruke tačnosti dodeliti brojna vrednost. U ovom slučaju aritmetička naredba ima oblik

$$a = \psi \qquad (8.3.1)$$

gde je

a - ime promenljive dvostruke tačnosti,

ψ - aritmetički izraz.

Ranije smo videli da ako je jedan od argumenata neke aritmetičke operacije u aritmetičkom izrazu ψ, u obliku pokretnog zareza, a drugi celobrojni, tada će medjurezultat ove operacije biti u pokretnom zarezu. Medjutim, sada argumenti aritmetičkog izraza ψ mogu biti veličine u pokretnom zarezu obične i dvostruke tačnosti i celobrojne veličine. U ovakvim aritmetičkim izrazima medjurezultati se javljaju u obliku dvostruke tačnosti, ako je barem jedan argument dvostruke tačnosti. Medjutim, ako aritmetički izraz ψ ne sadrži argumente dvostruke tačnosti, tada će izračunata vrednost aritmetičkog izraza biti prevedena u oblik dvostruke tačnosti i dodeljena promenljivoj a u (8.3.1).

8.3.2. Naredba ulaza

Kada se promenljivoj dvostruke tačnosti dodeljuje brojna vrednost sa ulaza, tada se u ulaznom slogu mora opisati polje koje sadrži konstantu dvostruke tačnosti. Ovaj opis polja piše se u obliku

$$nDk.d \qquad (8.3.2)$$

gde je

n - neoznačen ceo broj, koji ukazuje na broj ponavljanja opisa D,

D - simbol FORTRAN-jezika, koji ukazuje da polje sadrži konstantu dvostruke tačnosti,

k - neoznačen ceo broj, koji ukazuje na broj kolona polja na kartici koje sadrži konstantu dvostruke tačnosti,

d - neoznačen ceo broj, koji ukazuje na broj decimalnih mesta konstante dvostruke tačnosti,

Opis (8.3.2) navodi se na odgovarajućem mestu FORMAT-naredbe. Konstanta dvostruke tačnosti, sa odgovarajućeg polja kartice, dodeljuje se odgovarajućoj promenljivoj dvostruke tačnosti u listi naredbe ulaza. Ovakva konstanta registruje se u dva memorijska registra, kako je to prikazano na sl. 8.1.1.

Pored opisa (8.3.2) može se koristiti i opis

nFk.d (8.3.3)

gde je značenje simbola isto kao i u slučaju opisa konstante obične tačnosti. Oblik (8.3.2) i (8.3.3) ne razlikuju se bitno kada se radi o naredbi ulaza. U oba slučaja konstanta sa ulaza dodeljuje se promenljivoj u listi i registruje se na isti način u memoriji računara. Oblik (8.3.2) je pogodan za zapis vrlo malih i velikih brojnih vrednosti, dok je u svim drugim slučajevima pogodnije koristiti oblik (8.3.3).

8.4. Izdavanje brojnih vrednosti promenljivih dvostruke tačnosti

Za izdavanje brojnih vrednosti promenljivih dvostruke tačnosti može se koristiti jedan od navedenih opisa (8.3.2) ili (8.3.3). Kao i u slučaju obične tačnosti, opis (8.3.2) je pogodan kada se radi o vrlo malim ili vrlo velikim brojnim vrednostima, dok je u svim drugim slučajevima pogodniji opis (8.3.3). Ovo iz razloga što je eksponencijalni oblik broja nepogodniji za čitanje nego oblik sa fiksnim brojem celih i decimalnih mesta. U slučaju dvostruke tačnosti eksponencijalni oblik se izdaje sa slovom D umesto slova E koje se koristi za običnu tačnost.

Primer

Sastaviti funkcijski potprogram koji izračunava sa dvostrukom tač - nošću

$$\sin \frac{\pi}{4} x \approx \sum_{i=0}^{5} a_{2i+1}\, x^{2i+1} \qquad (8.3.4)$$

za $|x| \leq 1$, gde je

$a_1 = 0,785\ 398\ 163\ 397\ 426\ 5$

$a_3 = -0,080\ 745\ 512\ 187\ 669\ 4$

$a_5 = 0,002\ 490\ 394\ 565\ 299\ 5$

$a_7 = -0,000\ 036\ 576\ 187\ 395\ 3$

$a_9 = 0,000\ 000\ 313\ 333\ 683\ 3$

$a_{11} = -0,000\ 000\ 001\ 734\ 798\ 7$

Izračunavanje po formuli (8.3.4) daje vrednosti sinusne funkcije sa relativnom greškom

$$\varepsilon < 16.10^{-16} \qquad (8.3.5)$$

Algoritam potprograma prikazan je na sl. 8.3.1.

Brojne vrednosti konstanata a_{2i+1} dodeljuju se elementima niza b_{i+1}, $i = 0, 1, \ldots, 4$, a zatim izračunava

$$y = x^2$$

da bi se u ciklusu odredila vrednost polinoma

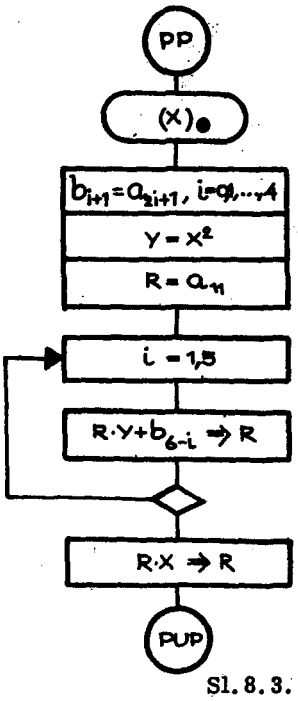
$$R = b_1 + y\left\{b_2 + y\left[b_3 + y\,(b_4 + y\,(b_5 + b_6\, y))\right]\right\} \qquad (8.3.6)$$

što odgovara polinomu

$$R = a_1 + x^2\left\{a_3 + x^2\left[a_5 + x^2\,(a_7 + x^2\,(a_9 + a_{11}\, x^2))\right]\right\} \qquad (8.3.7)$$

Po izlasku iz ciklusa dobijena vrednost R se množi sa x, čime je odredjena vrednost aproksimacionog polinoma (8.3.4) za datu vrednost x.

Funkcijski potprogram sastavljen prema algoritmu na sl. 8.3.1 ima sledeći izgled



Sl. 8. 3. 1

```
      FUNCTION SINP4X(X)
      DOUBLE PRECISION SINP4X,B(5),X,Y
      B(1)=0.7853981633974265
      B(2)=-8.07455121876694D-2
      B(3)=2.4903945652995D-3
      B(4)=-3.657618739530-5
      B(5)=3.1333368330-7
      Y=X*X
      SINP4X=-1.73479870-9
      DO 10 I=1,5
   10 SINP4X=SINP4X*Y+B(6-I)
      SINP4X=SINP4X*X
      RETURN
      END
```

Program koji za zadat ugao u radijanima (x), izračunava ugao $(\pi/4)\cdot x$ i $\sin((\pi/4)\cdot x)$, ima sledeći izgled:

```
      DOUBLE PRECISION X,Y,ALFA,SINP4X
      WRITE(6,50)
   50 FORMAT(' ',4X,'X',7X,'UGAO U RADIJANIMA',14X,'SINUS'/)
  300 READ(5,100,END=101) X
  100 FORMAT(F8.5)
      Y=SINP4X(X)
      ALFA=3.1415926535897932/4.*X
      WRITE(6,200) X,ALFA,Y
  200 FORMAT(' ',F8.5,F22.16,D26.16)
      GO TO 300
  101 STOP
      END
```

Rezultati se štampaju u obliku tabele

X	UGAO U RADIJANIMA	SINUS
1.00000	0.7853981633974480	0.70710678118654500 00
-0.40000	-0.3141592653589792	-0.30901699437494870 00
0.0	0.0	0.0
-0.00001	-0.0000078539816340	-0.7853981633893519D-05
0.52000	0.4084070449666730	0.39714789063477930 00

8.5. Izračunavanje elementarnih funkcija sa dvostrukom tačnošću

Ako se proračun izvodi sa dvostrukom tačnošću, tada je potrebno i sve elementarne funkcije, koje se javljaju u proračunu, izračunati sa dvostrukom tačnošću. Već je rečeno da se u slučaju obične tačnosti ove funkcije računaju preko funkcijskih potprograma sa propisanim imenima, koji se automatski pozivaju pri prevodjenju programa sa FORTRAN-jezika na mašinski jezik. Potprogrami koji izračunavaju elementarne funkcije sa dvostrukom tačnošću imaju takodje propisana imena, a prvo slovo ovih imena je uvek D, što ukazuje da se radi o potprogramu sa dvostrukom tačnošću. U tabeli 8.5.1. dat je spisak propisanih imena ovih potprograma.

Pored funkcijskih potprograma navedenih u tabeli 8.5.1, u FORTRAN-jeziku postoji i izvestan broj funkcija koje se pišu na isti način kao i funkcijski potprogrami, ali se ne javljaju jedanput u programu na mašinskom jeziku, već onoliko puta koliko puta su zapisane u programu. To znači da se ova funkcija zamenjuje nizom mašinskih naredbi na svakom mestu programa na kojem je zapisana. Spisak ovih funkcija sa propisanim imenima dat je u tabeli 8.5.2.

U tabelama 8.5.1 i 8.5.2 uvedene su sledeće oznake:

$x, x_1, x_2, \ldots$ - aritmetički izrazi,

R - realna veličina koja se registruje u obliku pokretnog zareza,

C - celobrojna veličina, koja se registruje u obliku celog broja,

M - maksimalna vrednost celog broja (M=2 147 483 647),

P - maksimalna vrednost broja registrovanog u obliku pokretnog zareza ($P \approx 7,2 \cdot 10^{75}$),

[y] - celobrojni deo broja y,

DR - realna veličina koja se registruje u obliku pokretnog zareza dvostruke tačnosti,

DP - maksimalna vrednost broja registrovanog u obliku pokretnog zareza dvostruke tačnosti ($DP \approx 7,2 \cdot 10^{75}$).

Tabela 8.5.1.

Način pisanja		Argumenti		Funkcija		Opis
U Fortranu	U matematici	Vrsta	Ograničenje	Vrsta	Relativna greška je manja od	
DEXP(x)	e^x	DR	$x \leqslant 174{,}673$	DR	$2{,}33\cdot10^{-15}$	Eksponencijalna funkcija
DLOG(x)	$\ln x$	DR	$x > 0$	DR	$3{,}31\cdot10^{-16}$	Prirodni logaritam
DLOG10(x)	$\log x$	DR	$x > 0$	DR	$6{,}14\cdot10^{-16}$	Dekadni logaritam
DSQRT(x)	$\sqrt{x}$	DR	$x \geqslant 0$	DR	$1{,}08\cdot10^{-16}$	Kvadratni koren
DSIN(x)	sin(x)	DR	$\lvert x\rvert < 0{,}35\cdot10^{16}$	DR	$4{,}08\cdot10^{-16}$	Trigonometrijske funkcije
DCOS(x)	cos(x)	DR	$\lvert x\rvert < 0{,}35\cdot10^{16}$	DR	$4{,}08\cdot10^{-16}$	
DTAN(x)	tg(x)	DR	$\lvert x\rvert < 0{,}35\cdot10^{16}$, $\lvert x\rvert \neq (k+1/2)\pi$, $k=0,1,\ldots$	DR	$3{,}79\cdot10^{-12}$	
DCOTAN(x)	ctg(x)	DR	$\lvert x\rvert < 0{,}35\cdot10^{16}$, $\lvert x\rvert \neq k\pi$, $k=0,1,\ldots$	DR	$8{,}61\cdot10^{-13}$	
DARSIN(x)	arcsin(x)	DR	$\lvert x\rvert \leqslant 1$	DR	$2{,}40\cdot10^{-16}$	Inverzne trigonometrijske funkcije
DARCOS(x)	arccos(x)	DR	$\lvert x\rvert \leqslant 1$	DR	$2{,}72\cdot10^{-16}$	
DATAN(x)	arctg(x)	DR	$\lvert x\rvert \leqslant P$	DR	$2{,}08\cdot10^{-16}$	
DATAN2(x_1,x_2)	arctg(x_1/x_2)	DR	$\lvert x_1\rvert, \lvert x_2\rvert \leqslant P$ osim $x_1=x_2=0$	DR	$2{,}08\cdot10^{-16}$	
DSINH(x)	sinh(x)	DR	$\lvert x\rvert < 174{,}673$	DR	$3{,}59\cdot10^{-16}$	Hiperboličke funkcije
DCOSH(x)	cosh(x)	DR	$\lvert x\rvert < 174{,}673$	DR	$4{,}81\cdot10^{-16}$	
DTANH(x)	tgh(x)	DR	$\lvert x\rvert \leqslant P$	DR	$4{,}45\cdot10^{-17}$	
DMAX1(x_1,x_2,...)	max(x_1,x_2,...)	DR	$\lvert x_1\rvert, \lvert x_2\rvert, \ldots \leqslant P$	DR	-	Nalaženje najveće vrednosti
DMIN1(x_1,x_2,...)	min(x_1,x_2,...)	DR	$\lvert x_1\rvert, \lvert x_2\rvert, \ldots \leqslant P$	DR	-	Nalaženje najmanje vrednosti
DERF(x)	$\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$	DR	$\lvert x\rvert \leqslant P$	DR	$1{,}70\cdot10^{-16}$	Funkcija greške
DERFC(x)	$1-\frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt$	DR	$\lvert x\rvert \leqslant P$	DR	$5{,}02\cdot10^{-16}$	Komplement funkcije greške
DGAMMA(x)	$\int_0^\infty e^{-t}t^{x-1}dt$	DR	$0{,}13\cdot10^{-75} < x < 57{,}5744$	DR	$6{,}2\cdot10^{-14}$	Gama-funkcija
DLGAMA(x)	$\ln\int_0^\infty e^{-t}t^{x-1}dt$	DR	$0 < x < 4{,}2913\cdot10^{73}$	DR	$2{,}38\cdot10^{-15}$	Logaritam gama-funkcije

Tabela 8.5.2.

Način pisanja		Argumenti		Funkcija		Opis
U Fortranu	U matematici	Vrsta	Ograničenje	Vrsta	Relativna greška je manja od	
DFLOAT(x)	-	C	$\lvert x\rvert \leqslant M$	DR	-	Prevodjenje iz oblika celog broja u oblik pokretnog zareza i obratno
IDINT(x)	$[x]$	DR	$\lvert x\rvert \leqslant M$	C	-	
SNGL(x)	-	DR	$\lvert x\rvert \leqslant DP$	R	-	Prevodjenje iz oblika dvostruke tačnosti u običnu tačnost i obratno
DBLE(x)	-	R	$\lvert x\rvert \leqslant P$	DR	-	
DSIGN(x_1,x_2)	$\lvert x_1\rvert \operatorname{sign} x_2$	DR	$\lvert x_1\rvert, \lvert x_2\rvert \leqslant DP$	DR	-	Prenošenje znaka
DMOD(x_1,x_2)	$x_1-\left[x_1/x_2\right]\cdot x_2$	DR	$\lvert x_1\rvert, \lvert x_2\rvert \leqslant DP$	DR	-	Modularna aritmetika
DABS(x)	$\lvert x\rvert$	DR	$\lvert x\rvert \leqslant DP$	DR	-	Apsolutna vrednost

Navedena relativna greška funkcija, u tabeli 8.5.1, predstavlja najveću statistički dobijenu relativnu grešku za razne vrednosti argumenta iz dozvoljenog intervala.

Primer

Uzmimo primer naveden na kraju odeljka 4.9, s tim što ćemo sada izračunavanje elementarnih funkcija izvršiti preko potprograma sa dvostrukom tačnošću. U ovom zadatku treba izračunati

$$y_1 = 1-e^{-x}\sin 2x + \log(\cos^2 x)\cdot \operatorname{tg} x$$

$$y_2 = \arcsin\left(\frac{x}{100}\right) + \ln|x|\cdot \operatorname{arctg} x$$

$$y_3 = \sqrt{|1-\operatorname{tgh} x|} + \sinh x - 2\cosh x$$

za zadatih 7 vrednosti argumenta x.

FORTRAN-program u slučaju korišćenja potprograma sa dvostrukom tačnošću ima sledeći izgled:

```
      DOUBLE PRECISION X,Y1,Y2,Y3
      WRITE(6,100)
  100 FORMAT('1',4X,'X',14X,'Y1',15X,'Y2',15X,'Y3'/)
      DO 101 I=1,7
      READ(5,200) X
  200 FORMAT(F8.4)
      Y1=1.-DEXP(-X)*DSIN(2.*X)+DLOG10(DCOS(X)**2)*DTAN(X)
      Y2=DARSIN(X/100.)+DLOG(DABS(X))*DATAN(X)
      Y3=DSQRT(DABS(1.-DTANH(X)))+DSINH(X)-2.*DCOSH(X)
  101 WRITE(6,300) X,Y1,Y2,Y3
  300 FORMAT(' ',F8.4,2X,3D17.7)
      STOP
      END
```

Izlazni rezultati su štampani u obliku tabele:

X	Y1	Y2	Y3
-75.3427	-0.5828197D 32	-0.7584952D 01	-0.7888785D 33
-34.2885	-0.3989881D 15	-0.5799392D 01	-0.1167678D 16
-28.0032	-0.7487762D 12	-0.5399274D 01	-0.2176339D 13
-2.3410	-0.9710959D 01	-0.1016114D 01	-0.1422784D 02
0.5684	0.3911757D 00	-0.2863057D 00	-0.1035361D 01
5.2028	0.2229444D 01	0.2329440D 01	-0.9089073D 02
17.3333	0.4728656D 02	0.4490720D 01	-0.1685491D 08

Razlike izmedju ovih rezultata i rezultata u primeru na kraju odeljka 4.9, nastaju zbog izračunavanja elementarnih funkcija na 16 važećih cifara, prema 7 važećih cifara kada se radi sa običnom tačnošću.

9. ALGORITMI SA KOMPLEKSNIM KONSTANTAMA I PROMENLJIVIM

U nekim oblastima naučnih i tehničkih izračunavanja javljaju se kompleksne veličine i operacije sa ovakvim veličinama. Kako se aritmetičke operacije sa kompleksnim veličinama svode na aritmetičke operacije sa realnim veličinama, pri čemu se vodi računa o realnom i imaginarnom delu kompleksne veličine, to se ovakve operacije mogu programirati pomoću naredbi u FORTRAN-jeziku koje operišu sa realnim veličinama. Medjutim, kako jedna aritmetička operacija nad kompleksnim argumentima zahteva veći broj aritmetičkih operacija nad realnim veličinama, to je programiranje aritmetičkih operacija nad kompleksnim veličinama vrlo zametan posao. Tako, ako su z_1 i z_2 kompleksni brojevi

$$z_1 = a + bi \qquad (9.1)$$
$$z_2 = c + di$$

gde su a, b, c i d realne veličine, a $i = \sqrt{-1}$, tada je

$$z_1 + z_2 = (a + c) + (b + d)\, i$$
$$z_1 - z_2 = (a - c) + (b - d)\, i \qquad (9.2)$$
$$z_1 \cdot z_2 = (ac - bd) + (ad + bc)i$$
$$\frac{z_1}{z_2} = \frac{ac+bd}{c^2+d^2} + \frac{bc-ad}{c^2+d^2}\, i$$

Kao što se vidi, aritmetička operacija deljenja kompleksnih veličina zahteva dve operacije stepenovanja, četiri operacije množenja i četiri operacije sabiranja, odnosno oduzimanja realnih veličina. Da bi se izbeglo

programiranje aritmetičkih operacija nad kompleksnim veličinama u FORTRAN-jeziku, postoji mogućnost rada sa kompleksnim veličinama bez razdvajanja realnog i imaginarnog dela ovih veličina.

9.1. Definicija kompleksne konstante

Kompleksna konstanta se piše u obliku

$$(a, b) \qquad (9.1.1)$$

gde su a i b mešovite konstante u FORTRAN-jeziku. Konstanta (9.1.1) u matematičkoj notaciji predstavlja kompleksni broj

$$a + bi \qquad (9.1.2)$$

Ako je realni ili imaginarni deo jednak nuli, i tada mora biti naveden u obliku (9.1.1). Za registrovanje kompleksne konstante (9.1.1) u memoriji računara se koriste dva registra. U jednom registru se nalazi realni deo kompleksne konstante, a u drugom imaginarni deo. Oba dela se registruju u obliku pokretnog zareza.

Primeri

a) Dozvoljeni oblici kompleksnih konstanti

(1.4, -3.2)
(0.0, 45.)
(0.0, 0.0)
(12.3E-2, 5.4)

b) Nedozvoljeni oblici kompleksnih konstanti

(A, B)
(A, 4.0)
(12.4E-3)

9.2. Definicija kompleksne promenljive

Ime kompleksne promenljive definiše se na isti način kao i ime realne promenljive. Medjutim, da bi se imena ovih promenljivih razlikovala,

imena kompleksnih promenljivih moraju biti deklarisana kao kompleksne promenljive. Ova deklaracija može biti eksplicitna, pomoću opisne naredbe

COMPLES lista (9.2.1)

gde je

COMPLEX - službena reč kojom se deklarišu imena kompleksnih promenljivih,

lista - spisak imena promenljivih, medju sobom razdvojenih zarezima, koja se deklarišu kao kompleksne promenljive.

Tako, opisna naredba

COMPLEX X,Y,NAPON,A(10) (9.2.2)

deklariše promenljive X, Y i NAPON kao kompleksne promenljive i niz A, kao niz sa 10 elemenata, od kojih je svaki kompleksni broj.

Pored eksplicitne deklaracije (9.2.1) kompleksnih promenljivih, mogu se ove promenljive deklarisati implicitno opisnom naredbom

IMPLICIT lista (9.2.3)

gde je

IMPLICIT - službena reč, a

lista - spisak elemenata, medju sobom razdvojenih zarezima.

Element liste (9.2.3) je oblika

vrsta ($lista_1$) (9.2.4)

gde je

vrsta - službena reč COMPLEX, REAL, INTEGER ili LOGICAL,

$lista_1$ - spisak velikih slova engleske azbuke, medju sobom razdvojena zarezima.

Implicitnom deklaracijom (9.2.3) deklarišu se sva imena promenljivih koja počinju jednim od navedenih slova izmedju zagrada u (9.2.4), kao promeljive odredjene vrste, a prema tome koja službena reč stoji namesto reči vrsta u (9.2.4).

Tako opisna naredba

IMPLICIT LOGICAL(A,S),COMPLEX(T,R,Z) (9.2.5)

ima sledeće značenje: promenljive čija imena počinju slovima A i S, deklarišu se kao logičke promenljive, a promenljive čija imena počinju slovima T, R ili Z, kao kompleksne promenljive.

Ako slova koja se navode izmedju zagrada u (9.2.4) slede u azbučnom redu, tada se može pisati samo prvo i zadnje slovo izmedju kojih se stavlja povlaka (-). Tako naredba

```
IMPLICIT COMPLEX(D-H,T)                    (9.2.6)
```

deklariše promenljive čija imena počinju slovima D, E, F, G, H i T kao kompleksne promenljive.

Opisne naredbe za deklarisanje vrste promenljivih pišu se na početku programa pre svih drugih opisnih naredbi, odnosno pre prve izvršne naredbe programa ako drugih opisnih naredbi nema, i to tako da se najpre navodi eksplicitna, a zatim implicitna opisna naredba za deklarisanje vrste promenljivih.

Za svaku kompleksnu promenljivu u memoriji računara biće rezervisana dva registra, za registrovanje realnog i imaginarnog dela promenljive, kao brojeva u pokretnom zarezu.

Opisnim naredbama (9.2.1) i (9.2.3) deklarišu se i imena funkcijskih naredbi i potprograma, ako je rezultat ovih potprograma kompleksni broj.

9.3. Dodeljivanje vrednosti kompleksnim promenljivim

9.3.1. Aritmetička naredba

Kompleksnoj promenljivoj može se dodeliti vrednost pomoću aritmetičke naredbe

$$a = \psi \qquad (9.3.1)$$

gde je

ψ - aritmetički izraz, a

a - ime kompleksne promenljive.

Aritmetički izraz ψ može kao argumente imati celobrojne, realne i kompleksne konstante i promenljive. Rezultat aritmetičke operacije izmedju dva argumenta, od kojih je bar jedan kompleksan, biće uvek kompleksna konstanta. Izračunata vrednost aritmetičkog izraza ψ, kao kompleksna

konstanta, dodeljuje se kompleksnoj promenljivoj a na levoj strani znaka jednakosti. Ako je vrednost aritmetičkog izraza ψ realan broj, tada će biti formirana kompleksna konstanta, čiji će realan deo biti vrednost aritmetičkog izraza, a imaginarni deo će biti nula. Ovakva kompleksna konstanta biće dodeljena promenljivoj a .

U aritmetičkom izrazu ψ mogu se koristiti četiri aritmetičke operacije izmedju celobrojnih, realnih i kompleksnih konstanti i promenljivih.

Može se koristiti i operacija stepenovanja, pri čemu u slučaju da je osnova kompleksna veličina, stepen može biti samo celobrojna veličina.

9.3.2. Naredba ulaza

Kada se kompleksnoj promenljivoj dodeljuje vrednost sa ulaza, tada se u ulaznom slogu pojavljuju dva polja koja sadrže dve realne konstante. Prvo polje sadrži realnu konstantu koja se uzima kao realan deo kompleksne promenljive, a drugo polje sadrži realnu konstantu koja se uzima kao imaginaran deo kompleksne promenljive. Ova polja se opisuju pomoću opisa za mešovite konstante (F ili E opis).

Tako, ako kompleksne promenljive X i Y, i celobrojna promeljiva J dobijaju brojne vrednosti sa ulaza, tada se može pisati

```
      READ(5,100) X,Y,J
100   FORMAT(2(F8.3,E12.5),I4)
```

U ovom slučaju na jednoj kartici je opisano 5 polja:

- prvo polje od 8 kolona, sa opisom F8.3, sadrži realni deo kompleksne promenljive X,
- drugo polje od 12 kolona, sa opisom E12.5, sadrži imaginarni deo kompleksne promenljive X,
- treće polje od 8 kolona, sa opisom F8.3, sadrži realni deo kompleksne promenljive Y,
- četvrto polje od 12 kolona, sa opisom E12.5, sadrži imaginarni deo kompleksne promenljive Y, i
- peto polje od 4 kolone, sa opisom I4, sadrži ceo broj koji se dodeljuje promenljivoj J.

9.4. Izdavanje vrednosti kompleksnih promenljivih

Za izdavanje vrednosti kompleksnih promenljivih važi slično kao i za dodeljivanje vrednosti kompleksnim promenljivim sa ulaza. I ovde se u listi izlazne naredbe navodi ime kompleksne promenljive, a u izlaznom slogu opisuju se dva polja koja sadrže realne brojeve. Prvo polje sadrži realni deo, a drugo imaginarni deo kompleksne promenljive. Ova polja se opisuju opisima za mešovite konstante (F ili E opis).

Tako ako se želi izdati vrednost kompleksne promenljive Z, može se pisati

```
      WRITE(6,200) Z
200   FORMAT(' ',E11.4,3X,F8.4)
```

U ovom slučaju izlazni slog sadrži:

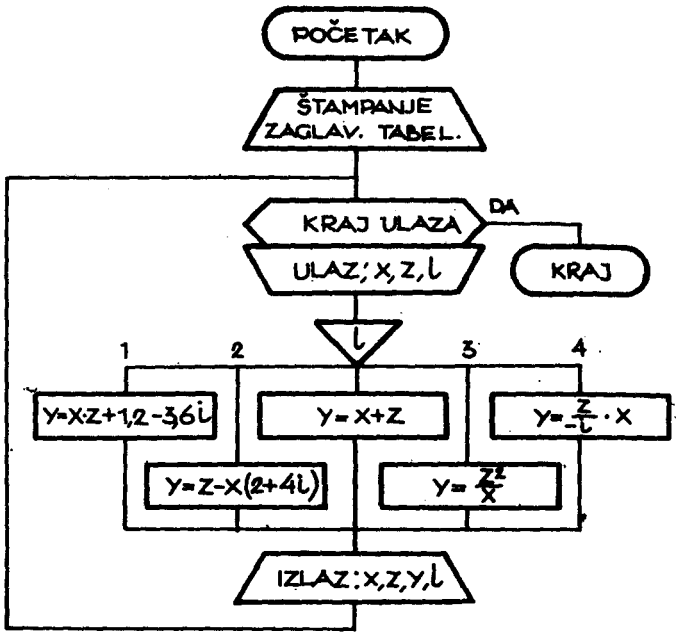
- jedan blanko simbol (novi red na štampaču),
- polje od 11 simbola, sa opisom E11.4, koje se odnosi na realni deo kompleksne promenljive Z,
- polje od 3 medjuprostora, sa opisom 3X, i
- polje od 8 simbola, sa opisom F8.4, koje se odnosi na imaginarni deo kompleksne promenljive Z.

Primer

Zadat je realan broj x, kompleksni broj z i jednocifreni ceo broj ℓ. Izračunati kompleksni broj y po jednoj od formula

$$y = \begin{cases} xz+1,2-3,6i.. & \text{ako je } \ell = 1 \\ z-x(2+4i) \ \ldots & \text{ako je } \ell = 2 \\ \frac{z^2}{x} \ \ldots\ldots\ldots & \text{ako je } \ell = 3 \\ \frac{z}{-i} \cdot x \ \ldots\ldots & \text{ako je } \ell = 4 \\ x+z \ \ldots\ldots\ldots & \text{ako je } \ell \neq 1,2,3,4 \end{cases}$$

Program sastaviti tako da se brojevi x, z i ℓ nalaze na jednoj kartici i da ovakvih kartica može biti proizvoljan broj. Algoritam za ovo izračunavanje prikazan je na sl. 9.4.1.



Sl. 9.4.1

Program na FORTRAN-jeziku, sastavljen po algoritmu na sl. 9.4.1. ima sledeći izgled:

```
      COMPLEX Z,Y
      WRITE(6,200)
  200 FORMAT(' ',3X,'X',10X,'Z',22X,'Y',16X,'L'/
     *' ',8X,'REAL.  IMAG.',8X,'REAL.',9X,
     *'IMAG.'/)
  400 READ(5,300,END=500) X,Z,L
  300 FORMAT(F5.1,F6.2,F5.2,I1)
      GO TO (1,2,3,4),L
      Y=X+Z
  401 WRITE(6,201) X,Z,Y,L
  201 FORMAT(' ',F5.1,F8.2,F7.2,1X,2E15.6,I4)
      GO TO 400
    1 Y=X*Z+(1.2,-3.6)
      GO TO 401
    2 Y=Z-X*(2.0,4.0)
      GO TO 401
    3 Y=Z*Z/X
      GO TO 401
    4 Y=Z/(0.0,-1.0)*X
      GO TO 401
  500 STOP
      END
```

Rezultati su štampani u obliku tabele

X	Z		Y		L
	REAL.	IMAG.	REAL.	IMAG.	
25.2	36.78	-1.26	0.536183E 02	-0.367800E 01	3
2.1	0.0	4.00	0.210000E 01	0.400000E 01	5
1.0	2.20	-4.10	0.420000E 01	-0.999994E-01	2
2.0	1.00	1.00	0.0	0.100000E 01	3
6.0	2.00	3.00	0.132000E 02	0.144000E 02	1
1.5	8.00	-2.00	0.300000E 01	0.120000E 02	4

9.5. Kompleksne veličine dvostruke tačnosti

Već smo videli da se kompleksna veličina u računaru prikazuje pomoću dva realna broja, koja se registruju u obliku pokretnog zareza. Prvi od ovih brojeva predstavlja realni, a drugi imaginarni deo kompleksnog broja. Kako se brojevi u pokretnom zarezu mogu u računaru registrovati i u obliku dvostruke tačnosti (glava 8), to se i kompleksne konstante i promenljive mogu pojaviti u obliku dvostruke tačnosti.

Kompleksna konstanta dvostruke tačnosti piše se u obliku

(a, b) (9.5.1)

gde su a i b mešovite konstante dvostruke tačnosti u FORTRAN-jeziku.

Ime kompleksne promenljive deklariše se pomoću eksplicitne naredbe za deklarisanje vrste promenljivih. Medjutim, ako se želi da deklarisane kompleksne promenljive budu dvostruke tačnosti, tada opisna naredba ima oblik

COMPLEX * 16 lista (9.5.2)

gde broj 16 ukazuje da se 16 podregistara koristi za registrovanje brojne vrednosti kompleksne promenljive. Kako svaki registar sadrži 4 podregistra, to znači da će 4 registra u memoriji biti angažovano za registrovanje kompleksne promenljive dvostruke tačnosti, dva za realni i dva za imaginarni deo kompleksnog broja. Važno je uočiti da se dvostruka tačnost kompleksnih promenljivih ne može deklarisati pomoću naredbe DOUBLE PRECISION, jer se u listi ove naredbe mogu navesti samo imena realnih promenljivih.

Tako se ne može pisati

```
COMPLEX A,B
DOUBLE PRECISION A,B
```

već

```
COMPLEX*16 A,B
```

O zadavanju broja podregistara za registrovanje promenljivih, u opisnim naredbama, biće reči u odeljku 10.1.

Kompleksnoj promenljivoj dvostruke tačnosti može se dodeliti vrednost pomoću aritmetičke naredbe

$$a = \Psi \qquad (9.5.2)$$

gde je a ime promenljive deklarisane kao kompleksna promenljiva dvostruke tačnosti. Aritmetički izraz Ψ može imati za argumente celobrojne konstante i promenljive, kao i realne i kompleksne konstante i promenljive obične ili dvostruke tačnosti. Rezultat aritmetičke operacije izmedju dva argumenta, od kojih je bar jedan kompleksna veličina dvostruke tačnosti, biće uvek kompleksna konstanta dvostruke tačnosti. Izračunata vrednost aritmetičkog izraza Ψ, kao kompleksna konstanta dvostruke tačnosti dodeljuje se promenljivoj a na levoj strani znaka jednakosti u naredbi (9.5.2).

Dodeljivanje vrednosti kompleksnoj promenljivoj dvostruke tačnosti sa ulaza, vrši se na sličan način kao i u slučaju kompleksne promenljive obi obične tačnosti, samo što se u slučaju dvostruke tačnosti za opis polja u ulaznom slogu koristi opis mešovitih konstanti dvostruke tačnosti. Isto važi i u slučaju izdavanja vrednosti kompleksnih promenljivih dvostruke tačnosti.

Primer

Primer na kraju odeljka 9.4. rešiti sa dvostrukom tačnošću. U ovom slučaju FORTRAN-program ima sledeći izgled

```
      COMPLEX*16 Z,Y
      DOUBLE PRECISION X
      WRITE(6,200)
  200 FORMAT(' ',3X,'X',10X,'Z',22X,'Y',16X,'L'/
     *' ',8X,'REAL.  IMAG.',8X,'REAL.',9X,
     *'IMAG.'/)
  400 READ(5,300,END=500) X,Z,L
  300 FORMAT(F5.1,F6.2,F5.2,I1)
      GO TO (1,2,3,4),L
      Y=X+Z
```

```
401 WRITE(6,201) X,Z,Y,L
201 FORMAT(' ',F5.1,F8.2,F7.2,1X,2E15.6,I4)
    GO TO 400
  1 Y=X*Z+(1.2D0,-3.6D0)
    GO TO 401
  2 Y=Z-X*(2.0D0,4.0D0)
    GO TO 401
  3 Y=Z*Z/X
    GO TO 401
  4 Y=Z/(0.0D0,-1.0D0)*X
    GO TO 401
500 STOP
    END
```

Rezultati se dobijaju u obliku tabele

X	Z		Y		L
	REAL.	IMAG.	REAL.	IMAG.	
25.2	36.78	-1.26	0.536183D 02	-0.367800D 01	3
2.1	0.0	4.00	0.210000D 01	0.400000D 01	5
-1.0	2.20	-4.10	0.420000D 01	-0.100000D 00	2
2.0	1.00	1.00	0.0	0.100000D 01	3
6.0	2.00	3.00	0.132000D 02	0.144000D 02	1
1.5	8.00	-2.00	0.300000D 01	0.120000D 02	4

Razlike u rezultatima, kada je primer rešavan sa običnom tačnošću (na kraju odeljka 9.4), i prikazanih rezultata u slučaju dvostruke tačnosti, potiču iz sledećih razloga.

- vrednosti promenljivih x i z, u slučaju dvostruke tačnosti, biće prevedene sa većom tačnošću iz dekadnog brojnog sistema u binarno kodirani heksadekadni brojni sistem u pokretnom zarezu,

- vrednosti konstanti koje se javljaju u programu takodje su sa većom tačnošću registrovane u memoriji,

- izračunavanje po navedenim formulama izvršava se sa većim brojem važećih cifara argumenata, i medjurezultati pojedinih aritmetičkih operacija, ako je jedan argument ovih operacija dvostruke tačnosti, biće takodje dvostruke tačnosti.

Sve ovo čini tačnijim proračun izveden sa dvostrukom tačnošću od onog sa običnom tačnošću.

9.6. Izračunavanje kompleksnih elementarnih funkcija

Izračunavanje elemeñtarnih funkcija za kompleksne vrednosti argumenata daje kompleksni broj, koji je iste vrste kao i argument elementar-

ne funkcije. Tako, ako je argument kompleksni broj obične tačnosti, biće i vrednost funkcije kompleksni broj obične tačnosti, odnosno ako je argument kompleksni broj dvostruke tačnosti biće i vrednost funkcije kompleksni broj dvostruke tačnosti. Ime kompleksnih elementarnih funkcija obične tačnosti počinje slovom C, a ime kompleksnih funkcija dvostruke tačnosti počinje slovima CD.

U tabeli 9.6.1. dat je spisak funkcijskih potprograma sa propisanim imenima, koji se mogu koristiti u FORTRAN-jeziku za izračunavanje kompleksnih elementarnih funkcija.

Pored funkcijskih potprograma navedenih u tabeli 9.6.1., u FORTRAN-jeziku postoji i izvestan broj funkcija koje se pišu na isti način kao i funkcijski potprogrami, ali se ne javljaju jedanput u programu na mašinskom jeziku, već onoliko puta koliko puta su zapisane u programu. Spisak ovih funkcija sa propisanim imenima dat je u tabeli 9.6.2.

U tabelama 9.6.1 i 9.6.2 korišćene su sledeće oznake:

K - kompleksna veličina obične tačnosti,
KK - kompleksna veličina dvostruke tačnosti,
R - realna veličina obične tačnosti,
RR - realna veličina dvostruke tačnosti,
z - kompleksna veličina, z=a+bi,
P - maksimalna vrednost konstante u pokretnom zarezu obične tačnosti ($P \approx 7,2 \cdot 10^{75}$),
DP - maksimalna vrednost konstante u pokretnom zarezu dvostuke tačnosti ($DP \approx 7,2 \cdot 10^{75}$)

Tabela 9.6.1

Način pisanja		Argumenti		Funkcija		Opis
U Fortranu	U matemat.	Vrsta	Ograničenje	Vrsta	Rel.greš. je manja od	
CEXP(z)	e^z	K	$\|a\| < 174,673$ $\|b\| \leq 0,8235 \cdot 10^6$	K	$1,18 \cdot 10^{-6}$	Eksponencijalna funkcija
CDEXP(z)		KK	$\|a\| < 174,673$ $\|b\| \leq 0,3537 \cdot 10^{16}$	KK	$3,63 \cdot 10^{-15}$	
CLOG(z)	Ln(z)	K	$z \neq 0+0i$	K	$2,00 \cdot 10^{-6}$	Prirodni logaritam
CDLOG(z)		KK	$z \neq 0+0i$	KK	$8,73 \cdot 10^{-15}$	
CSQRT(z)	$\sqrt{z}$	K	$\|a\|, \|b\| \leq$	K	$1,61 \cdot 10^{-6}$	Kvadratni koren
CDSQRT(z)		KK	$\|a\|, \|b\| \leq$	KK	$9,86 \cdot 10^{-16}$	
CSIN(z)	sin(z)	K	$\|a\| < 0,8235 \cdot 10^6$ $\|b\| \leq 174,673$	K	$1,97 \cdot 10^{-6}$	Trigonometrijske funkcije
CDSIN(z)		KK	$\|a\| < 0,3537 \cdot 10^{16}$ $\|b\| \leq 174,673$	KK	$3,72 \cdot 10^{-16}$	
CCOS(z)	cos(z)	K	$\|a\| < 0,8235 \cdot 10^6$ $\|b\| \leq 174,673$	K	$1,79 \cdot 10^{-6}$	
CDCOS(z)		KK	$\|a\| < 0,3537 \cdot 10^{16}$ $\|b\| \leq 174,673$	KK	$5,16 \cdot 10^{-15}$	
CABS(z)	$\|z\|=\|a+bi\|$	K	$(a^2+b^2)^{1/2} \leq P$	R	$1,87 \cdot 10^{-6}$	Apsolutna vrednost
CDABS(z)		KK	$(a^2+b^2)^{1/2} \leq DP$	RR	$3,32 \cdot 10^{-15}$	

Tabela 9.6.2

Način pisanja		Argumenti		Funkcija		Opis
U Fortranu	U matemat.	Vrsta	Ograničenje	Vrsta	Rel.greš.je manja od	
REAL(z)	Re(z)	K	$\|a\|, \|b\| \leq P$	R	-	Realni deo kompleksnog broja
AIMAG(z)	Im(z)	K	$\|a\|, \|b\| \leq P$	R	-	Imaginarni deo kompleksnog broja
CMPLX(a,b)	a+bi	R	$\|a\|, \|b\| \leq P$	K	-	Formiranje kompleksnog broja od dva realna broja
DCMPLX(a,b)		RR	$\|a\|, \|b\| \leq DP$	KK	-	
CONJG(z)	conj(z)	K	$\|a\|, \|b\| \leq P$	K	-	Konjugovan broj kompleksnog broja
DCONJG(z)		KK	$\|a\|, \|b\| \leq DP$	KK	-	

Primer

Za zadatu vrednost kompleksnog broja z izračunati funkcije

$$y_1 = \left[e^{z+1} + \mathrm{Ln}(3,6+4,2i)\right]^{\frac{1}{2}} \cdot |\cos z \,|\, \sin 2z$$

$$y_2 = \left[\mathrm{Re}(y_1)+I_m(y_1)\right] + \left[\mathrm{Re}(y_1)-I_m(y_1)\right] i$$

$$y_3 = \mathrm{conj}(y_2)+3,8-4i$$

u običnoj tačnosti.

Za izračunavanje u običnoj tačnosti FORTRAN-program ima sledeći izgled

```
      COMPLEX Y(3),Z
      WRITE(6,5)
    5 FORMAT(' ',6X,'REALNI DEO   IMAGINARNI',
     *' DEO'/)
      READ(5,10) Z
   10 FORMAT(2F8.4)
      Y(1)=CSQRT(CEXP(Z+1.)+CLOG((3.6,4.2)))*
     *CABS(Z)*CSIN(2.*Z)
      Y(2)=CMPLX(REAL(Y(1))+AIMAG(Y(1)),
     $REAL(Y(1))-AIMAG(Y(1)))
      Y(3)=CONJG(Y(2))+(3.8,-4.0)
      WRITE(6,20) Z,(Y(I),I=1,3)
   20 FORMAT(' ','Z =',2E15.7/' ','Y1=',2E15.7/
     *' ','Y2=',2E15.7/' ','Y3=',2E15.7)
      STOP
      END
```

Rezultati se dobijaju u obliku tabele

```
                REALNI DEO    IMAGINARNI DEO

Z = -0.4160000E 01  0.2012000E 02
Y1= -0.3258756E 19 -0.2785778E 19
Y2= -0.6044533E 19 -0.4729780E 18
Y3= -0.6044533E 19  0.4729780E 18
```

10. RACIONALNO KORIŠĆENJE UNUTRAŠNJE MEMORIJE RAČUNARA

U unutrašnjoj ili operativnoj memoriji računara nalaze se naredbe i podaci u internom kodu računara. Automatska obrada podataka odvija se po odredjenom programu. Za ovu obradu neophodna je komunikacija izmedju memorije i komandnog organa u cilju izvršavanja pojedinih naredbi programa, kao i izmedju memorije i aritmetičkog organa u cilju obrade podataka. Prema tome, od unutrašnje memorije se zahteva velika brzina upisa i izdavanja informacija. Medjutim, ovakve brze memorije predstavljaju skupe tehničke uredjaje, pa je unutrašnja memorija po pravilu vrlo ograničenog kapaciteta. Zato je problem racionalnog korišćenja unutrašnje memorije računara vrlo važan u programiranju. Ovaj problem se rešava na dva načina.

- izborom optimalne dužine podataka, i
- višestrukim korišćenjem memorijskog prostora.

10.1. Promenljiva dužina podataka

Najmanja adresiva jedinica memorije je podregistar, u kojem se može registrovati jedan karakter. Kod memorija koje su organizovane po registrima, ovi se sastoje od odredjenog broja podregistara. Tako registar memorije kod računara IBM-360/44 sastoji se od 4 podregistara, a podregistar od 8 ćelija. Za racionalno korišćenje unutrašnje memorije potrebno je obezbediti optimalan broj podregistara za pojedine informacije u računaru. Broj podregistara koji se koristi za registrovanje podatka u memoriji

zove se dužina podatka. U programima koje smo do sada pisali nije posebno ukazivano na dužinu podataka. Za sve podatke koji su korišćeni u programu, pretpostavljana je tzv. standardna dužina podataka. Standardna dužina podataka u FORTRAN-jeziku biće primenjena za sve one podatke za koje nije zahtevana drugačija dužina. Pored standardne dužine postoji minimalna i maksimalna dužina podataka. Minimalna dužina predstavlja najmanji broj podregistara, a maksimalna najveći broj podregistara koji se može koristiti za odredjenu informaciju u FORTRAN-jeziku. Standardna, minimalna i maksimalna dužina podataka u FORTRAN-jeziku prikazana je u tabeli 10.1.1.

Tabela 10.1.1

Vrsta podatka	Dužina podatka		
	Standardna	Minimalna	Maksimalna
Celobrojna promenljiva	4	2	4
Realna konstanta ili realna promenljiva	4	4	8
Kompleksna konstanta ili kompleksna promenljiva	8	8	16
Logička konstanta ili logička promenljiva	4	1	4

Celobrojna konstanta u programu uvek se registruje u jednom memorijskom registru, tj. ima dužinu 4. Za promenljive u programu čija je vrsta definisana unutrašnjom konvencijom FORTRAN-jezika važi standardna dužina podataka.

Za promenljive u programu čija je vrsta definisana opisnom naredbom za implicitnu deklaraciju vrste dužina podataka može biti različita i ukazuje se u naredbi

IMPLICIT lista (10.1.1)

gde je element liste u obliku

vrsta* s ($lista_1$) (10.1.2)

gde je s dužina podatka koja se odnosi na odgovarajuću vrstu promenljivih, čija imena počinju slovima navedenim u $lista_1$.

Ako promenljive koje se implicitno deklarišu imaju standardnu dužinu onda se element liste piše u obliku

$$\text{vrsta}\,(\text{lista}_1) \qquad (10.1.3)$$

tj. oblik bez navodjenja dužine promenljivih.

Tako se može pisati

```
IMPLICIT INTEGER*2(A,B,C),REAL*8(R,S,T)
```

što znači da će promenljive čija imena počinju slovima A, B i C biti celobrojne i svaka od njih registrovana u dva podregistra memorije, tj. u jednom memorijskom registru mogu se registrovati brojne vrednosti za dve ovakve promenljive, a promenljive R, S i T biće realne promenljive dvostruke tačnosti i svaka od njih biće registrovana u 8 podregistara, tj. u dva memorijska registra.

Ako je celobrojna promenljiva dužine dva podregistra, tada njena brojna vrednost x mora biti u intervalu

$$-2^{15} \leqslant x \leqslant 2^{15} - 1 = 32767 \qquad (10.1.4)$$

U naredbi za eksplicitnu deklaraciju vrste promenljivih može biti ukazana dužina promenljivih. To može biti učinjeno na više načina:

a) Definisanje dužine promenljivih za sve promenljive koje se navode u listi ove naredbe. U ovom slučaju naredba ima oblik

$$\text{vrsta}*s\ \text{lista} \qquad (10.1.5)$$

gde je

vrsta - službena reč INTEGER, REAL, COMPLEX ili LOGICAL,

s - ceo neoznačen broj koji ukazuje na dužinu promenljivih i elemenata nizova u listi,

lista - spisak imena promenljivih i nizova medju sobom razdvojenih zarezima.

Tako se može pisati

```
LOGICAL*1 A,DELTA,ALFA(10)                    (10.1.6)
```

što deklariše promenljive A, DELTA i 10 elemenata niza ALFA kao logičke promenljive dužine 1, tj. vrednosti ovih promenljivih biće registrovane svaka u po jednom podregistru.

b) Definisanje dužina promenljivih za svaku promenljivu ponaosob. U ovom slučaju naredba ima oblik

vrsta lista (10.1.7)

gde je

vrsta - službena reč INTEGER, REAL, COMPLEX ili LOGICAL,

lista - spisak elemenata medju sobom razdvojenih zarezima.

Elemenat liste, u (10.1,7) ima oblik:

ime*s (10.1.8)

ako se odnosi na promenljivu, odnosno

$$\text{ime}*s(\text{lista}_i) \qquad (10.1.9)$$

ako se odnosi na niz, gde je

ime - naziv promenljive ili niza,

s - dužina promenljive, odnosno elementa niza,

lista_i - spisak, od najviše 7, celih neoznačenih brojeva, medju sobom razdvojenih zarezima, koji definišu maksimalne vrednosti pojedinih indeksa niza.

Tako se može pisati

```
COMPLEX BETA,RAD*16,GRAD*16(10,15)          (10.1.10)
```

što znači da je promenljiva BETA kompleksna promenljiva obične tačnosti a RAD kompleksna promenljiva dvostrukе tačnosti, kao i svih 150 elemenata dvodimenzionalnog niza GRAD.

c) Definisanje dužina promenljivih može biti učinjeno i kombinovanjem zajedničke dužine(a) i pojedinačnih dužina (b). U ovom slučaju naredba ima oblik

vrsta*s lista (10.1.11)

gde su elementi liste imena promenljivih i nizova, pri čemu je njihova dužina definisana sa brojem s u (10.1.11), ili su elementi liste oblika (10.1.9), pri čemu je njihova dužina posebno definisana.

Tako se može pisati

```
INTEGER*2 LIST,MASA(50),GAMA*4                    (10.1.12)
```

što znači da su promenljiva LIST i 50 elemenata niza MASA deklarisani kao celobrojne promenljive dužine 2 podregistra, a promenljiva GAMA kao celobrojna promenljiva dužine 4 podregistra.

U odeljku 8.2 uvedena je opisna naredba DOUBLE PRCISION, za deklarisanje realnih promenljivih dvostruke tačnosti. Medjutim, eksplicitna deklaracija pomoću naredbe

```
REAL*8 lista                                      (10.1.13)
```

ima potpuno isti efekat kao naredba

```
DOUBLE PRECISION lista                            (10.1.14)
```

gde se podrazumeva da su liste u naredbi (10.1.13) i (10.1.14) jednake.

10.2. Višestruko korišćenje memorijskog prostora u okviru jedne programske jedinice

Svakom imenu promenljive ili elementa niza dodeljuje se u memoriji računara fizičko mesto u kojem se čuva vrednost ove promenljive. Kao što smo videli, ovo fizičko mesto može biti najmanje jedan podregistar, a najviše 16 njih u memoriji računara. Prostor u memoriji potreban za registrovanje jedne konstante zvaćemo polje. Više polja čine zonu u memoriji računara. Prema onome što smo do sada videli svako polje u memoriji registruje vrednost jedne promenljive ili jednog elementa niza. U toku izvršavanja programa iz polja se izdaje registrovana konstanta ili se upisuje, a u zavisnosti od mesta odgovarajuće promenljive u naredbi koja se izvršava.

Tako, ako se ime promenljive nalazi na desnoj strani znaka jednakosti, u aritmetičkoj naredbi, tada se vrši izdavanje vrednosti odgovarajuće promenljive iz memorije, a ako se ime promenljive nalazi na levoj strani jednakosti vrši se upis nove vrednosti u odgovarajuće memorijsko polje.

Kako FORTRAN-jezik dozvoljava veliki izbor u imenovanju promenljivih, to se u toku izrade programa uvode imena promenljivih prema njihovom značenju u problemu koji se rešava, kao i prema mnemotehničkim olakšicama u pisanju programa. Medjutim, vrlo često ovakvo imenovanje promenljivih dovodi do neracionalnog korišćenja memorijskog prostora. Zato u FORTRAN-jeziku postoji opisna naredba kojom se može zahtevati da više promenljivih ili nizova imaju zajedničku zonu u memoriji. U narednom izlaganju upoznaćemo se sa definisanjem zajedničke zone u memoriji u okviru jedne programske jedinice.

10.2.1. Zajednička polja za promenljive jednakih dužina

Opisna naredba za definisanje zajedničkih polja ima oblik

EQUIVALENCE lista (10.2.1)

gde je

EQUIVALENCE - službena reč,

lista - spisak elemenata medju sobom razdvojenih zarezima.

Element liste u (10.2.1) je oblika

($lista_1$) (10.2.2)

gde je $lista_1$ spisak imena promenljivih iste dužine medju sobom razdvojenih zarezima.

Tako se može pisati

```
REAL*8 MASA,DUZINA,JOT*4
INTEGER*2 A,BETA,C
EQUIVALENCE (A,BETA,C),(JOT,G),(MASA,DUZINA)
```

To znači da će celobrojnim promenljivim A, BETA i C biti dodeljeno isto polje u memoriji računara dužine 2 podregistra, a promenljivim JOT i G polje dužine 4 podregistara (1 memorijski registar), i na kraju promenljivim MASA i DUŽINA biće dodeljeno polje dužine dva memorijska registra, pošto su to realne promenljive dvostruke tačnosti. Prema tome, ako se ne bi koristila opisna naredba EQUIVALENCE za registrovanje promenljivih A, BETA C, JOT, G, MASA i DUŽINA bilo bi angažovano 3x2+2x4+2x8 = 30 podregistara. Medjutim, upotrebom naredbe EQUIVALENCE koristi se 2+4+ +8=14 podregistara memorije.

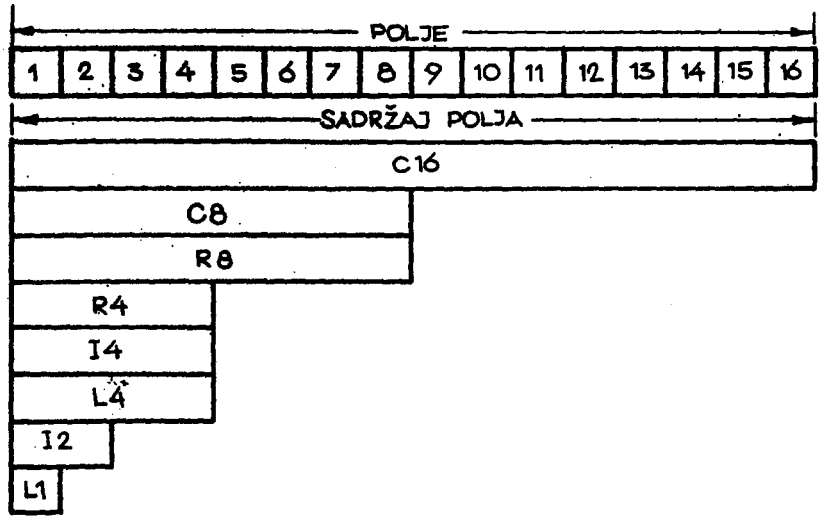
Opisna naredba EQUIVALENCE piše se pre prve izvršne naredbe programa ili potprograma. U odnosu na ostale opisne naredbe ova naredba se navodi posle naredbe za deklarisanje vrste i dimenzije nizova.

10.2.2. Zajednička polja za promenljive različitih dužina

Ako su promenljive različitih dužina, tada je polje odredjeno promenljivom najveće dužine. Sve promenljive navode se u listi u (10.2.2). Kao što je poznato, kompleksna promenljiva dvostruke tačnosti predstavlja promenljivu najveće dužine (16 podregistara), a logička promenljiva predstavlja promenljivu najmanje dužine (1 podregistar). Ako se u zajedničkom polju nalazi po jedna promenljiva različite dužine, tada svaka od njih počinje da se registruje od prvog levog podregistra polja, čiju ćemo adresu označiti sa 1, prema podregistrima veće relativne adrese (2, 3, ...). Na sl. 10. 2.1 prikazano je polje od 16 podregistara i označen raspored registrovanja promenljivih različite dužine.

Na sl. 10.2.1 uvedene su sledeće oznake:

C16 - kompleksna konstanta dvostruke tačnosti (dužine 16),
C8 - kompleksna konstanta obične tačnosti (dužine 8),
R8 - realna konstanta dvostruke tačnosti (dužine 8),
R4 - realna konstanta obične tačnosti (dužine 4),
I4 - celobrojna konstanta (dužine 4),
L4 - logička konstanta (dužine 4),
I2 - celobrojna konstanta (dužine 2),
L1 - logička konstanta (dužine 1).



Sl. 10.2.1

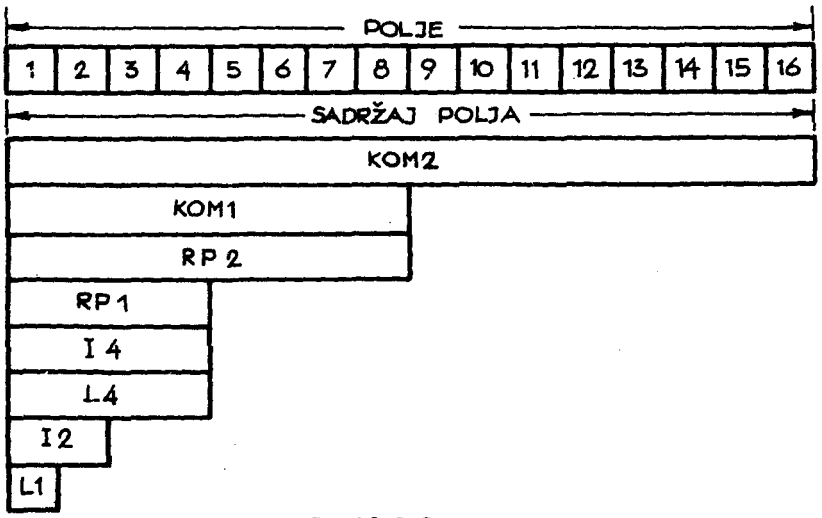
Prema tome, ako se žele registrovati u polju kompleksne promenljive dvostruke tačnosti KOM2, realna promenljiva dvostruke tačnosti RP2, kompleksna promenljiva obične tačnosti KOM1, realna promenljiva obične tačnosti RP1, celobrojne promenljive I4 i I2 dužine 4, odnosno 2, i logičke promenljive L4 i L1 dužine 4, odnosno 1, tada će opisne naredbe imati sledeći izgled

```
COMPLEX KOM1,KOM2*16
REAL RP2*8
INTEGER I2*2
LOGICAL L4,L1*1
EQUIVALENCE (KOM2,KOM1,RP2,RP1,I4,I2,L4,L1)
```

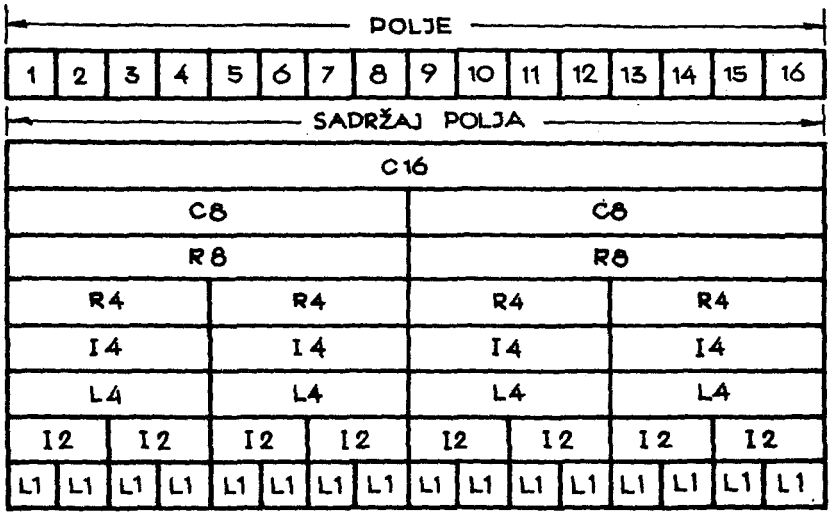
Promenljive RP1 i I4 nisu deklarisane po vrsti i dužini, jer za njih važi unutrašnja konvencija FORTRAN-jezika. U ovom slučaju polje sa rasporedom promenljivih prikazano je na sl. 10.2.2.

Medjutim, u jednom polju veće dužine može se registrovati veći broj promenljivih manje dužine. Na sl. 10.2.3 prikazano je polje od 16 podregistara sa mogućim registrovanjem promenljivih manje dužine. Oznake na sl. 10.2.3 iste su kao već korišćene oznake na sl. 10.2.1.

Opisne naredbe koje deklarišu korišćenje polja od 16 podregistara na



Sl. 10.2.2

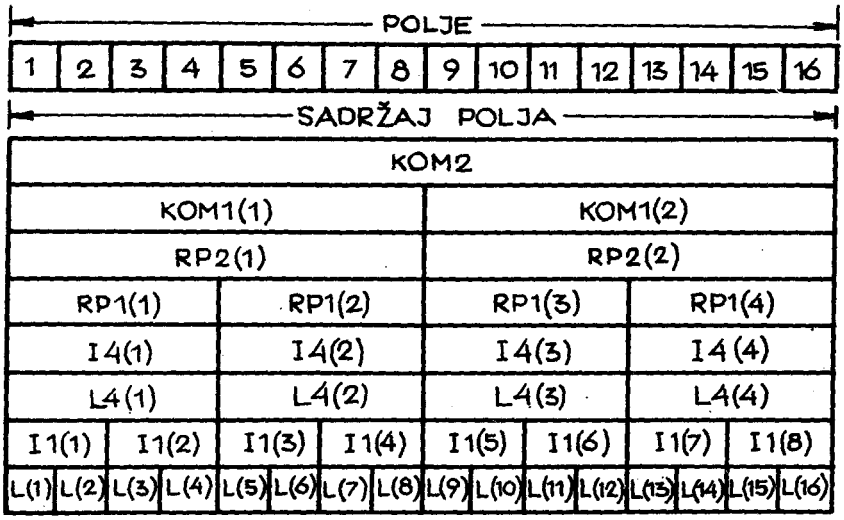


Sl. 10.2.3

način prikazan na sl. 10.2.3. imaju sledeći izgled:

```
      COMPLEX KOM2*16,KOM1(2)
      REAL RP2*8(2),RP1(4)
      INTEGER I4(4),I1*2(8)
      LOGICAL L4(4),L*1(16)
      EQUIVALENCE(KOM2,KOM1(1),RP2(1),RP1(1),
     *I4(1),I1(1),L4(1),L(1))
```

Na sl. 10.2.4 prikazan je raspored promenljivih unutar memorijskog polja od 16 podregistara



Sl. 10.2.4

Kao što se vidi, u naredbi EQUIVALENCE, u ovom slučaju, treba navesti prve elemente nizova. Detaljnije o zajedničkim zonama nizova biće reči u sledećem odeljku.

10.2.3. Zajednička zona za nizove

Elementi dva niza ili više nizova mogu imati zajedničku zonu u memoriji. U ovom slučaju element liste u (10.2.2) jeste oblika

$$\text{ime}(\text{lista}_2) \qquad (10.2.3)$$

gde je

ime - naziv niza,

lista_2 - spisak, od najviše 7, neoznačenih celih brojeva, medju sobom razdvojenih zarezima. Ovi brojevi definišu konkretan elemenat niza.

Pored oblika (10.2.3) na konkretan element niza može se ukazati i preko odgovarajućeg elementa jednodimenzionalnog niza (vidi odeljak 5.5, relacija 5.5.6), tj.

ime(j) (10.2.4)

gde je j ceo neoznačen broj odredjen relacijom (5.5.6).

Kao što je poznato, nizovi se registruju u registrima memorije čije adrese slede jedna za drugom. U EQUIVALENCE-naredbi navode se konkretni elementi nizova koji će imati zajedničko polje, a s obzirom na to da se ostali elementi niza registruju u susednim registrima, to će i drugi elementi nizova imati zajednička polja.

Tako opisne naredbe

```
DIMENSION A(5,7),B(35)
EQUIVALENCE (A(1,1),B(1))
```

definišu dvodimenzioni niz A i jednodimenzionalni niz B od po 35 elemenata, kao nizove koji se registruju u zajedničkoj zoni od 35 registara memorije. Isti efekat će imati i zapis

```
DIMENSION A(5,7),B(35)
EQUIVALENCE (A(1),B(1))
```

gde je u prvom slučaju, u naredbi EQUIVALENCE konkretan element niza A ukazan oblikom (10.2.3), a u drugom slučaju oblikom (10.2.4).

Međutim, ako nizovi ne sadrže isti broj elemenata tada će biti definisana zajednička zona u kojoj će raspored nizova zavisiti od navedenih konkretnih elemenata u naredbi EQUIVANENCE. Tako će naredbama

```
DIMENSION A(10),B(5),C(2,2)
EQUIVALENCE (A(5),B(1),C(1,1))
```

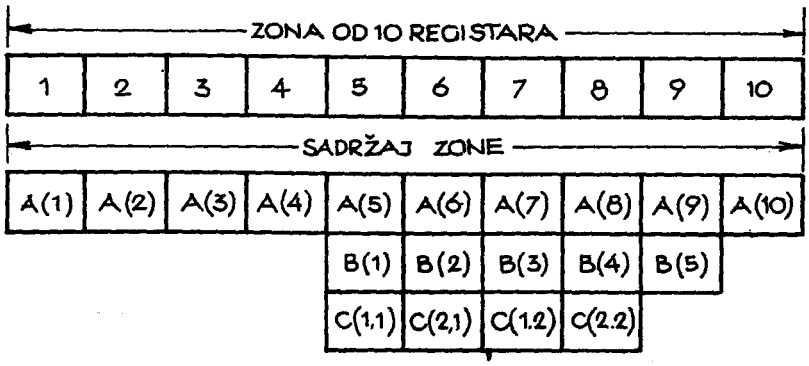
biti definisana zajednička zona od 10 registara u kojoj će nizovi biti raspoređeni na način prikazan na sl. 10.2.5.

Isti raspored nizova proizvodi i zapis

```
DIMENSION A(10),B(5),C(2,2)
EQUIVALENCE (A(6),B(2),C(2,1))
```

kao i drugi zapisi prema sl. 10.2.5, koji definišu odgovarajuće elemente nizova A, B i C u istom polju memorijske zone.

Ako su nizovi sa elementima različitih dužina, tada se elementi nizova registruju u zoni od navedenog početka sleva na desno, prema odgovarajućoj dužini elemenata. Pri ovome se u jednom memorijskom registru mo-



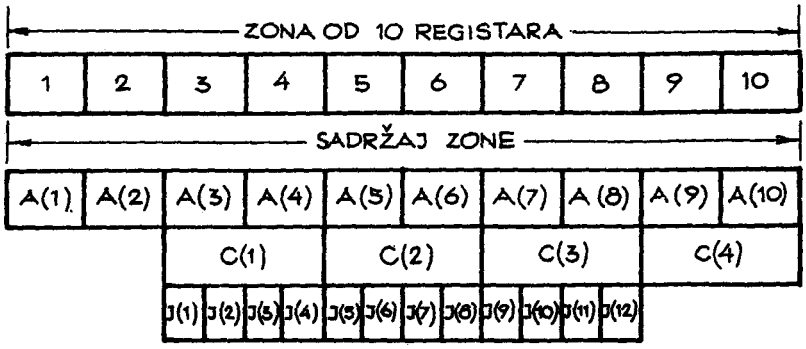
Sl. 10.2.5

ra nalaziti podatak registrovan prema propisanom načinu registrovanja odgovarajućeg podatka.

Tako se može pisati

```
COMPLEX C(4)
INTEGER J*2(12)
DIMENSION A(10)
EQUIVALENCE(A(3),C(1),J(1))
```

Raspored elemenata nizova prikazan je na sl. 10.2.6. Elemenat C(1) niza C zauzima dva registra u memoriji, kao kompleksna promenljiva obične tačnosti. U istom polju registrovani su elementi A(3) i A(4) niza A, odnosno elementi J(1), J(2), J(3), J(4) niza J.

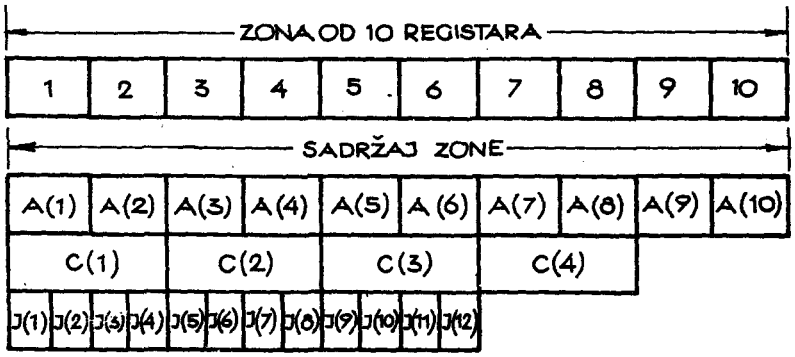


Sl. 10.2.6

Isti nizovi mogu se registrovati i od početka zajedničke zone; tada bi opisne naredbe imale sledeći izgled:

```
COMPLEX C(4)
INTEGER J*2(12)
DIMENSION A(10)
EQUIVALENCE(A(1),C(1),J(1))
```

U ovom slučaju raspored elemenata nizova prikazan je na sl. 10.2.7.



Sl. 10.2.7

10.3. Višestruko korišćenje memorijskog prostora od strane više programskih jedinica

Naredba EQUIVALENCE omogućuje višestruko korišćenje memorijskog prostora u okviru jednog programa ili potprograma. Medjutim, potprogrami se vrlo često koriste u programiranju, i nije teško pretpostaviti da se izvestan broj podataka koristi u programu kao i u jednom potprogramu ili u više potprograma, ili pak da se isti memorijski prostor koristi u više programskih jedinica. U ovom slučaju treba omogućiti da polje ili zona u memoriji bude zajednička za više programskih jedinica. Ovakve zone u memoriji mogu se definisati na dva načina, kao

- neimenovane zajedničke zone u memoriji, i kao
- imenovane zajedničke zone u memoriji.

10.2.1. Neimenovana zajednička zona u memoriji

Opisna naredba za definisanje neimenovane zajedničke zone u memoriji, piše se u obliku

COMMON lista (10.3.1)

gde je

COMMON - službena reč,

lista - spisak elementa medju sobom razdvojenih zarezima.

Elemenat liste može biti ime promenljive ili niza. Ako je elemenat liste ime niza, onda se iza imena izmedju zagrada mogu navesti maksimalne vrednosti indeksa niza, tj.

ime(lista$_1$) (10.3.2)

gde je

ime - naziv niza,

lista$_1$ - spisak, od najviše 7, neoznačenih celih brojeva, koji definišu maksimalne vrednosti pojedinih indeksa.

Ako se jedna naredba (10.3.1) nalazi u programu, a druga ovakva naredba u potprogramu, sa ekvivalentnim listama, tada će odgovarajuće promenljive iz jedne i druge liste imati zajednička polja u memoriji. Pod ekvivalentnim listama podrazumevaju se liste sa istim redosledom promenljivih po vrsti i dužini.

Tako, ako se u programu nalazi naredba

COMMON A1, A2, A3, J1, J2

a u potprogramu naredba

COMMON X1, X2, X3, K1, K2

tada će promenljive A1 i X1 imati zajednički registar u memoriji, kao i promenljive A2 i X2, A3 i X3, J1 i K1, kao i J2 i K2.

Odmah treba uočiti da ovakva zajednička zona izmedju programa i potprograma predstavlja jedan način za ulaz podataka u potprogram, bez njihovog navodjenja kao argumenata potprograma, kao i za izlaz rezultata potprograma.

Ako se COMMON-naredbom želi dobiti zona za zajedničkim podacima za više programskih jedinica, tada sve COMMON-naredbe moraju ima-

ti ekvivalentne liste u celini ili jednim njihovim delom sleva nadesno.

Tako, ako se naredbe

```
INTEGER*2 J,K
LOGICAL*1 L1,L2
COMMON A(30),J,K,L1,L2
```

nalaze u programu, a naredbe

```
INTEGER*2 C,D
LOGICAL*1 F1,F2
COMMON B(30),C,D,F1,F2
```

u potprogramu, liste COMMON-naredbi su ekvivalentne i u ovakvoj zajedničkoj zoni mogu se prenositi i vrednosti iz programa u potprogram, i obratno pošto se liste slažu po redu, vrsti i dužini promenljivih i nizova.

Ako se zajedničkom zonom ne prenose vrednosti promenljivih, već se samo racionalno koristi memorijski prostor, tada redosled promenljivih u listi može biti proizvoljan. Medjutim, ovaj redosled mora biti usaglašen u programu i potprogramu. Ova saglasnost može se postići na dva načina

- navodjenjem promenljivih u redosledu po njihovoj opadajućoj dužini, ili
- navodjenjem promenljivih u proizvoljnom redosledu sa uvodjenjem fiktivnih promenljivih da bi se usaglasile dužine promenljivih u listama COMMON-naredbi.

10.3.2. Imenovana zajednička zona u memoriji

Zajednička zona u memoriji kao i pojedini njeni delovi mogu dobiti ime. U ovom slučaju element liste u (10.3.1) ima oblik

$$/\text{ime}/\ \text{lista}_2 \qquad (10.3.3)$$

gde je

ime - naziv zone koju čine promenljive koje slede u listi_2,

lista_2 - gradi se na isti način kao i lista u (10.3.1).

Naziv zone definiše se na isti način kao i ime promenljive i uvek se piše izmedju kosih crta.

Tako se može pisati

```
COMMON /ZONA1/A(5),B(10)/ZONA2/KSI(5),JOT
```

gde elementi nizova A i B čine deo zone u memoriji, koja nosi ime ZONA1, a elementi niza KSI i promenljiva JOT čine deo zone u memoriji sa imenom ZONA2.

U istoj COMMON-naredbi mogu se nalaziti imenovani i neimenovani delovi zone u memoriji. Ako se iza imenovanog dela, želi navesti neimenovan deo zone, tada se oni razdvajaju sa dve kose crte.

Tako se može pisati

```
COMMON A/BETA/B1,B2,B3//C,D,E(10)
```

Ovakva zajednička zona sadrži neimenovani deo, koji čine promenljive A, C, D i niz E(10), i imenovani deo, koji čine promenljive B1, B2 i B3 sa imenom BETA.

Ako se nadje više COMMON-naredbi u jednoj programskoj jedinici sa n različitih lista, tj.

$$\begin{aligned} &\text{COMMON lista}_1 \\ &\text{COMMON lista}_2 \\ &\quad\vdots \\ &\text{COMMON lista}_n \end{aligned} \qquad (10.3.4)$$

onda je njihov efekat isti kao da je jedna naredba oblika

$$\text{COMMON lista}_1, \text{lista}_2, \ldots, \text{lista}_n \qquad (10.3.5)$$

Naredba COMMON, kao opisna naredba, navodi se pre prve izvršne naredbe programa. Ako postoje i druge opisne naredbe u programu, tada je njihov redosled sledeći:

1. Opisne naredbe za eksplicitnu deklaraciju vrste (REAL, INTEGER, COMPLEX, LOGICAL, DOUBLE PRECISION),
2. Opisna naredba za implicitnu deklaraciju vrste (IMPLICIT),
3. Opisna naredba za navodjenje imena potprograma, koji se javljaju kao argumenti drugih potprograma (EXTERNAL),
4. Opisna naredba za dimenzionisanje nizova (DIMENSION),
5. Opisna naredba za definisanje zajedničke zone u raznim programskim jedinicama (COMMON),

6. Opisna naredba za definisanje zajedničke zone u jednoj programskoj jedinici (EQUIVALENCE),

7. Funkcijske naredbe, i

8. Izvršne naredbe programa.

11. DODELJIVANJE POČETNIH VREDNOSTI PROMENLJIVIM

U mnogim problemima, pored ulaznih podataka, javlja se i izvestan broj konstanti, koje ulaze u proračun sa nepromenjenim vrednostima pri svakom izvodjenju proračuna. Ovakve konstante dodeljuju se promenljivim jedanput na početku proračuna.

Takvo dodeljivanje početnih vrednosti promenljivim, očigledno, može se izvršiti pomoću aritmetičkih naredbi, kod kojih će se na desnoj strani znaka jednakosti nalaziti konstanta, a na levoj ime promenljive kojoj se ova konstanta dodeljuje. Medjutim, aritmetičke naredbe predstavljaju izvršne naredbe programa i posle prevodjenja programa sa FORTRAN-jezika na mašinski jezik, ove naredbe ostaju u programu na mašinskom jeziku, čime zauzimaju prostor u memoriji i pored toga što će se izvršiti samo jedanput na početku programa. Prema tome, ovako dodeljivanje početnih vrednosti predstavlja neekonomično korišćenje memorijskog prostora.

U ovoj glavi biće objašnjene naredbe koje omogućuju dodeljivanje početnih vrednosti promenljivim u fazi prevodjenja programa sa FORTRAN-jezika na mašinski jezik. Prema tome, ove naredbe se ne javljaju u programu na mašinskom jeziku, ali obezbedjuju postavljanje konstanti u memorijskim registrima, bez korišćenja izvršnih naredbi FORTRAN-jezika.

11.1. Dodeljivanje početnih vrednosti promenljivim naredbom za eksplicitnu deklaraciju vrste promenljivih

Već smo videli da opisna naredba za eksplicitnu deklaraciju vrste promenljive (REAL, INTEGER, COMPLEX, LOGICAL) ima dve funkcije:

- deklariše vrstu promenljivih po imenima promenljivih,
- definiše dužine promenljivih.

Pored navedenih funkcija, ova naredba se može koristiti i za dodeljivanje početnih vrednosti promenljivim i nizovima koji se pojavljuju u listi ove naredbe. Tako, opšta funkcija ove naredbe može se opisati na sledeći način

$$\text{vrsta} * s\ \text{lista} \qquad (11.1.1)$$

gde je

vrsta - službena reč (REAL, INTEGER, COMPLEX ili LOGICAL),

s - neobavezan ceo neoznačen broj koji definiše dužinu promenljivih za koje to nije posebno ukazano u listi ove naredbe,

lista - spisak elemenata medju sobom razdvojenih zarezima.

Ako se deklariše ime promenljive u listi, onda je elemenat liste

$$\text{ime} * s_1/k/ \qquad (11.1.2)$$

gde je

ime - naziv promenljive,

s_1 - neobavezan ceo neoznačen broj koji definiše dužinu promenljive, a

k - neobavezna konstanta, koja se dodeljuje kao početna vrednost promenljivoj.

Ako se deklariše ime niza u listi (11.1.1), onda je element liste

$$\text{ime}*s_2(\text{lista}_1)/\text{lista}_2\ / \qquad (11.1.3)$$

gde je

ime - naziv niza,

s_2 - neobavezan ceo neoznačen broj, koji definiše dužinu elemenata niza,

lista_1 - neobavezan spisak, od najviše 7, neoznačenih, celih brojeva, medju sobom razdvojenih zarezima, koji definišu maksimalne vrednosti indeksa niza,

lista_2 - neobavezan spisak konstanti, medju sobom razdvojenih zarezima, koje se dodeljuju kao početne vrednosti elementima niza.

Ako su u lista_2 uzastopne konstante medju sobom jednake, tada element liste_2 može imati oblik

m * k (11.1.4)

gde je

m - ceo neoznačen broj koji ukazuje na broj konstanti sa vrednošću k,

k - konstanta koja se ponavlja m puta.

Tako se može pisati

```
INTEGER*2 JOT/186/,ALFA(5,10)/50*0/
REAL MASA/-7.2E3/,BETA(10)/5*1.0,5*2.0/
COMPLEX*16 DELTA/(-1.4D-2,3.2D4)/
LOGICAL LOG*1/.TRUE./,PLUS/.FALSE./
```

čime se postiže sledeće dejstvo

- promenljiva JOT i elementi niza ALFA deklarišu se kao celobrojne promenljive dužine dva podregistra, i promenljivoj JOT se dodeljuje početna vrednost 186, a svih 50 elemenata niza ALFA dobijaju početnu vrednost nula,
- promenljiva MASA i elementi niza BETA deklarišu se kao realne promenljive dužine 4 podregistara, i promenljivoj MASA dodeljuje se početna vrednost - 7200, a prvih pet elemenata niza BETA dobijaju brojnu vrednost 1, dok sledećih 5 elemenata vrednost 2,
- kompleksnoj promenljivoj dvostruke tačnosti DELTA dodeljuje se početna vrednost -0,014+32000·i,
- logičkoj promenljivoj LOG dužine jedan podregistar dodeljuje se početna vrednost .TRUE., a logičkoj promenljivoj PLUS, dužine 4 podregistra, dodeljuje se vrednost .FALSE.

11.2. Naredba za dodeljivanje početnih vrednosti

Dodeljivanje početnih vrednosti može se izvršiti i pomoću posebne naredbe oblika

DATA lista (11.2.1)

gde je

DATA - službena reč,

lista - spisak elemenata medju sobom razdvojenih zarezima.

Elementi liste, u (11.2.1), imaju oblik

$$lista_1/lista_2/ \qquad (11.2.2)$$

gde je

$lista_1$ - spisak imena promenljivih sa indeksom ili bez njega i imena nizova medju sobom razdvojenih zarezima,

$lista_2$ - spisak konstanata koje se sleva nadesno dodeljuju promenljivim i elementima nizova navedenim u $listi_1$, medju sobom razdvojenih zarezima.

Konstante u $listi_2$ mogu biti celobrojne, realne, kompleksne, heksadekadne, logičke ili alfabetske (literali). Ako je više uzastopnih konstanti jednako, može se pisati

$$m * k \qquad (11.2.3)$$

gde je

m - ceo neoznačen broj koji ukazuje na broj ponavljanja konstante k,

k - konstanta koja se ponavlja.

Tako se može pisati

```
LOGICAL L(2)
DIMENSION A(10),C(5)
DATA N/12/,A/10*0./,B,C/3.2,5*1.5/,L/.TRUE.,.FALSE./
```

čime se postiže sledeće dejstvo:

- celobrojna promenljiva N dobija brojnu vrednost 12,
- deset elemenata niza A dobijaju vrednost nula,
- promenljiva B dobija vrednost 3,2, a pet elemenata niza C dobijaju vrednosti 1,5, i
- logičke promenljive L(1) i L(2) niza L dobijaju vrednosti .TRUE., odnosno .FALSE.

Ako se želi proizvoljan sadržaj postaviti u memorijski registar, pogodno je koristiti heksadekadne konstante.

Heksadekadna konstanta piše se kao niz heksadekadnih cifara o, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E i F ispred kojih stoji slovo Z.

Kako binarni kod heksadekadnih cifara sadrži četiri binarne cifre, to znači da u jednom podregistru sa 8 ćelija mogu da se registruju dve heksadekadne cifre. Pošto je binarni kod heksadekadnih cifara potpun ravnomeran kod, to znači da se pomoću heksadekadnih cifara može u memorij-

skom registru postaviti proizvoljan binarni sadržaj. Maksimalni broj cifara koji može imati heksadekadna konstanta zavisi od definisane dužine promenljive. Kako se dužina promenljive odredjuje po broju podregistara za njeno registrovanje, a jedan podregistar sadrži dve heksadekadne cifre, to je maksimalni broj cifara heksadekadne konstante dva puta veći od dužine promenljive kojoj se dodeljuje ova konstanta. Ako je broj cifara heksadekadne konstante veći od dozvoljenog za datu dužinu promenljive, odbacuju se heksadekadne cifre s leve strane; ako je pak broj cifara manji, s leve strane se dodaju nule. Tako se može pisati

```
      COMPLEX C,D*16
      INTEGER*2 BETA,ALFA
      LOGICAL LOG*1
      DATA C/'3.8.1970'/,D/16HSREDNJA VREDNOST/,
     *     BETA/'AB'/,ALFA/ZC1C5/,LOG/ZD3/
```

čime se postiže sledeće:

- promenljivoj C, čija je dužina 8, dodeljuje se početna vrednost literal

  3.8.1970

  koji je definisan izmedju apostrofa,
- promenljivoj D, čija je dužina 16, dodeljuje se kao početna vrednost literal

  SREDNJA VREDNOST

  koji je definisan opisom 16H,
- promenljivoj BETA, čija je dužina 2, dodeljuje se kao početna vrednost literal

  AB

  koji je definisan izmedju apostrofa;
- promenljivoj ALFA, čija je dužina 2, dodeljuje se kao početna vrednost heksadekadna konstanta

  C1C5

  koja je definisana početnim slovom Z, i
- promenljivoj LOG, čija je dužina 1, dodeljuje se kao početna vrednost heksadekadna konstanta

  D3

  koja je definisana početnim slovom Z.

Naredba (11.2.1) piše se pre izvršnih naredbi programa, i tada je redosled opisnih naredbi sledeći:

1) Opisne naredbe za eksplicitnu i implicitnu deklaraciju vrste (REAL, INTEGER, DOUBLE PRECISION, COMPLEX, LOGICAL, IMPLICIT),

2) Opisna naredba za navodjenje imena potprograma koji se javljaju kao argumenti drugih potprograma (EXTERNAL),

3) Opisna naredba za definisanje dimenzije nizova i maksimalnih vrednosti indeksa nizova (DIMENSION),

4) Opisna naredba za definisanje zajedničkih zona (COMMON),

5) Opisna naredba za definisanje zajedničkih polja (EQUIVALENCE),

6) Opisna naredba za postavljanje početnih vrednosti (DATA),

7) Funkcijske naredbe, i

8) Izvršne naredbe programa.

Naredba (11.2.1) ne može se koristiti za dodeljivanje početnih vrednosti promenljivim koje ulaze u zajedničku zonu programskih jednica (COMMON-zona). Ovakvim promenljivim dodeljuje se početna vrednost pomoću posebne programske jedinice BLOCK DATA, koja je opisana u sledećem odeljku.

11.3. Programska jedinica za dodeljivanje početnih vrednosti zajedničkim zonama u memoriji

Postavljanje početnih vrednosti promenljivim i nizovima koji čine zajedničku zonu za više programskih jedinica vrši se preko posebne programske jedinice, čija je struktura sledeća

```
BLOCK DATA
  |                                        (11.3.1)
  |
END
```

gde je

BLOCK DATA - službena reč, koja označava programsku jedinicu za postavljanje početnih vrednosti u zajedničkim zonama u memoriji,

END - službena reč koja označava fizički kraj programske jedinice.

Izmedju prve naredbe programske jedinice (BLOCK DATA) i zadnje naredbe (END), ne sme se nalaziti nijedna izvršna naredba. Ako se koriste naredbe za eksplicitnu ili implicitnu deklaraciju vrste promenljivih one se moraju pisati neposredno iza prve naredbe potprograma (BLOCK DATA). Iza ovih naredbi navodi se naredba COMMON u kojoj se mogu navesti samo imenovane zajedničke zone, i to sve promenljive ovih zona, bez obzira da li dobijaju početne vrednosti ili ne. Ova programska jedinica se posebno ne poziva u programu, već čini sastavni deo programa, i izvršava se na početku programa.

Tako se može pisati

```
BLOK DATA
REAL KOR2
COMMON/KON/PI,E,KOR2
DATA PI,E,KOR2/3.141593,2.718282,1.414214/
END
```

čime se postiže sledeće:

- zajednička zona sa imenom KON sadrži promenljive PI, E, KOR2, kojima se dodeljuju početne vrednosti,
- promenljivoj PI dodeljuje se početna vrednost 3.141593,
- promenljivoj E dodeljuje se početna vrednost 2.718282, i
- promeljivoj KOR2 dodeljuje se početna vrednost 1.414214.

12. OPŠTE MOGUĆNOSTI UNOŠENJA I IZDAVANJA PODATAKA

U ovoj glavi biće izložene dalje mogućnosti unošenja i izdavanja podataka u FORTRAN-jeziku. Prvi deo materijala (odeljak 12.1.i 12.2)predstavlja dalje mogućnosti opisne naredbe FORMAT, a drugi deo (odeljak 12.3) odnosi se na unošenje i izdavanje podataka bez korišćenja naredbe FORMAT.

12.1. Dalje mogućnosti naredbe FORMAT

12.1.1. Opšti opis podataka

Polje ulaznog ili izlaznog sloga, koje sadrži celobrojne, mešovite, kompleksne ili logičke konstante, može imati opšti opis

$$nGk.d \qquad (12.1.1)$$

gde je

- n - ceo neoznačen broj koji ukazuje na broj ponavljanja opisa,
- G - simbol FORTRAN-jezika,
- k - ceo neoznačen broj koji ukazuje na dužinu polja u ulaznom, odnosno izlaznom, slogu,
- d - ceo neoznačen broj koji ukazuje na broj važećih cifara kada se opisuje polje sa mešovitim ili kompleksnim brojevima. Ova konstanta je bez značaja kada se opisuje polje sa celobrojnom ili logičkom konstantom.

Ako se izdaje mešoviti broj x u intervalu

$$0,1 \leq |x| < 10^d \qquad (12.1.2)$$

gde je d parametar u specifikaciji (12.1.1), tada se broj izdaje bez izložioca. U suprotnom slučaju, mešoviti broj se izdaje sa izložiocem E ili D, što zavisi od definisane dužine promenljive, čija je to brojna vrednost. Medjutim, pri izdavanju mešovitih brojeva u širini polja k treba uvek predvideti mesta za izložilac broja.

Pri korišćenju opisa (12.1.1) treba voditi računa o sledećem: ako je dužina polja (k) nedovoljna za smeštaj brojnog podatka, tada će biti polje ispunjeno sa k zvezdica (*).

Primer

Sastaviti program koji dodeljuje početne vrednosti promenljivim i štampa njihove vrednosti na sledeći način

a) Celobrojne promenljive

ALFA = 4236

JOT = 5

b) Realna promenljiva dvostruke tačnosti

D = 324.12

c) Realna promenljiva obične tačnosti

R = -125.6

d) Kompleksna promenljiva

KOM = (14.2, -3.8)

e) Logička promenljiva

LOG = .FALSE.

Program ima sledeći izgled

```
      INTEGER*2 ALFA/4236/,JOT/5/
      REAL*8 D/324.12/,R*4/-125.6/
      COMPLEX KOM/(14.2,-3.8)/
      LOGICAL*1 LOG/.FALSE./
      WRITE(6,10) ALFA,JOT,D,R,KOM,LOG
   10 FORMAT(' ',2G4,G12.5,G11.2,2G11.3,G3)
      STOP
      END
```

Vrednosti promenljivih štampaju se u obliku

```
4236   5  324.12        -0.13E 03    14.2        -3.80        F
```

Pošto je vrednost promenljive R van intervala (12.1.2 , to je štampanje izvršeno u eksponencijalnom obliku (sa zaokruženjem).

12.1.2. Koeficijent razmere

Ako se konstanta u ulaznom, odnosno, izlaznom slogu opisuje opisom F, E ili D, tada se može uz ovaj opis primeniti koeficijent razmere u obliku

mPnFk.d (12.1.3)

ili

mPnEk.d (12.1.4)

ili

mPnDk.d (12.1.5)

gde je

m - ceo označen broj koji ukazuje na koeficijent razmere,

P - simbol FORTRAN-jezika.

Ako se opis (12.1.3) koristi za opis polja u ulaznom slogu, tada ima sledeće dejstvo

$$u = s \cdot 10^{-m} \qquad (12.1.6)$$

gde je

s - vrednost konstante u ulaznom slogu, a

u - vrednost konstante koja će biti registrovana u memoriji.

Ako se opis (12.1.3) koristi za opis polja u izlaznom slogu, tada on ima sledeće dejstvo

$$s = u \cdot 10^{m} \qquad (12.1.7)$$

gde je

u - vrednost konstante u memoriji, a

s - vrednost konstante koja će biti izdata u izlaznom slogu.

Ako se opis (12.1.4) ili (12.1.5) primenjuje na konstante u poljima ulaznog sloga, on je bez dejstva. Ako se ovi opisi primenjuju na izlazu, tada ne utiču na brojnu vrednost konstanti, već samo na oblik štampanja tako što će se vrednost mantise povećati 10 puta, a eksponent umanjiti za m.

Koeficijent razmere primenjen na jedan opis u FORMAT-naredbi ostaje u važnosti i na svim ostalim opisima koji slede iza ovog opisa. Ako se želi ukinuti važnost koeficijenta razmere, treba zapisati opis F, E ili D u obliku (12.1.3), (12.1.4) ili (12.1.5) u kojem će biti m = 0.

12.1.3. Razmeštaj polja u ulazno-izlaznom slogu

Posebnim opisom može se definisati početak polja u ulaznom, odnosno u izlaznom slogu. Ovaj opis se piše u obliku

Tn (12.1.8)

gde je

T - simbol FORTRAN-jezika,

n - ceo neoznačen broj, manji od maksimalne dužine sloga ili jednak ovo , i označava početak polja u slogu.

Medjutim, kada se radi o izlazu treba imati u vidu da je prvi simbol u izlaznom slogu komandnog karaktera i da se odnosi na vertikalno pomeranje papira na štampaču. Tako je drugi simbol izlaznog sloga u stvari prvi simbol koji se štampa, pa n u (12.1.8) ukazuje na (n-1)-vi simbol u jednom štampanom redu.

Primer

Sledeći program

```
      A=27.13
      B=-0.014
      WRITE(6,10) A,B
   10 FORMAT(' A=',2PF6.0/T4,E10.1,T1,' B=')
      STOP
      END
```

daje štampani dokument u obliku

```
 A= 2713.
 B= -14.0E-03
```

Pošto je na opis F, za promenljivu A, primenjen koeficijent razmere 2P, to je brojna vrednost promenljive A, povećana 10 puta pre štampanja. Kako se koeficijent razmere prenosi i na sledeće opise koji slede, to je i za opis E primenjen isti koeficijent razmere. Medjutim, brojna

vrednost promenljive B, neće biti promenjena, ali će oblik štampanja biti takav da će mantisa biti pomnožena sa 10^2 , a eksponent umanjen za 2. U drugom izlaznom slogu raspored polja u slogu je definisan opisima T.

12.1.4. Opis heksadekadnih konstanti

U odeljku 11.2 videli smo da se početna vrednost promenljivih može postaviti pomoću heksadekadnih konstanti. Medjutim, heksadekadna konstanta se može naći i u polju ulaznog ili izlaznog sloga. U ovom slučaju opisuje se sa

nZk (12.1.9)

gde je

n - ceo neoznačen broj koji označava broj ponavljanja opisa,

Z - simbol FORTRAN-jezika, i

k - ceo neoznačen broj koji definiše dužinu polja u slogu.

Ako je heksadekadna konstanta duža od dužine polja, tada se odbacuju cifre sleva. Ako je heksadekadna konstanta kraća od dužine polja, vrši se dopuna sleva nulama u slučaju ulaza, odnosno znacima blanko u slučaju izlaza.

12.1.5. Opis alfabetskih podataka

FORTRAN-jezik je definisan pre svega za opis problema, u kojima se pretežno javljaju izračunavanja po odredjenim formulama; drugim rečima za obradu brojnih podataka. Alfabetski podaci koji predstavljaju niz simbola, javljaju se u obliku literala, kao neimenovani podaci, a samim tim, ne pružaju se veće mogućnosti manipulacije sa njima u programu. Medjutim, da bi i u FORTRAN-jeziku postojala mogućnost za veće manipulacije sa alfabetskim podacima, uveden je opis

nAk (12.1.10)

gde je

n - ceo neoznačen broj koji označava broj ponavljanja opisa,

A - simbol FORTRAN-jezika, i

k - ceo neoznačen broj koji defniše dužinu alfabetskog podatka.

Opis (12.1.10) može se koristiti za ulaz i izlaz alfabetskih podataka. Ovaj opis se odnosi na alfabetske podatke, kojima se dodeljuje ime na isti način kao i ime promenljivim. Dužina alfabetskog podatka odgovara dužini promenljive, čije ime se dodeljuje alfabetskom podatku. Ime ovakvog alfabetskog podatka piše se u listi ulazne naredbe, kada se unosi, odnosno u listi izlazne naredbe, kada se izdaje alfabetski podatak.

Ako je dužina alfabetskog podatka (k) jednaka po broju simbola sa deklarisanom dužinom promenljive kojoj se dodeljuje alfabetski podatak tada odgovarajuće polje u ulaznom, odnosno u izlaznom slogu sadrži k simbola. Ako je dužina alfabetskog podatka manja od dužine polja, tada se alfabetski podatak postavlja u polju sleva nadesno, a u ostali deo polja postavljaju se znaci blanko. Ako je dužina alfabetskog podatka veća od dužine polja, tada se alfabetski podatak postavlja sleva nadesno, a višak simbola se odbacuje.

Primer

U 1. koloni kartice bušen je ceo broj k, a od 2. do 10. kolone broj x. Izračunati vrednost α, β ili γ u zavisnosti od vrednosti broja k, na sledeći način

$$k = \begin{cases} 1 & \text{izračunati } \alpha = x^2 + 1 \\ 2 & \text{" } \beta = 2x + 3,5 \\ 3 & \text{" } \gamma = x^2 - 2x - 4 \end{cases}$$

Program sastaviti tako da se može koristiti za proizvoljan broj ulaznih kartica.

Program na FORTRAN-jeziku ima sledeći izgled:

```
      DIMENSION A(3)
      DATA A(1)/'ALFA'/,A(2)/'BETA'/,A(3)/'GAMA'/
300   READ(5,100,END=500) K,X
100   FORMAT(I1,F9.4)
      GO TO (1,2,3),K
    1 Y=X*X+1.
400   WRITE(6,200) A(K),Y
200   FORMAT(' ',A4,' = ',E12.5)
      GO TO 300
    2 Y=2.*X+3.5
      GO TO 400
```

```
  3 Y=X*X-2.*X-4.
    GO TO 400
500 STOP
    END
```

Za ulazne podatke date u tabeli 12.1.1, rezultati se dobijaju u obliku:

Tabela 12.1.1

k	x
2	1,25
1	4,00
3	2,20
1	-1,00

```
BETA =  0.600000E 01
ALFA =  0.170000E 02
GAMA = -0.356000E 01
ALFA =  0.200000E 01
```

Elementima niza A dodeljene su kao početne vrednosti, alfabetski podaci ALFA, BETA i GAMA, i štampanje teksta izvršeno je pozivanjem odgovarajućeg elementa niza A.

12.2. Promene FORMAT-naredbe za vreme izvršavanja programa

Prema dosadašnjem izlaganju FORMAT-naredba se piše u programu i njen oblik se ne može menjati za vreme izvršavanja. Medjutim, u nekim slučajevima je pogodno da se ova naredba može menjati za vreme izvršavanja programa. Ovo se može postići na dva načina:

- postavljanjem sadržaja FORMAT-naredbe sa ulaza, i
- postavljanje sadržaja FORMAT-naredbe kao početne vrednosti niza.

12.2.1. Postavljanje sadržaja FORMAT-naredbe sa ulaza

U odeljku 4.8.5 objašnjeno je da se literal u FORMAT-naredbi, koja je pridružena naredbi ulaza, zamenjuje sadržajem odgovarajućeg polja u ulaznom slogu. Prema tome, ako je opisna naredba napisana u obliku

```
j FORMAT('literal')                    (12.2.1)
```

a izvršna naredba ulaza, pomoću koje se unosi novi sadržaj FORMAT-naredbe

```
READ (i,j)                             (12.2.2)
```

tada će izvršnom naredbom (12. 2. 2) biti postavljen novi sadržaj naredbe (12. 2. 1) na predvidjenoj dužini izmedju apostrofa.

Na ovaj način sadržaj FORMAT-naredbe može se menjati proizvoljan broj puta za vreme izvršavanja programa. Medjutim, ove promene se odnose samo na pisani tekst (literal) koji se nalazi izmedju apostrofa.

Tako je ovo pogodan način kada se unose različita objašnjenja koja treba štampati uz rezultate.

12. 2. 2. Postavljanje sadržaja FORMAT-naredbe kao vrednosti niza

Sadržaj FORMAT-naredbe, uključujući spoljnu otvorenu i zatvorenu zagradu, može se postaviti kao vrednost niza sa odgovarajućim brojem elemenata u zavisnosti od broja simbola koje sadrži FORMAT-naredba. Ova vrednost može biti postavljena kao početna vrednost niza ili dodeljena elementima niza sa ulaza. Naredbom

READ(i, ime)lista (12. 2. 3)

odnosno

WRITE(i, ime)lista (12. 2. 4)

gde je

ime - naziv niza čijim elementima je dodeljena vrednost sadržaja FORMAT-naredbe.

Primer

Tako raniji primer na kraju odeljka 12. 1. 5, može biti zapisan bez FORMAT-naredbe, i tada program ima sledeći izgled

```
      DIMENSION A(3),FORUL(3),FORIZ(5)
      DATA A/'ALFABETAGAMA'/,FORUL/'(I1,F9.4)'/,
     *FORIZ/20H(' ',A4,' = ',E12.5)/
  300 READ(5,FORUL,END=500) K,X
      GO TO (1,2,3),K
    1 Y=X*X+1.
  400 WRITE(6,FORIZ) A(K),Y
      GO TO 300
    2 Y=2.*X+3.5
      GO TO 400
    3 Y=X*X-2.*X-4.
      GO TO 400
  500 STOP
      END
```

Za ulazne podatke date u tabeli 12.1.1, rezultati se dobijaju, kao i ranije, u obliku

```
BETA =  0.60000E 01
ALFA =  0.17000E 02
GAMA = -0.35600E 01
ALFA =  0.20000E 01
```

12.3. Unošenje i izdavanje podataka po njihovom imenu

Unošenje i izdavanje podataka vrši se navodjenjem imena promenljivih, u listi odgovarajućih izvršnih naredbi, i navodjenjem opisa u listi FORMAT-naredbe. Pri ovome se jedna ista informacija o vrsti podataka navodi dva puta. Prvi put je to rečeno preko imena promenljive, a drugi put preko opisa polja u ulaznom, odnosno u izlaznom slogu.

Postoji mogućnost da se unošenje i izdavanje podataka vrši isključivo preko imena promenljivih i nizova. Ovo se postiže naredbom

NAMELIST lista (12.3.1)

gde je

NAMELIST - službena reč,

lista - spisak elemenata koji se medju sobom ne razdvajaju zarezima.

Elementi liste imaju oblik

/ime/lista$_1$ (12.3.2)

gde je

ime - naziv koji se sastoji od jednog do šest alfanumeričkih simbola, od kojih prvi mora biti slovo,

lista$_1$ - spisak imena promenljivih i nizova medju sobom razdvojenih zarezima.

U ovom slučaju naredba ulaza ima oblik

READ(i, ime) (12.3.3)

odnosno u slučaju izlaza

WRITE(i, ime) (12.3.4)

gde je

i - ceo neoznačen broj ili ime celobrojne promenljive, kojim se definiše ulazno-izlazna jedinica,

ime - naziv promenljivih sadržanih u $lista_1$ u (12.3.2).

Kako ime u (12.3.3), odnosno (12.3.4), ukazuje na spisak promenljivih i nizova, to znači da je ovim definisana lista ulazne, odnosno izlazne, naredbe. Opis polja ulaznog, odnosno izlaznog sloga u ovom slučaju nije zadat, ali se zato ulazni, odnosno izlazni podaci moraju pisati u obliku

$$\text{ime lista}_2 \ \& \ \text{END} \qquad (12.3.5)$$

gde je

ime - naziv promenljivih sadržanih u $lista_1$ u (12.3.2),

$lista_2$ - spisak elemenata medju sobom razdvojenih zarezima, i

END - službena reč, koja označava kraj ulaznih podataka.

Elementi $liste_2$ su oblika

$$ime_p = k \qquad (12.3.6)$$

ili

$$ime_n = lista_3 \qquad (12.3.7)$$

gde je

ime_p - ime promenljive, koje mora biti sadržano u $lista_1$,

k - konstanta koja se dodeljuje promenljivoj sa imenom ime_p ,

ime_n - ime niza, koje mora biti sadržano u $lista_1$,

$lista_3$ - spisak konstanti medju sobom razdvojenih zarezima, koje se redom dodeljuju elementima niza čije je ime na levoj strani znaka jednakosti.

Ako je u $lista_3$ više uzastopnih konstanti jednako, onda element liste može imati oblik

$$m*k_1 \qquad (12.3.8)$$

gde je

m - ceo neoznačen broj koji ukazuje na broj ponavljanja konstante,

k_1 - konstanta koja se ponavlja.

Ulazni podaci koji se unose pomoću naredbe (12.1.1) moraju biti bušeni počev od 2. kolone kartice.

Naredba (12.3.1) može se nalaziti bilo gde u programu, ali mora biti ispred prve naredbe u kojoj se koriste imena promenljivih iz liste$_1$ u (12.3.2). Pomoću ove naredbe mogu se unostiti i izdavati vrednosti svih vrsta promenljivih: celobrojne, mešovite, kompleksne ili logičke konstante.

Primer

Zadata su dva kompleksna broja C_1 i C_2, i 10 elemenata niza A sa realnim konstantama obične tačnosti, kao i 3 elementa niza L sa logičkim konstantama. Izračunati C, P i K po formulama

$$C = \frac{C_1}{C_2}$$

$$P = \prod_{i=1}^{10} A_i$$

$$K = L_1 \wedge L_2 \wedge L_3$$

Program na FORTRAN-jeziku ima sledeći izgled:

```
      COMPLEX C,C1,C2
      LOGICAL*1 L(3),K
      DIMENSION A(10)
      NAMELIST/ULAZ/C1,C2,L,A/IZLAZ/C,P,K
      READ(5,ULAZ)
      C=C1/C2
      P=1
      DO 10 I=1,10
   10 P=P*A(I)
      K=L(1).AND.L(2).AND.L(3)
      WRITE(6,IZLAZ)
      STOP
      END
```

Ulazni podaci se pripremaju na karticama, tako da se buše od 2. kolone kartice. Za ulazne podatke

```
&ULAZ C1=(2.,4.2),C2=(-1.,5.23),L=T,T,F,
A=4.,4*1.,5*2.,&END
```

rezultati se štampaju u obliku

```
&IZLAZ
C=.          (0.7.419616,-0.51705462),P=  128.00000     ,K=F
&END
```

13. KORIŠĆENJE SPOLJNIH MEMORIJA

Kao spoljne memorije kod računara najčešće se koriste magnetni disk i magnetna traka. Ove memorije su relativno spore u odnosu na operativnu (feritnu) memoriju računara, ali su zato velikog kapaciteta. I magnetni disk i magnetna traka su po svojoj prirodi medijum na koji se informacija upisuje i sa kojeg se izdaje serijski, i to kod magnetnog diska bit po bit, a kod magnetne trake znak po znak (karakter).

13.1 Magnetni disk

13.1.1. Definisanje podataka

Svi podaci koji će se prenositi na magnetni disk ili izdavati sa diska moraju biti definisani u jednoj grupi podataka, pomoću opisne naredbe

DEFINE FILE lista (13.1.1)

gde je

DEFINE FILE - službena reč,

lista - spisak elemenata medju sobom razdvojenih zarezima.

Elementi u listi (13.1.1) su oblika

g (s, d, f, p) (13.1.2)

gde je

g - ceo neoznačen broj koji predstavlja identifikacioni broj grupe,

s - ceo neoznačen broj koji definiše broj slogova u grupi sa identifikacionim brojem g,

d - ceo neoznačen broj koji definiše maksimalnu dužinu sloga u grupi sa identifikacionim brojem g,

f - slovo L, E ili U koje definiše način prenošenja ili izdavanja podataka, i

p - ime celobrojne promenljive, čija vrednost definiše slog na disku.

Dužina sloga d može se izraziti brojem podregistara ili brojem registara u memoriji čiji se sadržaj izdaje na disk ili postavlja sa diska jednim slogom obrazovanim za ovakvu komunikaciju.

Slovo L definiše da se slog za komunikaciju izmedju diska i unutrašnje memorije obrazuje prema FORMAT-naredbi ili bez ove naredbe. Maksimalna dužina sloga (d) izražava se brojem podregistara u memoriji.

Slovo E definiše da se slog za komunikaciju izmedju diska i unutrašnje memorije obrazuje prema FORMAT-naredbi. Maksimalna dužina sloga (d) izražava se brojem znakova.

Slovo U definiše da se slog za komunikaciju izmedju diska i unutrašnje memorije obrazuje bez upotrebe FORMAT-naredbe. Maksimalna dužina sloga izražava se brojem registara u memoriji.

Posle svakog obraćanja disku vrednost celobrojne promenljive p biva uvećana za jedinicu, čime ukazuje na sledeći slog na disku.

13.1.2. Pozicioniranje glave diska

Da bi se omogućilo pozicioniranje glave diska pre nego što dodje do izvršne naredbe unošenja ili izdavanja informacija sa diska, uvedena je naredba

FIND (g'r) . (13.1.3)

gde je

FIND - službena reč,

g - ceo neoznačen broj ili celobrojna promenljiva čija vrednost predstavlja identifikacioni broj grupe,

r - ceo neoznačen broj ili aritmetički izraz čija vrednost ukazuje na relativan položaj sloga u grupi sa identifikacionim brojem g,

13.1.3. Prenos podataka

13.1.3.1. Upis podataka na disk

Upis podataka iz unutrašnje memorije računara na disk vrši se izvršnom naredbom

WRITE (g'r, j) lista (13.1.4)

gde je:

WRITE - službena reč,

g - ceo neoznačen broj ili ime celobrojne promenljive čija vrednost predstavlja identifikacioni broj grupe,

r - ceo neoznačen broj ili aritmetički izraz čija vrednost ukazuje na relativan položaj sloga u grupi sa identifikacionim brojem, g, i

j - neobavezan parametar, koji ako se navodi može biti obeležje jedne FORMAT-naredbe, ili ime jednog niza, čiji sadržaj odgovara sadržaju jedne FORMAT-naredbe kojom se definišu izlazni podaci,

lista - spisak imena promenljivih i nizova, medju sobom razdvojenih zarezima, čije će se vrednosti prenositi.

13.1.3.2. Izdavanje podatka sa diska

Izdavanje podataka sa diska i njihovo prenošenje u unutrašnju memoriju računara vrši se izvršnom naredbom

READ (g'r, j, ERR=n) lista (13.1.5)

gde je

READ - službena reč,

g - ceo neoznačen broj ili ime celobrojne promenljive čija vrednost predstavlja identifikacioni broj grupe,

r - ceo neoznačen broj ili aritmetički izraz čija vrednost ukazuje na relativan položaj sloga u grupi sa identifikacionim brojem g,

j - neobavezan parametar, koji ako se navodi može biti obeležje jedne FORMAT-naredbe, ili ime jednog niza čiji sadržaj odgovara sadržaju jedne FORMAT-naredbe kojom se definišu izlazni podaci,

n - neobavezan parametar, koji ako se navodi predstavlja obeležje jedne izvršne naredbe na koju se vrši prelazak u slučaju da se otkrije greška na disku za vreme prenošenja podataka u unutrašnju memoriju računara, i

lista - spisak imena promenljivih i nizova, medju sobom razdvojenih zarezima, kojima se dodeljuju brojne vrednosti sa diska.

Primer

Sastaviti program koji elementu $a_{i,j}$ matrice

$$A = \begin{Vmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,56} & \cdots & a_{1,60} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,56} & \cdots & a_{2,60} \\ \vdots & \vdots & & \vdots & & \vdots \\ a_{5,1} & a_{5,2} & \cdots & a_{5,56} & \cdots & a_{5,60} \\ a_{20,1} & a_{20,2} & \cdots & a_{20,56} & \cdots & a_{20,60} \end{Vmatrix} \qquad (13\ 1.6)$$

dodeljuje brojnu vrednost

$$a_{i,j} = 100i+j \qquad (13.1.7)$$

Ovako formiranu matricu upisati na magnetni disk, a zatim preneti elemente

$$\begin{matrix} a_{1,56} & \cdots & a_{1,60} \\ \vdots & & \vdots \\ a_{5,56} & \cdots & a_{5,60} \end{matrix} \qquad (13.1.8)$$

sa diska u unutrašnju memoriju računara i štampati njihove vrednosti.

Program ima sledeći izgled

```
      INTEGER A(20,60),B(60)
      DEFINE FILE 2(20,60,U,K)
      DO 10 I=1,20
      DO 20 J=1,60
   20 A(I,J)=100*I+J
   10 WRITE(2'I)(A(I,J),J=1,60)
      DO 30 N=1,5
      READ(2'N,ERR=500)(B(J),J=1,60)
```

```
30  WRITE(6,40)(B(J),J=56,60)
40  FORMAT(' ',5I6)
500 STOP
    END
```

Elementi matrice A čine grupu podataka sa identifikacionim brojem 2. Svaka vrsta matrice čini jedan slog, pa prema tome grupa je sačinjena od 20 slogova. Svaki slog sadrži 60 elemenata jedne vrste matrice. Upis na disk vrši se vrsta po vrsta matrice, što čini ukupno 20 slogova za upis. Izdavanje sa diska vrši se u pet slogova, čiji su sadržaji redom prva, druga, treća, četvrta i peta vrsta. Iz svake vrste matrice koja se prenese sa diska u unutrašnju memoriju računara, štampa se zadnjih 5 elemenata vrste od 56. do 60. elementa. Tako štampani dokument ima sledeći izgled

156	157	158	159	160
256	257	258	259	260
356	357	358	359	360
456	457	458	459	460
556	557	558	559	560

U slučaju da se pri prenosu podataka sa diska u unutrašnju memoriju računara otkrije greška, tada dolazi do prelaska na naredbu zaustavljanja (sa obeležjem 500), čime se prekida dalji rad po programu.

13.2. Magnetna traka

13.2.1. Prenos podataka

13.2.1.1. Upis podataka na magnetnu traku

Upis podataka iz unutrašnje memorije računara na magnetnu traku vrši se izvršnom naredbom

WRITE(i) lista (13.2.1)

gde je

WRITE - službena reč,

i - ceo neoznačen broj ili celobrojna promenljiva, čija vrednost ukazuje na jedinicu magnetne trake,

lista - spisak imena promenljivih i nizova medju sobom razdvojenih zarezima, čije se brojne vrednosti upisuju na magnetnu traku.

Naredbi (13.2.1) nije pridružena opisna FORMAT-naredba, jer oblik (13.2.1) pretpostavlja prenošenje podataka u internom kodu računara, tako da FORMAT-naredba nije potrebna.

Pored oblika (13.2.1) može se koristiti i oblik

WRITE (i, j) lista (13.2.2)

koji ima isto značenje kao i naredba izlaza opisana u odeljku 4.5., s tim što sada i ukazuje na jedinicu magnetne trake. Prema tome, naredbi (13.2 2.) pridružuje se FORMAT-naredba sa obeležjem j.

13.2.1.2. Izdavanje podataka sa magnetne trake

Izdavanje podataka sa magnetne trake i njihov upis u unutrašnju memoriju računara vrši se izvršnom naredbom

READ(i) lista (13.2.3)

gde je

READ - službena reč,

i - ceo neoznačen broj ili celobrojna promenljiva, čija brojna vrednost ukazuje na jedinicu magnetne trake,

lista - spisak imena promenljivih i nizova medju sobom razdvojenih zarezima, kojima se dodeljuju brojne vrednosti sa magnetne trake.

Naredbi (13.2.3) nije pridružena opisna FORMAT-naredba, jer oblik (13.2.3) pretpostavlja da se sa trake izdaju podaci koji su upisani u internom kodu računara, naredbom (13.2.1).

Ako su dužine liste i sloga na magnetnoj traci jednake, tada se sve izdate informacije sa trake upisuju u ukazane memorijske registre imenima u listi naredbe (13.2.3). Ako je dužina liste manja od dužine sloga na traci, tada se prenosi samo ona dužina sloga koja odgovara listi. Medjutim, ako je dužina liste veća od dužine sloga na traci, tada se ovakva naredba (13.2.3) neće izvršiti i dolazi do prekida rada po programu.

Pored oblika (13.2.3) može se koristiti i oblik

READ(i, j) lista (13.2.4)

koji ima isto značenje kao i naredba ulaza opisana u odeljku 4.4.2, s tim što i sada ukazuje na jedinicu magnetne trake. Prema tome, naredbi (13.2.4) pridružuje se FORMAT-naredba sa obeležjem j.

13.2.2. Oznaka kraja grupe podataka

Kraj grupe podataka na magnetnoj traci označava se posebnim znakom koji se upisuje naredbom

END FILE g (13.2.5)

gde je

END FILE - službena reč,

g - ceo neoznačen broj ili celobrojna promenljiva, čija brojna vrednost predstavlja identifikacioni broj grupe podataka.

13.2.3. Premotavanje magnetne trake

13.2.3.1. Vraćanje trake na prethodan slog

Vraćanje trake na prethodan slog postiže se naredbom

BACKSPACE g (13.2.6)

gde je

BACKSPACE - službena reč

g - ceo neoznačen broj ili celobrojna promenljiva, čija brojna vrednost predstavlja identifikacioni broj grupe podataka.

13.2.3.2. Vraćanje trake na početak grupe

Vraćanje trake na početak grupe postiže se naredbom

REWIND g (13.2.7)

gde je

REWIND - službena reč,

g - ceo neoznačen broj ili celobrojna promenljiva, čija brojna vrednost predstavlja identifikacioni broj grupe podataka.

LITERATURA

[1] Fredric Stuart: FORTRAN Programming, John Wiley & Sons, Inc., New York, 1969.

[2] John Blatt: Introduction to FORTRAN IV Programming: Using the Watfor Compiler, Goodyear Publishing Company, Pacific Palisades, California, 1968.

[3] IBM System/360, FORTRAN IV Language (C28-6515).

[4] Milan Žokalj: FORTRAN IV, Koordinacioni odbor korisnika mašina za obradu podataka Jugoslavije, 1969.

[5] IBM System/360, FORTRAN IV Library Subprograms (C28-6596-2)

SAVREMENA RAČUNSKA TEHNIKA I NJENA PRIMENA

U ovoj seriji Matematičkog instituta dosada su publikovane sledeće knjige:

1. *Nedeljko Parezanović*
   Algoritmi i programski jezik FORTAN IV,
   Beograd, 1972., str. 272.
2. *Pavle Pejović i Nedeljko Parezanović*
   Analogni elektronski računari i njihova primena
   Beograd, 1972.

U pripremi za štampu:

3. *Jurij Stepanenko*
   Dinamika prostornih mehanizama, Beograd, 1972.
4. *Dragiša Stojanović*
   Ekonomsko-matematički modeli linearnog programiranja,
   Beograd, 1972.
5. *Mirko Stojaković*
   Algoritmi i automati, Beograd, 1972.

Nedeljko Parezanović

ALGORITMI I PROGRAMSKI JEZIK FORTRAN IV

Jezičku redakciju izvršila	Branka ŽIVKOVIĆ
Nacrt za korice i tehnički urednik	Milan ČAVČIĆ
Rukopis za snimanje kucala . . .	Živka KIRIN
Korekture izvršili	Nedeljko PAREZANOVIĆ i Milan ČAVČIĆ
Tiraž	2000 primeraka
Štampanje završeno	juna 1972.