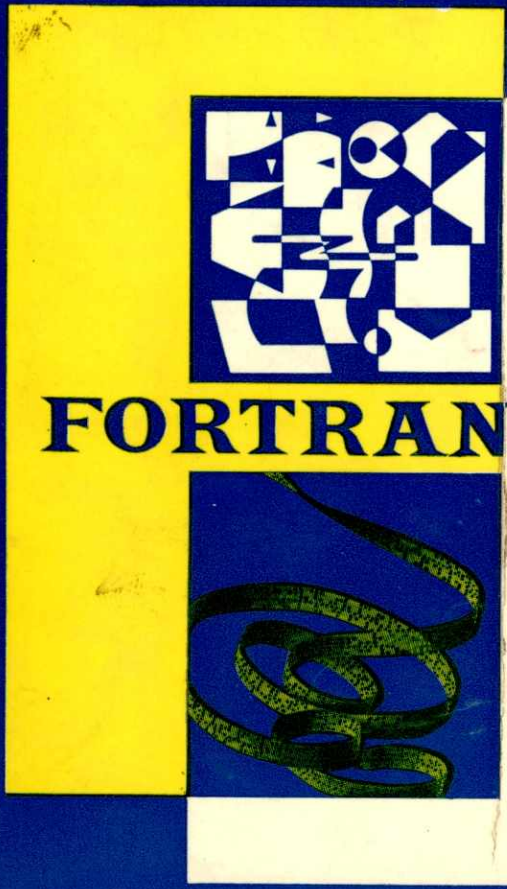
MILAN ŽOKALJ, dipl. inž.



FORTRAN IV

Milan Žokalj, dipl. inž.

FORTRAN IV

Def. : Slog = tekst na 1 kartici (V)

kod READ

Slog kod WRITE = 1 red

1969

Izdavač:
KOORDINACIONI ODBOR KORISNIKA MAŠINA ZA OBRADU PODATAKA JUGOSLAVIJE

Stručna recenzija:
Dr NEDELJKO PAREZANOVIĆ
Mr MIOMIR TODOSIJEVIĆ

Naslovna strana:
Akademski slikar MILAN UZUN

Ilustracije:
MIODRAG ŠURJANAC, stud. arh.

Meteur:
VIKTOR ANDRIĆ

Štampa:
NOVINSKO-IZDAVAČKO I ŠTAMPARSKO PODUZEĆE »VJESNIK«, ZAGREB

PREDGOVOR

Prvo izdanje udžbenika, čiji je naslov bio FORTRAN-programiranje, rasprodato je još početkom prošle godine. Da se zadovolji sve veće interesovanje za FORTRAN, izrađen je ovaj udžbenik, u kome je zadržana podela materije iz prethodnog izdanja, ali je čitava knjiga iz osnova nanovo napisana.

Prva glava znatno potpunije obuhvata osnovne pojmove o elektronskom cifarskom računaru. Dato je više slika koje ilustruju principe glavnih organa računara, kôdove i mašinske nosioce informacija. Sistem programiranja (»softver«), koji je kod savremenih računara jako razvijen, opisan je po svojim glavnim logičkim komponentama, tako da i neupućen čitalac može sagledati mesto i ulogu jezika FORTRAN u sklopu celog sistema. Programiranje je bliže definisano kao jedna od faza rada pri rešavanju problema. Istaknut je hronološki red svih faza jer je primećena tendencija da se prethodnim fazama ne poklanja potrebna pažnja, a posebno se one često pogrešno smatraju delom faze programiranja.

Potpuno nova materija u ovoj knjizi izložena je u glavi o logičkim izrazima i naredbama. Ti elementi jezika predviđeni su standardom za potpuni FORTRAN IV i imaju široku primenu u programiranju.

Potprogrami su kao programske jedinice bili poznati i pre FORTRAN-a IV, međutim na postojećim računarima u zemlji nisu se mogli koristiti pre pojave računara tzv. treće generacije. Toj materiji je sada posvećen veći broj primera uz objašnjenje definicija, a ona je još posebno ilustrovana primerom jednog programa, koji sa potprogramima rešava sisteme linearnih algebarskih jednačina.

Glava sa primerima rešenih zadataka ima u ovoj knjizi potpuno nov sadržaj, a takođe i način izlaganja. Primeri su birani tako da po sadržaju budu interesantni čitaocima iz raznih stručnih oblasti. Tako se, na primer, program za blankiranje polja u štampanim tabelama ili program za grafičko predstavljanje toka funkcije mogu u principu iskoristiti u bilo kojem programu. Najsloženiji zadatak prikazan je na kraju: sistem linearnih jednačina podjednako je potreban pri rešavanju geodetskih problema kao i u ekonomskim modelima. Podela na jedan program sa više potprograma služi, osim toga, kao dopuna materije o potprogramima, koja je izložena u posebnoj glavi.

Da bi čitaocu bez ikakvog iskustva bila jasnija praktična strana rešavanja programiranog zadatka u računskom centru, pored opisa tog postupka na više mesta u tekstu, svi rešeni zadaci (Glava 11) ilustrovani su slikama ulaznih podataka, dijagrama toka, FORTRAN-programa i odštampanih rezultata. Svi programi su provereni na računaru IBM 360/44, ali su najvećim delom pisani sa repertoarom naredbi koje prihvataju i FORTRAN-prevodioci manjih računara.

Autor se zahvaljuje na korisnim savetima svojim saradnicima u Centru za automatsku obradu podataka (CAOP) Saveznog zavoda za statistiku, drugovima Radivoju Đorđeviću, Nenadu Pjeviću i Branku Todoroviću. Posebnu zahvalnost za pregled celokupnog teksta i dragocene stručne, metodske i terminološke primedbe autor duguje Dr Nedeljku Parezanoviću, šefu Računskog centra Matematičkog instituta u Beogradu, kao i Mr Miomiru Todosijeviću, saradniku CAOP-a, koji je

osim toga i detaljno pregledao mnogobrojne primere u tekstu i u glavi sa rešenim zadacima.

Beograd, april 1969. *AUTOR*

IZ PREDGOVORA »FORTRAN-PROGRAMIRANJU«

Uvođenjem automatizacije na bazi elektronskih računara u poslovanje nekih naših većih ustanova i privrednih organizacija, kod nas se pre nekoliko godina pojavila potreba za jednom novom specijalnošću — programerima elektronskih računara. Prvi elektronski cifarski računar sa unutrašnjim programom u našoj zemlji, IBM 705, počeo je da radi u Saveznom zavodu za statistiku u jesen 1960. godine, u prvo vreme samo na obradi velikih popisa koji su u to vreme sprovođeni.

Zbog potrebe za rešavanjem analitičkih problema, u Elektronskom centru SZS počelo je 1962. godine izučavanje sistema FORTRAN, o kojem je od proizvođača računara dobijena potrebna stručna literatura i programski materijal. U saradnji sa programerima-matematičarima Radivojem Đorđevićem i Radivojem Gavrićem, autor ove knjige je uveo FORTRAN-programiranje u praksu Elektronskog centra SZS. Prvi analitički zadatak, rešen pomoću FORTRANa, bilo je izračunavanje pokazatelja međusektorskih odnosa u privredi Jugoslavije. To je ujedno bila i prva praktična primena analize »input-output«, od kog vremena Savezni zavod za statistiku redovno vrši tu analizu i objavljuje njene rezultate.

Prvi kurs FORTRAN-programiranja održan je 1963. godine u Saveznom zavodu za statistiku za grupu inženjera i matematičara, saradnika beogradskih ustanova i preduzeća. Želja Saveznog zavoda za statistiku je bila da se stručnjaci raznih oblasti upoznaju sa savremenim načinom rešavanja naučno-istraživačkih, projektantskih i analitičkih zadataka. Kurs je vodio dr inž. Borislav Džodžo, naučni saradnik mašinskog instituta »Vladimir Farmakovski« iz Beograda.

Od održavanja prvog kursa, znatno se povećao broj računara u zemlji koji imaju FORTRAN-prevodioce. Istovremeno je rastao i interes stručnjaka u institutima i privrednim organizacijama za korišćenje elektronskih cifarskih računara. Rezultat je ova knjiga, koja predstavlja sintezu pedagoškog iskustva stranih autora, FORTRAN-priručnika za više tipova računara i praktičnog rada autora sa programerima Elektronskog centra SZS.

Beograd, novembar 1966. *AUTOR*

SADRŽAJ

UPUTSTVA ČITAOCU

Udžbenik FORTRAN IV pisan je sa ciljem da posluži kao tekst za samoučenje, ali se isto tako može upotrebiti i kao literatura uz kurseve o ovoj materiji, koji se održavaju pri računskim centrima. Takođe, može se koristiti i kao podsetnik pri praktičnom radu na programiranju.

U tekstu su uz definicije i pravila dati mnogobrojni primeri delova programâ ili čitavih programâ, primenjeni za rešavanje raznovrsnih zadataka. Kao što je podvučeno u odeljku 1.4, cilj ovog udžbenika je isključivo da izloži i objasni pravila FORTRAN-programiranja, tako da kriterijum izbora numeričkog postupka nije objašnjavan niti se usvojeni postupak preporučuje kao najbolji. Stručni čitalac će moći sam da napiše bolji program, na bazi prikladnijeg postupka i boljeg poznavanja prirode problema.

Naročito se preporučuje da se, uporedo sa izučavanjem gradiva, obrati dovoljno pažnje rešavanju zadataka, koji su dati u »Vežbama« na kraju svake glave. Ti zadaci su često dopuna izlaganja, a ne samo ilustracija pređene materije. Za karakteristične zadatke data su rešenja u prilogu, na kraju knjige.

Kada posle savlađivanja izloženog gradiva pristupi pisanju programa za realizaciju na nekom računaru, čitalac treba da ima u vidu da u FORTRAN-prevodiocima nekih računara postoje minimalne razlike u odnosu na pravila pisanja programa, izložena u ovoj knjizi. Zato uvek treba prethodno proučiti FORTRAN-priručnik određenog tipa računara, saznati sastav njegovih organa u konkretnom računskom centru i sve drugo što je specifično za dati računar (tabela standardnih funkcija itd.). Na izvesne osnovne razlike čitalac će biti upozoren već u tekstu ove knjige, tako da svoj program može uvek lako prilagoditi konkretnom računaru.

Autor duguje izvinjenje što zbog obima materije nije mogao da nađe i praktično isproba i prikaže izvesne detalje u pravilima jezika FORTRAN IV, koje obezbeđuju samo FORTRAN-prevodioci velikih računara. To se naročito odnosi na grupu naredbi i pravila za potprograme i specifikacije u naredbi FORMAT. Za dopunu, čitalac se upućuje na FORTRAN-priručnik računara koji će koristiti. Međutim, gradivo izloženo i ilustrovano primerima u ovoj knjizi je standardno i dovoljno za sve praktične potrebe.

Glava 1

SAVREMENI ELEKTRONSKI CIFARSKI RAČUNAR

1.1 *UVOD*

Savremeni elektronski cifarski (digitalni) računar je složena računska mašina, čiji zadatak nije samo numeričko rešavanje komplikovanih i opsežnih matematičkih zadataka nego i obrada informacija u najširem smislu. Da li će ta mašina biti računar za rešavanje matematički formulisanih problema iz raznih oblasti nauke i tehnike, mašina za automatsku obradu masovnih informacija ili tzv. procesni računar koji upravlja procesom proizvodnje — zavisi od potrebe korisnika. Veoma često, isti primerak računara je i jedno i drugo i treće, dakle univerzalna računska mašina za obradu podataka.

Univerzalni cifarski računar je neobično brza i fleksibilna mašina za računanje, manipulisanje i obradu informacija. I podaci koji će se obrađivati i instrukcije za rad unose se u memoriju računara i mogu se lako preneti ili brisati iz memorije. Računar može da vrši samo proste aritmetičke operacije, tj. sabiranje, oduzimanje, množenje i deljenje. On takođe može da obavlja proste logičke operacije, tj. da razlikuje nulu od značajnog broja i broj veći od nule od broja manjeg od nule. Ipak, kombinovanjem ovih skromnih sposobnosti računara sa njegovom elektronskom brzinom i automatskim načinom rada, računar može efikasno da obavlja takve zadatke kao što su komplikovana naučna izračunavanja, vođenje kompleksnih knjigovodstvenih i informacionih kartoteka i simulacija funkcionisanja složenog industrijskog preduzeća.

Danas postoji veliki broj različitih tipova cifarskih elektronskih računara. Među njima postoje velike razlike u veličini, ceni i nameni. Ono što im je svima zajedničko, može se formulisati sa neko-

liko atributa savremenog cifarskog računara: *univerzalni, cifarski, elektronski* računar, sa *unutrašnjim programom.*

Kada se za računar kaže da je digitalni ili *cifarski,* time se naglašava da on operiše sa diskretnim veličinama, ciframa, a ne sa kontinualnim fizičkim veličinama kao *analogni* računar. Digitalni računar, slično stolnoj računskoj mašini, operiše sa brojevima po pravilima aritmetike, drugim rečima operacije se izvode nad diskretnim veličinama. Na primer, kada se na binarne brojeve četiri (100) i jedan (001) primeni operacija oduzimanja, koja se u cifarskim računarima realizuje po pravilima oduzimanja binarnih brojeva, dobija se kao rezultat binarni broj tri (011). Tačnost operacija kod cifarskog računara utoliko je veća ukoliko je veći broj cifara operanada.

Kod analognog računara, vrednosti koje inače iskazujemo brojevima srazmerne su ili analogne nekoj fizičkoj veličini, kao što su na primer napon ili struja. Jednostavan primer analognog računara imamo u logaritmaru, logaritamskom računalu, kod kojeg je ta analogna fizička veličina — dužina. Sabiranjem dve dužine, srazmerne brojevima koje predstavljaju, dobija se treća dužina koja je srazmerna rezultatu, u ovom slučaju proizvodu ta dva broja. Analogne fizičke veličine uvek su po svojoj prirodi kontinualne. Tačnost analognih računara zavisi od konstrukcije, podešenosti i subjektivne veštine u čitanju rezultata. Ona je uvek znatno manja od tačnosti cifarskog računara.

Pojam *elektronski,* upotrebljen kao atribut računara, ima značaja samo po tome što karakterišu brzinu njegovog rada, koja je ogromna. Broj operacija u sekundi ide na nekoliko stotina hiljada, a kod najbržeg poznatog računara je približno tri miliona*. Elektronski računar, međutim, nije elektronski u celini; mehanizmi koji pokreću magnetne trake i diskove ili zupčanike i poluge u drugim uređajima, zbog inercije njihovih masa još uvek su spori. Važno je da su najmasovnije operacije na principu elektronskih elemenata.

Brzina je faktor koji najviše utiče na cenu računara, zato je ona kod raznih računara različita iako je opšti princip rada isti. Ona se kod računara obično izražava u mikrosekundama za operaciju ($1\ \mu s = 10^{-6}$ s), što je samo druga mera za istu karakteristiku (gore smo naveli broj operacija u sekundi).

Univerzalnost računara ogleda se u njegovoj upotrebljivosti za najrazličitije zadatke. Nasuprot takvom računaru, računar za specijalnu namenu ima u sebi »ugrađen« jedan nepromenljiv program,

* Računar CDC 6600

po kojem rešava samo jednu određenu vrstu zadataka. Kao i svaka specijalno konstruisana mašina, on može biti korisno i ekonomično sredstvo samo ako se zadatak iste vrste rešava ogroman broj puta. Savremeni cifarski elektronski računar je, međutim, univerzalna mašina; njena odlika je raznovrsnost zadataka koje može da rešava. Kada su programi pripremljeni, prelaz sa jednog zadatka na drugi obavlja se elektronskom brzinom.

Unutrašnji program je najznačajnija karakteristika savremenih računara iako je ovde predstavljamo kao poslednju. Ovaj izraz znači smeštanje celog programa u memoriju računara, a programom nazivamo potpun niz detaljnih instrukcija koje će računar izvršiti u toku rešavanja zadatka. Te instrukcije su kodirane u obliku i na način koji je specifičan za konstrukciju svakog računara. Svaka instrukcija nalaže računaru da izvrši jednu od operacija za koje je konstruisan. Računar automatski izvršava svaku instrukciju onim redom kojim se ona javlja u programu.

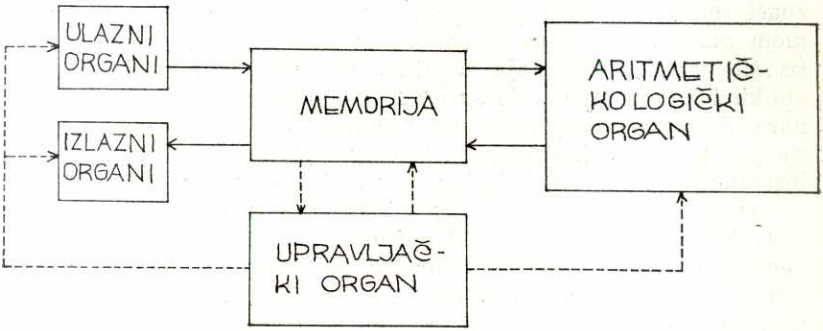
Određeni deo ukupnog broja instrukcija u programu su takve instrukcije koje zahtevaju od računara da vrši operacije nad *podacima;* podaci su, kao i instrukcije, kodirani na način koji je određen konstrukcijom računara. Na primer: (1) čitati podatke sa spoljnog nosioca informacija pomoću odabranog ulaznog uređaja i preneti ih u memoriju računara, (2) obraditi ove podatke, prenoseći ih iz memorije u druge organe računara u svrhu preuređivanja, računanja ili logičkog ispitivanja, (3) formirati i smestiti u memoriju rezultate obrade i (4) preneti rezultate iz memorije u odabrani izlazni uređaj u cilju njihovog registrovanja na pogodan spoljni nosilac informacija.

U programu se uvek nalaze i takve instrukcije koje nalažu računaru da izvršava izvesne operacije nad samim *instrukcijama.* Na primer: (1) ponoviti izvršavanje istog niza instrukcija, (2) modifikovati neku instrukciju vršeći nad njom jednu aritmetičku operaciju ili zameniti jednu instrukciju drugom, (3) koristeći sposobnost računara da obavlja proste logičke operacije, izabrati od nekoliko nizova jedan određeni niz instrukcija za izvršavanje. Znači, računar može da modifikuje, ponavlja i bira instrukcije u okviru jednog programa. Ovo je principijelno moguće zato što su i instrukcije smeštene u memoriju zajedno sa podacima i što su kodirane na isti način kao i podaci.

Unutrašnji program, pored toga što je uslov za fleksibilnu i automatsku obradu podataka po *jednom programu,* omogućuje i viši stepen automatizma — automatsko smenjivanje u memoriji *jednog programa drugim.*

1.2 *FUNKCIONALNA PODELA RAČUNARA*

Pri rešavanju problema, zadatog u vidu programa i podataka za računanje odnosno obradu, elektronski digitalni računar obavlja sledeće funkcije: unošenje ili *ulaz* informacija, čuvanje ili *pamćenje* informacija, *upravljanje radom svojih uređaja, aritmetičku i logičku* obradu i, na kraju, izdavanje ili *izlaz* informacija. Sve te funkcije su šematski prikazane na slici 1.1, koja predstavlja opštu organizaciju elektronskog cifarskog računara, bez obzira na tip i veličinu.



Slika 1.1 Opšta organizacija cifarskog elektronskog računara

Očigledna je paralela sa čovekom, koji ručno obavlja računski rad na stolnoj računskoj mašini. Na ovoj šemi čovek je zamenjen upravljačkim organom računara, stolna računska mašina aritmetičko-logičkim organom, olovka i hartija zamenjeni su ulaznim i izlaznim organima, a tablice kojima se čovek služi pri radu i njegova sopstvena memorija memorijom mašine.

Pune linije simbolički označavaju puteve kojima se kreću podaci, a isprekidane linije puteve kojim se prenose komande ili signali upravljanja.

1.2.1 *KÔD I MEDIJUM*

Sve informacije sa kojima radi računar predstavljaju se pomoću sistematizovanog skupa simbola koji se naziva *kôd.*

Sredstvo na koje je informacija ubeležena kodom, radi ulaza u računar ili posle izlaza iz računara, zove se spoljni nosilac informacije ili spoljni *medijum.*

Postoje različiti kodovi koji se koriste kod računara, pa čak i jedan isti računar po pravilu se služi sa nekoliko kodova. Jedan

kôd služi za upisivanje informacije na ulazni medijum, drugi za pamćenje informacije u memoriji, treći za komande upravljanja itd. Sa malim izuzecima, svi ti kodovi su tzv. binarnog ili dualnog tipa, uslovljenog prirodom elektronskih kola i fizičkih sredstava za pamćenje informacije. Računar je sastavljen od električnih kola ili krugova u kojima struja može da teče ili da ne teče, od relea čiji kontakti mogu biti otvoreni ili zatvoreni, od tranzistora i elektronki koje propuštaju ili blokiraju električne impulse, od feromagnetnih elemenata koji su namagnetisani pozitivno ili negativno itd. Sva ta sredstva svojom prirodom nameću upotrebu kodova koje zovemo dualnim ili binarnim.

Binarni sistem brojeva se sastoji od samo dva simbola ili dve cifre, 0 i 1. U tom sistemu jedinica označava postojanje, a nula odsustvo vrednosti. Brojevi se formiraju od cifara slično kao u dekadnom sistemu: idući s desna na levo poziciona vrednost cifara raste. U tabeli 1.1 dati su primeri koji ilustruju binarni brojni sistem.

TABELA 1.1

	Dekadni sistem (osnova 10)				Binarni sistem (osnova 2)						
Pozicija	3	2	1	0	6	5	4	3	2	1	0
Stepen	10^3	10^2	10^1	10^0	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Vrednost	1000	100	10	1	64	32	16	8	4	2	1
Brojevi				0							0
				1							1
				2						1	0
				3						1	1
				4					1	0	0
				5					1	0	1
				6					1	1	0
				7					1	1	1
				8				1	0	0	0
				9				1	0	0	1
			1	0				1	0	1	0
			3	2			1	0	0	0	0
			6	4		1	0	0	0	0	0
		1	0	0		1	1	0	0	0	0
		1	2	7		1	1	1	1	1	1

Svaka informacija koja ima brojnu vrednost može se izraziti brojem binarnog sistema. No kako veliki brojevi u tom sistemu zahtevaju mnogo cifara, često se pribegava kombinovanju, pa se binarno kodiraju samo dekadne cifre od 0 do 9. Tako su dekadne cifre predstavljene grupama po četiri binarne cifre. Na primer, dekadni broj 853 može se izraziti kao 1000 0101 0011. Ovakav način zove se binarno kodiranje dekadnih cifara (BCD).

Dekadna vrednost	8	5	3
Binarno kodiranje	1 0 0 0	0 1 0 1	0 0 1 1
Vrednost binarne pozicije	8 4 2 1	8 4 2 1	8 4 2 1

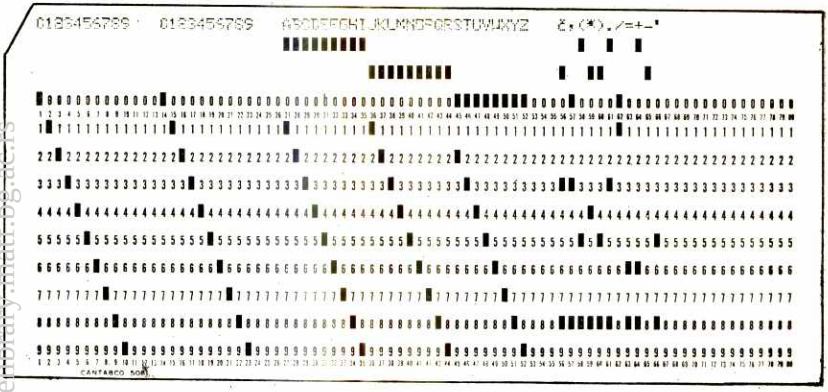
Opisujući dalje, prema slici 1.1, pojedine organe i uređaje računara, upoznaćemo razne kodove i medijume za registrovanje informacija kao i njihovu primenu u računaru.

1.2.2 *ULAZNI ORGANI*

Ulazni organ je mašina koja čita i unosi informacije u memoriju računara. Informacija može biti podatak ili instrukcija, a mora da bude kodirana tako da je mašina može prihvatiti. Svi ulazni organi mogu se svrstati u tri grupe: (a) oni koji zahtevaju da informacija bude prethodno ubeležena na neki mašinski nosilac, (b) oni pomoću kojih čovek ručno unosi informaciju i (c) takvi koji čitaju informaciju direktno u izvornom obliku. Jedan računar može da ima više istih ili raznih ulaznih organa.

Nosioci informacija ili medijumi, sa kojima može raditi savremeni računar, jesu bušene ili perforirane kartice, bušena papirna traka, magnetna traka, magnetni disk i dokumenti štampani magnetskim pismom ili pismom za optičko mašinsko čitanje. Informacija na tim nosiocima mora biti ubeležena takvim kodom da se može preobratiti u električne impulse.

Kartice su pravougaonog oblika standardne veličine, a izrađuju se od hartije visokih električkih i optičkih svojstava. Polja, koja se dobijaju podelom kartice na redove i kolone, buše se pomoću posebnih mašina, tzv. bušilica. Određene kombinacije bušenih rupica predstavljaju kodove za pojedine tipografske znake. Postoje dva tipa standardnih kartica: sa 80 kolona i pravougaonim rupicama i sa 90 kolona (2 × 45) i okruglim rupicama. 80-kolonska kartica je češća u praksi, pa se u daljem izlaganju isključivo govori o toj kartici (vidi sliku 1.2).

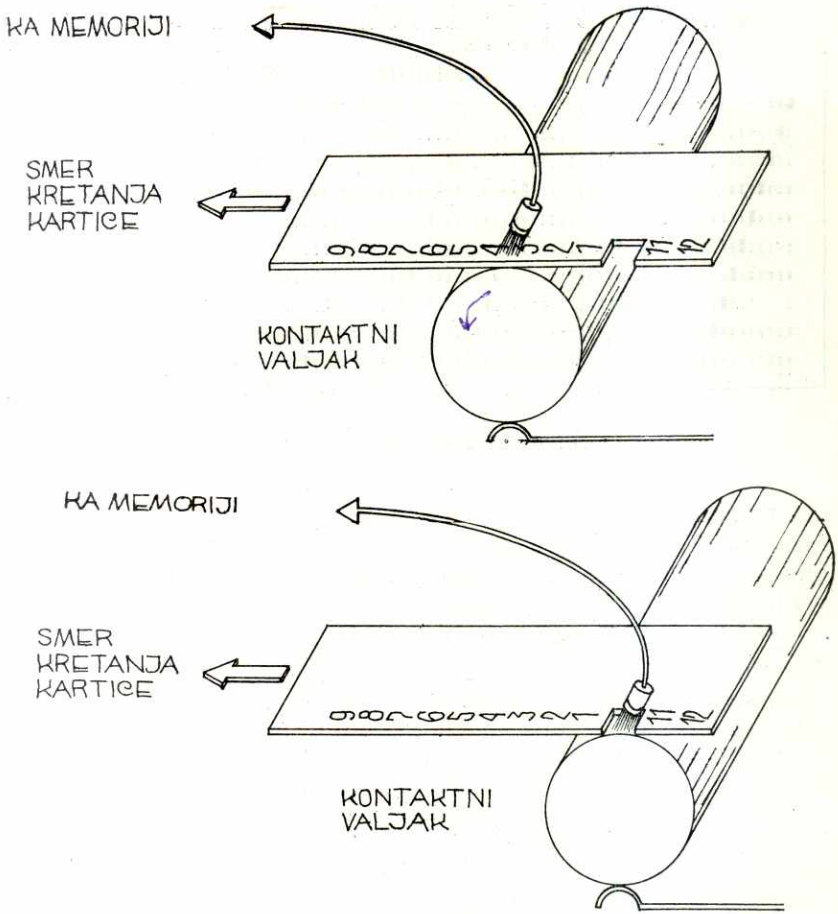


Slika 1.2 80-kolonska kartica

U 80-kolonsku karticu informacije se mogu kodirati na tri razna načina: binarne vrednosti po redovima (redni binarni kôd), binarne vrednosti po kolonama (kolonski binarni kôd) i dekadne vrednosti po kolonama (standardni kôd kartice).

Na slici 1.2 prikazana je 80-kolonska kartica, izbušena standardnim kodovima kartice. Ti kodovi postoje za tri vrste znakova: cifre, slova i specijalne znake. U svaku kolonu može se bušiti kôd svakog znaka, ali u jednu kolonu može stati uvek samo jedan znak. Kodovi za cifre sastoje se od samo jedne rupice u numerisanom delu kolone. Kodovi za slova imaju po dve rupice, jednu u numeričkom i jednu u tzv. zonskom delu (zonski deo sačinjavaju redovi 12 i 11, koji nisu numerisani, i red 0). Kodovi za specijalne znake sastoje se od jedne, dve ili tri rupice.

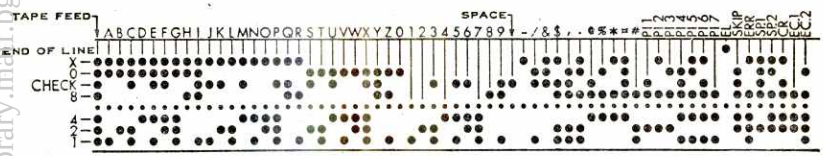
Čitač kartica je ulazni organ ili uređaj koji čita informacije sa kartica, pretvara ih u električne impulse i šalje u memoriju računara. Postoje dva principa čitanja: fotoelektrički i kontaktni (galvanski). U toku kretanja kartice kroz čitač, električni impulsi nastaju prolaskom svetlosnog mlaza kroz rupice ili kontaktom četkice i valjka. U nekim čitačima kartice se čitaju serijski (uzduž), a u nekima paralelno (popreko). Paralelni način je više rasprostranjen i takav čitač obično u istom prolazu čita karticu na dvema stanicama za čitanje, pa se pročitani sadržaji upoređuju radi kontrole. Svih 80 kolona čitaju se istovremeno. Tipična brzina rada čitača kartica je 1000 kartica u minuti.



Slika 1.3 Princip mašinskog čitanja kartice

Na slici 1.3 prikazan je kontaktni ili galvanski princip čitanja kartice. Vidi se samo jedna od 80 kontaktnih četkica, što odgovara jednoj koloni kartice. Impuls nastaje pri nailasku rupice između četkice i valjka. Numerička i zonska vrednost impulsa određuju se njihovim vremenskim rastojanjem od jednog sinhronizacionog impulsa, koji nastaje automatski na početku ciklusa kartice.

Papirna traka je beskrajna traka od hartije, širine pola do jednog inča (12,7 do 25,4 mm). Uzduž je podeljena na *kanale*, a popreko na kolone; preseci kanala i kolona su pozicije koje se buše okruglim rupicama pomoću posebnih mašina — perforatora. Svaki znak informacije ima svoju jedinstvenu kombinaciju rupica ili kôd i zauzima na traci jednu kolonu.

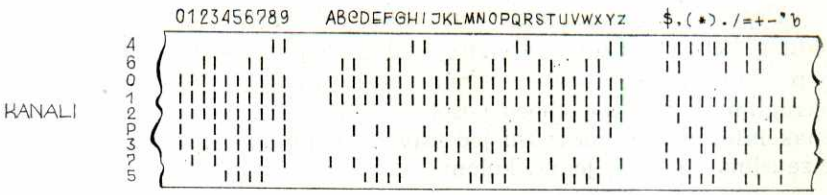


Slika 1.4 8-kanalna papirna traka

Na slici 1.4 vidimo jedan komad papirne trake izbušen 8-kanalnim kodom IBM. Numerički kanali 1, 2, 4 i 8 imaju binarne vrednosti, od kojih se bušenjem sastavljaju dekadne cifre 0 do 9. Bušenjem u kanalima O i X, kombinovano sa numeričkim kanalima, formiraju se kodovi za slova i specijalne znake. Rupica u kanalu END OF LINE znači kraj jednog sloga, tj. niza znakova koji zajedno čine jednu logičku celinu. Kanal CHECK sadrži rupicu samo ako je broj svih ostalih rupica u toj koloni paran. Ovaj kanal služi za tzv. kontrolu parnosti, koja se u računaru vrši brojanjem rupica u svakoj koloni. Ispravnim se smatraju svi kodovi kod kojih je ukupan broj rupica, zajedno sa onom u kanalu CHECK, neparan.

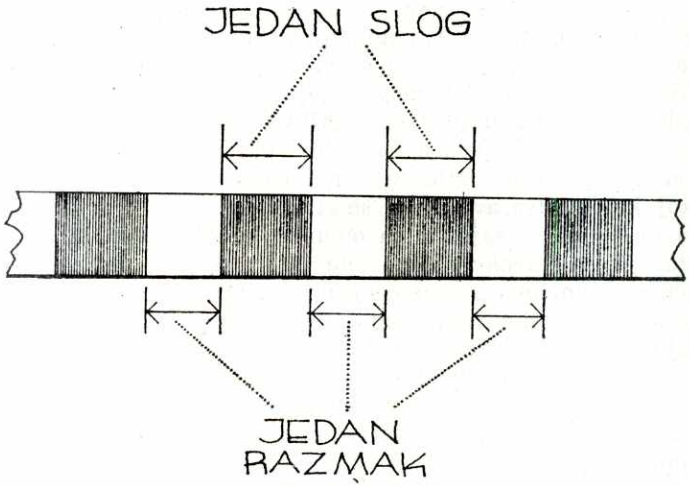
Čitač papirne trake obično čita informacije sa trake na fotoelektričnom principu. S jedne strane upravljen je na traku mlaz svetlosti; pri nailasku rupice mlaz prodire kroz traku i osvetljava fotoćeliju na drugoj strani, a fotoćelija stvara električni impuls. Tako se čitaju redom sve kolone, tj. znak po znak. Radi lakšeg rukovanja, duge trake se namotavaju na koturove od plastičnog materijala. Tipična brzina čitača papirne trake je 1500 znakova u sekundi.

Magnetna traka je fizički slična traci za magnetofon, dok je princip upisivanja informacije logički sličan onom kod papirne trake. Osnova trake je od plastičnog materijala, širine 1/2 inča (12,7 mm), a na nju je nanesen sloj metalnog oksida koji se može trajno namagnetisati. Površina ovog sloja deli se u uzdužnom smeru trake na kanale, a popreko na kolone; sićušni tragovi magnetizma dvaju polova u presecima kanala i kolona čine kombinaciju *bitova* ili kôd magnetne trake.



Slika 1.5 9-kanalni kôd magnetne trake

Na slici 1.5 šematski je prikazan jedan odsečak magnetne trake sa vidljivo upisanim kodovima znakova. Prikazani način kodiranja zove se *devetokanalni kôd,* koji se isključivo upotrebljava kod savremenih računara. Raspored kanala je znatno drukčiji od onoga kod papirne trake. I ovde postoji jedan kanal u koji se upisuju bitovi za kontrolu parnosti (vertikalna ili poprečna kontrola), tako da je broj bitova u svakoj koloni neparan.



Slika 1.6 Slogovi na magnetnoj traci

Na slici 1.6 dat je šematski prikaz jednog komada magnetne trake sa upisanim slogovima podataka. U svakom slogu ima više znakova upisanih 9-kanalnim kodom. Slogovi su razdvojeni razmacima (pauzama) standardne dužine 3/4 ili 1/2 inča (19 odnosno 12,7 mm), na kojima nema podataka. Radi obezbeđenja od nečitkosti

pojedinih bitova, svaki slog sadrži u poslednjoj koloni jedan ceo znak za horizontalnu ili uzdužnu kontrolu parnosti; bitovi tog znaka dopunjavaju zbir svih bitova u pojedinom kanalu tako da bude neparan.

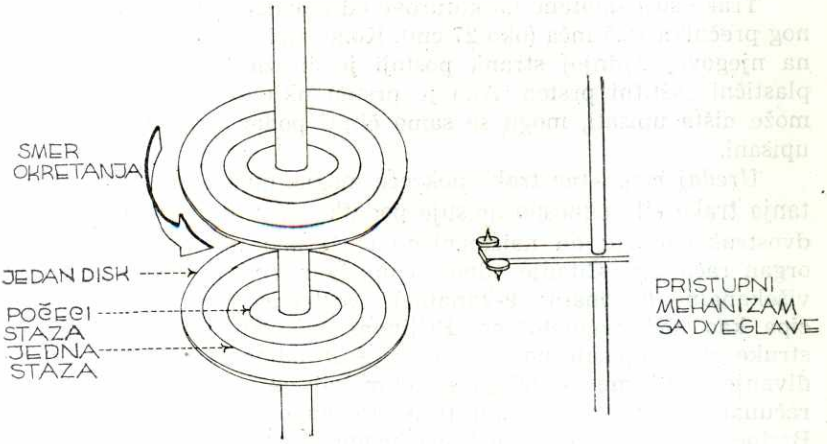
Trake su namotane na koturove od plastičnog materijala, spoljnog prečnika 10,5 inča (oko 27 cm). Koncentrično oko otvora kotura, na njegovoj zadnjoj strani, postoji jedan žleb u koji se stavlja plastični zaštitni prsten. Ako je prsten uklonjen, na traku se ne može ništa upisati, mogu se samo čitati podaci koji su prethodno upisani.

Uređaj magnetne trake pokreće magnetnu traku i u toku kretanja trake čita odnosno upisuje podatke na traku. Taj uređaj ima dvostruku ulogu: on naizmenično služi kao ulazni i kao izlazni organ računara. Čitanje odnosno upisivanje vrši se pomoću jedne višekanalne (normalno 9-kanalne) dvostruke glave, sličnog principa kao kod magnetofona. Prilikom upisivanja, jedan deo dvostruke glave upisuje podatke, a drugi ih odmah zatim čita; upoređivanjem upisanog sadržaja sa onim koji je primljen iz memorije računara, automatski se kontroliše ispravnost operacije upisivanja. Brzine rada savremenih jedinica magnetne trake kreću se između 60 i 120 hiljada kolona u sekundi.

Magnetni disk je nosilac informacije na istom fizičkom principu kao magnetna traka. Na koncentričnim kružnim stazama s obe strane diska upisuju se podaci devetokanalnim kodom (kao kod trake, samo bez devetog bita za kontrolu parnosti), s tim što su bitovi svakog znaka postavljeni duž staze jedan za drugim (serijski), a ne kao kod trake u jednoj koloni (paralelno). Disk se za vreme rada računara obrće u jedinici magnetnog diska konstantnom brzinom. Dok je pristup individualnom podatku na traci uslovljen odmotavanjem trake (sekvencijalni pristup), dotle je svaki podatak na disku pristupačan za veoma kratko vreme, zbog čega se disk naziva nosiocem ili medijumom sa direktnim pristupom.

Uređaj magnetnog diska održava disk u ravnomernom kružnom kretanju i, preko pokretne glave za čitanje-upisivanje, čita odnosno upisuje podatke na staze diska. I ovaj uređaj služi prema tome naizmenično za ulaz i za izlaz. Obično je više diskova fizički spojeno u jednu celinu — paket, pa zato uređaj ima jedinstven mehanizam sa više glava za čitanje-upisivanje (slika 1.7). Mehanizam sa glavama kreće se u odnosu na diskove radijalno i može zauzeti onoliko položaja koliko ima koncentričnih staza na disku. Tipična brzina kojom se podaci čitaju odnosno upisuju jeste približno 150 hiljada znakova u sekundi. Međutim, ovde treba

računati i sa vremenom koje je potrebno za dovođenje glave na odgovarajuću stazu, tj. za kretanje mehanizma koji nosi glave za čitanje-upisivanje.



Slika 1.7 Magnetni disk-princip

Dokumenti na koje se informacije ubeležavaju specijalnim magnetnim znacima ili znacima za optičko mašinsko čitanje, kao i ulazni uređaji za čitanje takvih dokumenata, nisu predviđeni za upotrebu u sistemu FORTRAN, pa zato o njima nećemo govoriti.

1.2.3 *IZLAZNI ORGANI*

Izlazni organ ili uređaj je mašina koja prima iz memorije računara informacije, transformisane u oblik električnih impulsa, i upisuje ih na neki od spoljnih nosilaca. Dva tipa nosilaca, njihove uređaje i kodove videli smo već opisujući funkciju ulaza u računar, jer isti uređaji po potrebi obavljaju i funkciju izlaza (magnetna traka i disk).

Pisaća mašina je još jedan takav organ računara koji naizmenično služi za ulaz i za izlaz. Ovaj uređaj služi isključivo operatoru za nadzor nad radom računara. Informacije izlaze štampanim znacima na list hartije, a unose se preko tastature. Standardna brzina izlaza je 10 do 15 znakova u sekundi.

Štampač je izlazni uređaj na kojem se štampaju rezultati, dobijeni radom programa u računaru. Podaci se iz centralne memorije primaju u pomoćnu memoriju štampača, odakle se odjednom štampa ceo jedan red, koji obično ima dužinu od 120 do 160 znakova. U ovoj knjizi ima više primera štampanih redova. Hartija se u štampaču kreće automatski, na osnovu komandi iz programa, i zbog toga listovi nisu međusobno odvojeni nego čine jedan »beskrajni« obrazac. Posle štampanja, pojedini listovi se lako odvajaju duž perforiranih linija. Često se izlazni podaci ne štampaju direktno nego najpre izlaze na magnetnu traku ili disk. Kasnije se posebnim standardnim programom vrši štampanje sa tih nosilaca. Tipična brzina rada štampača je oko 1000 redova u minuti.

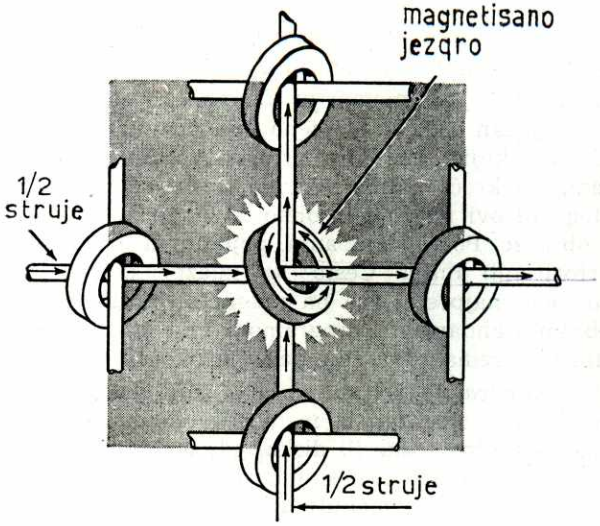
Bušač kartica je izlazni uređaj koji kao i štampač može direktno da buši podatke iz memorije računara ili ih buši naknadno, sa magnetne trake ili diska, gde su prethodno bili direktno upisani. Kodovi su najčešće standardni, dakle oni isti koje vidimo na slici 1.2 ili drugi, pomenuti u opisu kartice kao nosioca informacije.

1.2.4 *MEMORIJA*

Memorija je onaj organ računara koji može da čuva ili pamti informaciju, koju je računar primio preko jednog od ulaznih uređaja ili koju je stvorio u toku obrade po programu. Takođe, memorija je taj organ iz kojeg se rezultati šalju u izlazne uređaje, da bi bili magnetski upisani, direktno odštampani ili bušeni u kartice.

Memorija računara izrađuje se od fizičkih elemenata koji imaju dva stabilna stanja. Pri današnjem stanju razvoja tehnike, ti elementi su po pravilu magnetskog tipa i dele se na dve kategorije: takve kod kojih je kontinualan magnetski sloj nanesen na neku nemetalnu osnovu (magnetni film) i druge, koji se sastoje od diskretnih feromagnetnih jezgara prstenastog oblika (feritna jezgra). Kod prvih se binarno kodirana informacija upisuje i čuva u vidu kombinacija namagnetisanih tačaka u kontinualnom sloju, dok se kod drugih binarni kodovi informacije ubeležavaju suprotnim namagnetisanjem feritnih jezgara po grupama. Upisivanje i čitanje bitova informacije vrši se kod obe kategorije pomoću magnetnog polja, nastalog oko provodnika kroz koji protiče impuls električne struje (slika 1.8).

Sve vrste memorija, pomenute ili drugih vrsta, dele se na *pozicije*. Jedna pozicija može da primi jedan znak informacije, na



Slika 1.8 Upisivanje jednog bita kod feritne memorije

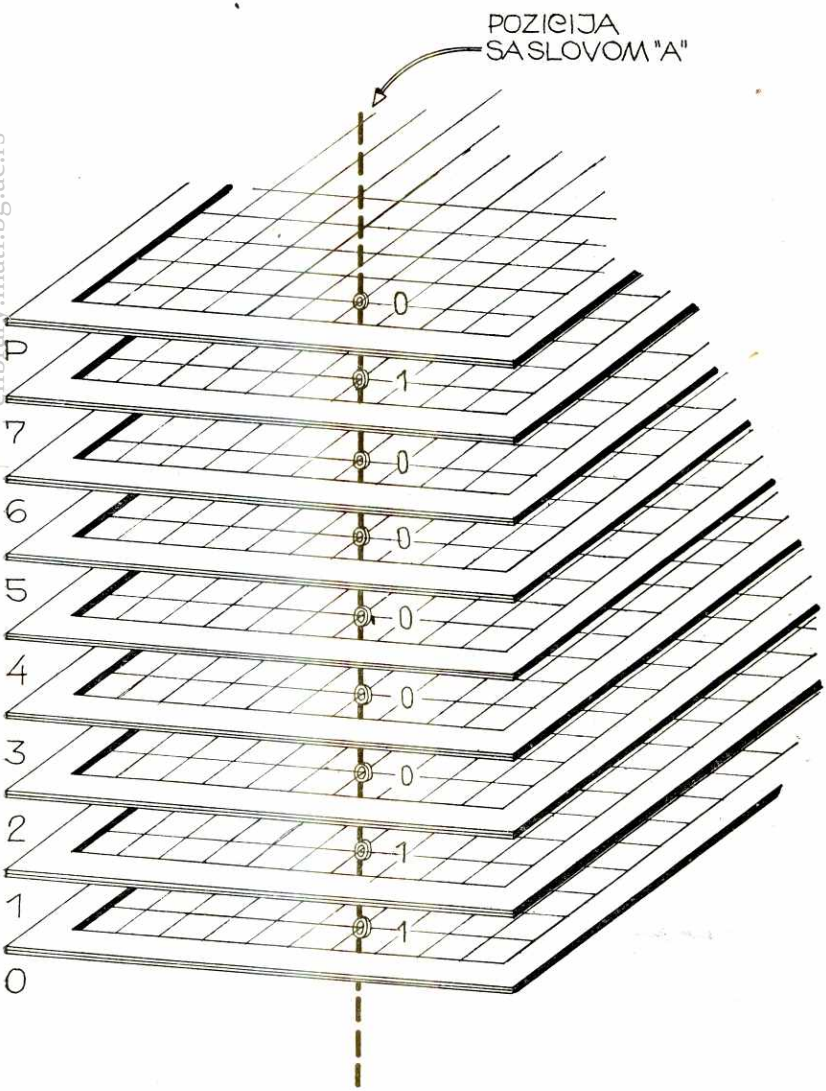
primer jedan od znakova na slici 1.2. Šematski prikaz feritne memorije, kod koje se svaka pozicija sastoji od 9 feritnih jezgara, vidimo na slici 1.9. Kôd za unošenje informacije u tako organizovanu memoriju u principu je isti kao devetokanalni kôd magnetne trake sa slike 1.5.

Više pozicija memorije čine jednu *mašinsku reč*. Mašinska reč fiksne dužine sadrži uvek isti broj pozicija, dok se za računare kod kojih broj pozicija u reči nije stalan kaže da im je mašinska reč promenljive dužine. Svaka pozicija ima jednu jedinstvenu *adresu;* to je redni broj pozicije u okviru memorije. Pomoću tog broja može se pročitati sadržaj adresirane pozicije, odnosno u nju upisati kôd nekog znaka.

Osim za prijem podataka, memorija računara služi i za smeštaj programa, koji se u memoriju unosi na isti način kao i podaci.

1.2.5 *ARITMETIČKO-LOGIČKI ORGAN*

Aritmetičko-logički organ se sastoji od većeg broja elementarnih elektronskih kola (krugova) za aritmetičke i logičke operacije. Kod nekih računara aritmetičke operacije se izvode direktno sa operandima u memoriji, kod drugih se operandi prenose u *regi-*



Slika 1.9 Feritna memorija (jedna pozicija sadrži 9-kanalni kôd za slovo A)

stre (akumulatore) aritmetičko-logičkog organa gde se obavlja operacija, pa se rezultat vraća u memoriju. Za aritmetičke operacije potrebna su u principu tri registra; na primer za množenje, u jednom registru se nalazi množenik, u drugom množilac, dok se u trećem formira rezultat. Mada je to u praksi konstruktivno drukčije izvedeno, ovakva predstava o sastavu aritmetičko-logičke jedinice je u principu tačna i za nas dovoljna. Registar prema tome ima kapacitet za prijem jednog operanda.

Sama aritmetička operacija se izvodi kod nekih računara znak po znak ili serijski, kao u stolnoj računskoj mašini, dok se kod drugih radi odjednom sa svim ciframa u registru, tj. paralelno. Prilikom izvođenja operacije, operandi se u registrima pomeraju ulevo ili udesno, kao i kod stolne računske mašine.

Osim četiri aritmetičke operacije, aritmetičko-logički organ može da izvodi i jednostavne logičke operacije. To su operacije kojima se ispituje neki broj da li je jednak ili različit od nule, da li je pozitivan ili negativan. Sve takve operacije u računaru zasnivaju se na dualnoj ili binarnoj prirodi njegovih elemenata (0 ili 1).

Aritmetičko-logički organ može da vrši samo najjednostavnije aritmetičke i logičke operacije. Još pri programiranju zadatka moraju se sve komplikovanije radnje razložiti metodama numeričke analize i logike na proste operacije. Na primer, za određeni integral funkcije $\sin x$ u datim granicama, mora se upotrebiti čitav niz prostih aritmetičkih i logičkih operacija.

1.2.6 *UPRAVLJAČKI ORGAN*

Upravljački organ aktivira i sinhronizuje rad svih ostalih organa i uređaja računara. Taj organ sadrži potrebne elemente za upravljanje svim funkcijama računara.

Iz memorije računara prenosi se u upravljački organ jedna po jedna instrukcija, gde se dekodira. Fizičkom izvršavanju instrukcije prethode komande, koje upravljački organ upućuje potrebnim elementima računara. Na osnovu tih komandi vrši se prenos operanada iz memorije u registre, aktivira se funkcija aritmetičko-logičkog organa, pokreću se i zaustavljaju ulazni i izlazni uređaji. Često je funkcija upravljačkog organa realizovana hijerarhijski, tj. jedan centralni upravljački organ povezan je sa više perifernih.

Od osnovnog značaja su dva registra upravljačkog organa: *registar instrukcije* i *registar adrese*. U registar instrukcije prenosi se iz memorije jedna instrukcija i tu interpretira razlaganjem na

kôd operacije i adresni deo. Kôd operacije kazuje šta treba da se radi (množenje, prenos itd.), dok adresni deo sadrži adrese operanada koji učestvuju u operaciji. Registar adrese sadrži adresu na kojoj se u memoriji nalazi sledeća instrukcija, koja će se preneti u registar instrukcije kada tekuća instrukcija bude izvršena. Sadržaj registra adrese formira se na dva načina: ili se u njega postavlja određena adresa pomoću neke instrukcije ili se, u odsustvu ovakve instrukcije, prethodni sadržaj automatski povećava tako da sadrži adresu sledeće instrukcije koja treba da se izvrši.

1.3 *SISTEM PROGRAMIRANJA*

Svi funkcionalni delovi računara, koje smo do sada opisali polazeći od njegove opšte organizacije predstavljene slikom 1.1, nazivaju se opštim imenom *tehnika računara* (hardware*) ili fizički vidljivi delovi. Savremeni univerzalni cifarski računar ne može se, međutim, zamisliti bez značajne grupe elemenata sasvim druge prirode, koji se zajednički nazivaju *sistem programiranja* (software*) ili nematerijalni delovi računara.

Jezici programiranja su skupovi instrukcija ili naredbi, zajedno sa pravilima njihove upotrebe. Videli smo da je jedna instrukcija kodirani nalog mašini da izvrši jednu elementarnu operaciju.

Najelementarniji jezik kojim se može napisati program jeste sopstveni interni jezik računara, koji zovemo *mašinskim* jezikom. *Instrukcija* na mašinskom jeziku sastoji se u principu od koda operacije i jednog ili više operanada. Sabiranje dva broja, koji se već nalaze u memoriji, programira se na primer pomoću sledećih mašinskih instrukcija za računar čije instrukcije imaju samo jedan operand**:

kôd	*operand*
10	001
11	005
50	000

što znači:

kôd 10 — izbrisati prethodni sadržaj akumulatora i preneti u njega vrednost koja se nalazi u memoriji na adresi 001,

---

* *hardware* i *software* su nazivi u literaturi na engleskom jeziku.

** Ovde nije uzet za primer jezik nekog određenog računara; upotrebljene instrukcije samo daju principijelnu predstavu o mašinskom jeziku uopšte.

kôd 11 — sabrati sa sadržajem akumulatora vrednost koja se nalazi u memoriji na adresi 005,
kôd 50 — sadržaj akumulatora preneti u memoriju, na adresu 000.

Prvi složeniji ali i za programiranje lakši jezik od mašinskog jeste tzv. *simbolički* mašinski jezik, koji se u engleskoj literaturi zove asembler-jezik. Na tom jeziku gornje operacije se na primer pišu:

kôd	*operand*
CLA	A
ADD	B
STO	C

Kodovi i adrese operanada predstavljeni su alfabetskim simbolima i ne mogu se direktno izvršiti u računaru; takav program mora se najpre prevesti na mašinski jezik. Prevodilac za simbolički jezik zove se u engleskoj literaturi kod većine računara *asembler*. U simboličkom jeziku, osim prostih instrukcija, postoje i tzv. makro-instrukcije koje se pri prevođenju zamenjuju sa više mašinskih instrukcija. Za asembler-jezik kaže se da je *mašinski orijentisan*.

Viši jezici, kao što su ALGOL, COBOL, FORTRAN, PL/1, nemaju više nikakve sličnosti sa mašinskim jezikom. Njihove instrukcije zovemo *naredbama* i u njima se primenjuje simbolika slična matematičkoj ili ekonomskoj. Na primer, u COBOL-u se umesto gornje tri instrukcije piše naredba:

ADD A TO B GIVING C

ili u FORTRANu, naredba:

C = A + B

Prevodioci za takve jezike, za koje se još kaže da su *problemski orijentisani* jezici, zovu se kompajleri ili *kompilatori* jer osim prostije funkcije sklapanja (assemble) obavljaju i složeniju funkciju logičkog sakupljanja i sređivanja (compile) programa na mašinskom jeziku.

Sistem *upravljačkih programa* je najvažniji element sistema programiranja (softvera) savremenog računara. To je sklop gotovih programa na mašinskom jeziku, koji se nalazi na nekom spoljnom nosiocu (disk ili traka) i čiji delovi prema potrebi automatski ulaze u memoriju računara, gde obavljaju funkcije upravljanja celokupnim njegovim radom.

Prevodioci su takođe programi na mašinskom jeziku, koji pod nadzorom upravljačkog sistema vrše prevođenje programa sa simboličkog ili problemski orijentisanog jezika na mašinski jezik. Svaki računar ima obično nekoliko prevodilaca, od kojih je jedan FORTRAN-prevodilac. Operacije čitanja podataka sa nekog spoljnog medijuma prevodilac prevodi tako da ih izvršava upravljački program, koji samo dobija osnovne informacije iz naredbe za ulaz, napisane u našem programu. Isto tako se izvode i operacije upisivanja podataka i rezultata na spoljne medijume.

Biblioteke programa su neophodan deo softvera i njima se služi sistem upravljačkih programa prilikom prevođenja, a takođe i prilikom izvršavanja gotovih mašinskih programa. Elementi biblioteka su programi-prevodioci, pomoćni standardni programi i izvorni programi, u neprevedenom i u prevedenom obliku. U bibliotekama se takođe nalaze, kao gotovi elementi, mašinski potprogrami za izračunavanje svih važnijih matematičkih funkcija (sin x, e^x, ln x itd.).

1.4 *PROGRAMIRANJE*

Da bi se jedan problem rešio pomoću računara, mora se obaviti niz pripremnih faza koje često zahtevaju od čoveka znatno više rada i vremena nego što će zahtevati sam računar za svoj deo posla. Ovde ćemo redom opisati ceo taj postupak, počev od sagledavanja potrebe da se problem rešava uz pomoć računara, pa do gotovih rezultata.

1°) *Definisanje problema* je prva faza u prilazu problemu koji se želi rešavati. Tu se donosi odluka o ciljevima koje treba postići, a preciziraju se i uslovi pod kojima će se zadatak rešavati u računaru. Sama studija teoretske strane problema može ponekad zahtevati i rad od više meseci. U ovoj fazi se od čoveka traži potpuno stručno poznavanje oblasti iz koje se zadatak postavlja i tu računar ne može pružiti nikakvu pomoć.

2°) *Matematičko formulisanje* sledi fazu definicije zadatka. Zadatak se obično može matematički formulisati na nekoliko načina i sada treba izabrati jedan od njih. Pored vladanja stručnom stranom problema, ovde je neophodno znanje odgovarajućih oblasti matematike.

3°) *Numerička analiza* je disciplina matematike koja omogućuje da se matematički formulisan zadatak pripremi za računanje. Ova faza je neophodna jer računar može da obavlja samo jednostavne

aritmetičke i logičke operacije. Istovremeno sa preciziranjem numeričkog postupka, ovde treba sagledati i moguće greške koje unose ulazni podaci i računske operacije i njihov uticaj u izabranom postupku.

Cilj ove knjige je isključivo da pruži materijal za izučavanje pravila FORTRAN-programiranja, radi čega gornje tri pripremne faze neće biti detaljnije obrađivane. Pretpostavlja se da je čitalac u mogućnosti da uspešno obavi taj deo posla i da je potrebno tako pripremljen problem izraziti u obliku FORTRAN-programa.

Izrada programa za računar jeste izražavanje gotovog numeričkog postupka u vidu niza operacija, koje će izvršiti računar. Sem u vrlo prostim slučajevima, ova faza rada se deli na grafičko predstavljanje postupka i zatim na pisanje naredbi na jednom od mogućih izvornih jezika. Jedan od takvih jezika je i FORTRAN IV, koji ćemo ovde izučavati. Grafičko predstavljanje postupka opisano je u Glavi 5, a pravila za pisanje FORTRAN-programa izlažu se kroz celu knjigu.

Kontrolisanje programa je faza koja se ponavlja sve dok program ne da zadovoljavajuće rezultate. Pre prevođenja u računaru, program se proverava tako što se jedan jednostavniji zadatak reši ručno, sledeći tok programiranih operacija. U toku prevođenja u računaru, prevodilac otkriva formalne greške i daje preko računara pismene poruke. Posle izvršenog prevođenja, program se proverava u radu sa probnim podacima; ta provera naziva se testiranje programa.

Rad programa je ona faza u kojoj računar automatski rešava programirani problem. Kao što ćemo videti, računaru se obično daje više grupa početnih veličina (ulaznih podataka) za računanje, koje on obrađuje automatski, jednu za drugom. Osim toga, zahvaljujući upravljačkim programima softvera, kada završi rad po jednom programu računar odmah nastavlja sa sledećim. Kod većih računara upravljački programi organizuju rad računara na *multiprogramski* način: pri takvom radu maksimalno se koristi kapacitet računara jer dok se za jedan program obavljaju relativno spore operacije ulaza i izlaza, istovremeno se vrše unutrašnje, računske i logičke operacije drugog programa.

Interpretacija rezultata je faza koju, posle završenog rada računara, obavlja čovek. Često rezultati koje daje program nisu konačan odgovor na problem. Program je mogao biti napravljen tako da izračuna jednu grupu mogućih rešenja, a da se na osnovu subjektivne interpretacije tih rešenja zadaju novi početni podaci za ponovni rad programa.

Iz svega što je rečeno o programiranju uopšte, samo faza *izrade programa* biće predmet našeg izučavanja. Praveći ovaj kratak pregled mogli smo sagledati da računar sam za sebe ne rešava probleme, da on ne oslobađa čoveka od odgovornosti za logično postavljanje zadatka, njegovo tačno matematičko formulisanje, numeričku analizu, izradu i kontrolu programa kao i interpretiranje rezultata.

1.5 *JEZIK FORTRAN*

Da bi elektronski računar mogao da rešava neki računski problem iz nauke ili tehnike, postupak kojim se taj problem rešava mora se, kao što smo videli, računaru saopštiti na jeziku koji on »razume«. Jezik računara sastoji se od elementarnih *instrukcija*. Svakoj instrukciji odgovara jedna elementarna operacija kao što je na pr. sabiranje, oduzimanje, pomeranje broja ulevo ili udesno, čitanje podatka sa magnetne trake, bušenje podatka u karticu itd. To znači da se postupak za rešavanje problema mora na neki način *prevesti* na jezik prostih instrukcija, da bi računar mogao da ga izvršava. Posao oko prevođenja, koji se naziva *programiranje* ili *kodiranje*, može u celini da uradi čovek ali mu u tome može pomagati i sam računar svojim programom za prevođenje. Takav program za prevođenje problemski orijentisanih jezika poznat je pod nazivom kompilator ili kompajler (compiler), a mi ćemo ga dalje zvati program-prevodilac ili prosto *prevodilac*.

Program-prevodilac je jedan veliki i složen skup instrukcija na jeziku računara. Za vreme prevođenja on se nalazi u memoriji računara, odakle upravlja radom svih njegovih uređaja. Pod kontrolom prevodioca, računar analizira tekst kojim je opisan postupak za rešenje zadatka i proizvodi nizove potrebnih elementarnih instrukcija. Te instrukcije čine progam, čijim se izvršavanjem može posle rešavati bezbroj zadataka iste vrste, svaki put sa novim početnim podacima.

Prevodilac, dakle, predstavlja vezu između jezika kojim je problem opisan i jezika računara. Jedan računar ima uvek samo jedan interni, sopstveni jezik ali obično može da rešava zadatke koji su opisani ili programirani na nekoliko više ili manje univerzalnih, standardnih *jezika programiranja*. Zato što se prevođenje vrši na posebni interni jezik svakog računara, prevodilaca ima mnogo, najmanje onoliko koliko i raznih tipova računara.

Prvi problemski orijentisan jezik programiranja, koji je ušao u široku upotrebu, razvili su sredinom prošle decenije stručnjaci firme IBM* za računar IBM 704, u zajednici sa prvim korisnicima tog tipa računara. Taj jezik, namenjen rešavanju matematički formulisanih problema iz oblasti nauke i tehnike, čini zajedno sa svojim prevodiocem jednu celinu ili sistem. Sistemu je dat naziv FORTRAN, što je skraćenica od engleskog naziva FORmula TRANslation System, a znači doslovno »sistem za prevođenje formulâ«. U toku daljih deset godina sistem FORTRAN je usavršen i 1965. je za njega propisan standard u SAD, obavezan za sve proizvođače računara. Standardizovane su dve verzije: potpuni FORTRAN IV i osnovni FORTRAN IV, pri čemu ovaj poslednji predstavlja jedan izvod ili podskup potpunog sistema, namenjen za manje računare.

U današnje vreme, cifarski elektronski računari se veoma široko koriste za rešavanje problema u nauci, tehnici i privrednom životu. Oni se angažuju pri projektovanju tehničkih sistema, u analizi rezultata raznih ispitivanja i eksperimenata, u upravljanju procesima rada, u vođenju raznih evidencija i u rešavanju mnogih zadataka iz oblasti privrednog poslovanja. Razlog sve veće primene je velika brzina rada računara, njegova sposobnost da čuva (pamti) velike količine informacija i da izvršava duge i komplikovane nizove operacija bez pomoći čoveka.

Prema oblastima delatnosti iz kojih potiču, zadatke na kojima se angažuju računari možemo svrstati u dve velike klase: naučno-tehničke i ekonomsko-komercijalne. U ove druge spadaju na pr. obračun dohotka, operativno poslovanje privrednog preduzeća, izrada statističkih pregleda prodaje, planiranje proizvodnje i sve vrste knjigovodstva. Primenu računara u rešavanju naučno-tehničkih problema, radi kojih je i stvoren sistem FORTRAN, ilustrovaćemo sa sledećih nekoliko primera.

Pri projektovanju tehničkih konstrukcija i tehnoloških procesa računa se sa velikim brojem promenljivih veličina. Kompletan proračun na bazi samo jedne odabrane kombinacije zahteva veliki računski rad, a da se dođe do optimalne konstrukcije ili procesa treba uporediti rezultate više takvih proračuna. Računar je od neocenjive koristi u tom radu jer se sa njim, uz neznatan dodatni rad čoveka, proračunski postupak može neprestano ponavljati sa novim kombinacijama parametara, sve dok se ne postigne optimalno rešenje.

* International Business Machines

Laboratorijska i terenska ispitivanja probnih serija proizvoda obično daju veoma mnogo podataka, koje treba srediti i analizirati. Za izračunavanje popravnih faktora, za eliminaciju nelogičnih ekstremnih izmerenih vrednosti i za dobijanje upotrebljivih prosečnih veličina često je neophodna pomoć elektronskog računara.

Istraživanja u oblasti sociologije, medicine, biologije i ekonomike vrše se na osnovi statističkog posmatranja velikog broja jedinica. Analitički rad, kojim se vrše uopštavanja i izvlače zaključci, zahteva primenu matematičko-statističkih postupaka sa ogromnim brojem aritmetičkih operacija. Za potrebe naučno-istraživačkog rada u ovim oblastima postoje čitave zbirke gotovih standardnih programa, pisanih najvećim delom na jeziku FORTRAN.

FORTRAN koji će ovde biti izložen obuhvata celokupnu materiju potpunog FORTRANa IV, prema važećem američkom standardu. Većina računara, koji su instalirani u našoj zemlji, raspolaže FORTRAN-prevodiocima za osnovni FORTRAN IV, ponekad sa izvesnim proširenjima u smeru ka potpunom FORTRANu IV. Samo mali broj može u celini da koristi širu standardnu verziju, pa čak i izvesne specifične dodatne osobine jezika, koje nisu predviđene standardom. Radi toga su u tekstu uvek posebno istaknuti oni elementi koji nisu sadržani u osnovnom FORTRANu IV, tj. takvi za čiju primenu je potreban prevodilac za potpunu ili još širu verziju jezika.

Glavne karakteristike FORTRAN-prevodilaca za računare koji su kod nas u upotrebi i još za neke druge, date su u tabeli 12.2 na kraju knjige. Čitaocu koji želi da za sopstveni program koristi neki određeni računar, preporučuje se da u konkretnom računskom centru prethodno dobije bliže podatke o sastavu računara, karakteristikama FORTRAN-prevodioca i pravilima korišćenja instalacije. O svemu što je potrebno za praktično korišćenje računara biće više reči kasnije.

1.6 *FORTRAN-program*

Postupak za rešavanje zadatka izražava se na FORTRANu *naredbama*. Drugim rečima, jedan FORTRAN-program sastoji se od niza FORTRAN-naredbi. Njih ima nekoliko vrsta. Suštinu postupka čine aritmetičke i logičke naredbe. Druga vrsta naredbi su one kojima se komanduje izvršavanje ulaznih i izlaznih operacija, kao što je na primer čitanje podataka iz jedne kartice i njihovo

unošenje u memoriju računara, štampanje jednog reda rezultata ili upisivanje međurezultata na magnetnu traku ili disk. Treća vrsta naredbi su takve pomoću kojih se upravlja redosledom izvršavanja svih ostalih naredbi u jednom programu. U četvrtu vrstu ubrajaju se naredbe koje daju izvesne informacije o programu, a pri tome ne prouzrokuju nikakvu akciju. Petu vrstu čine naredbe sa kojima se definišu i povezuju sa programom zasebne programske jedinice, tzv. potprogrami.

Uzete zajedno, sve FORTRAN-naredbe koje opisuju put i način kako će se rešavati jedan zadatak, sačinjavaju *izvorni program* na FORTRANu. Kada se izvorni program napiše i izbuši u kartice, računar ga uz pomoć FORTRAN-prevodioca prevede ili, bolje rečeno, razradi u *mašinski program,* tj. u niz elementarnih instrukcija koje računar jedino »razume«. Da bi se zadatak rešio, u računaru se izvršava taj mašinski program.

U sledećoj glavi ćemo preći na izučavanje elemenata od kojih se sastoje FORTRAN-naredbe, a to su konstante, promenljive, operatori, izrazi i funkcije. Posle savlađivanja tih elementarnih jedinica jezika, postepeno ćemo naučiti kako se od njih grade naredbe, a zatim i kako te naredbe treba kombinovati kada se rešavaju problemi iz raznih oblasti nauke i tehnike.

Glava 2

ELEMENTI JEZIKA FORTRAN

2.1 *KONSTANTE*

Počnimo razmatranjem dveju glavnih vrsta brojeva koje se mogu upotrebljavati u FORTRAN-programima: to su *celi* i *realni* brojevi. Treba odmah reći da te dve vrste imaju u FORTRANu svoje nazive ili atribute, koji se obavezno navode u izvesnim naredbama svakog programa. Tako je reč INTEGER atribut za cele brojeve, dok je reč REAL atribut za realne brojeve.

Ceo broj (INTEGER) ima u FORTRANu istu definiciju kao i u matematici: to može biti nula ili bilo koji pozitivan ili negativan ceo broj, s tim da mu apsolutna vrednost ne prelazi određenu veličinu, koja je različita za razne računare. U tabeli 12.2 navedeni su za izvesne tipove računara maksimalni broj cifara i maksimalna apsolutna vrednost celih brojeva. Pošto celi brojevi mogu imati isključivo samo cele vrednosti, bez decimala, to se oni u FORTRAN-programima upotrebljavaju samo u specijalnim slučajevima, što ćemo videti kasnije.

U FORTRANu se pretežno operiše realnim brojevima (REAL). Takvi brojevi u opštem slučaju imaju neku celobrojnu vrednost i određen broj decimala, mada vrednost decimalnog dela može biti i nula. U memoriji računara, za razliku od celih brojeva (INTEGER), realni brojevi imaju oblik pokretnog zareza, uobičajen u naučnom računanju. U tom obliku, brojevi se predstavljaju kao proizvod jedne decimalne *mantise*, čija je apsolutna vrednost jednaka ili veća od nule a manja od jedinice, i jedne *karakteristike* oblika 10^n, gde je n jedan pozitivan ili negativan celobrojni izložilac. Maksimalna vrednost realnog broja je opet zavisna od tipa i veličine računara. U tabeli 12.2 možemo za neke tipove računara videti koliki je dozvoljeni broj cifara mantise i kolika je maksimalna apsolutna vrednost pripadajućeg stepena.

Vidimo da su u FORTRANu definicije realnog i celog broja iste kao i u matematici: klasa celih brojeva sadržana je u klasi realnih. Specifičnost FORTRANa je, međutim, u internom obliku predstavljanja tih dveju vrsta brojeva u memoriji računara, odakle proizlazi i razlika u aritmetičkim pravilima za interni rad sa tim dvema vrstama brojeva. Pri aritmetičkim operacijama sa realnim brojevima u FORTRANu, mi ne moramo da vodimo računa o položaju decimalnog zareza. Pravilno »potpisivanje« brojeva prilikom sabiranja i oduzimanja, a isto tako i određivanje mesta decimalnog zareza kod proizvoda i količnika, jesu stvar o kojoj se automatski brine sam računar.

Realni broj u FORTRANu uvek je racionalan u matematičkom smislu, s tim što je broj cifara ograničen. Iracionalni brojevi se takođe mogu izraziti samo sa ograničenom tačnošću. Na primer, prost razlomak 1/3 može se realnim brojem u FORTRANu izraziti samo približno.

Broj decimala kod realnih brojeva u FORTRANu najčešće je 7 ili 8, mada ima računara kod kojih je i veći. Zato sistem FORTRAN raspolaže još jednom podvrstom realnih brojeva, tzv. realnim *brojevima dvostruke tačnosti.* Atribut za takve brojeve je DOUBLE PRECISION. Dvostruku tačnost, međutim, ne treba shvatiti kao dva puta veći broj decimala nego kod običnog realnog broja. Atribut DOUBLE PRECISION znači dvostruko veću dužinu ćelije* u memoriji u koju se smešta *binarna* vrednost tog realnog broja. Imajući to u vidu, mi ćemo se nadalje služiti terminom dvostruka tačnost. U tabeli 12.2 to se lepo može videti iz podataka o broju cifara mantise kod realnih brojeva obične i dvostruke tačnosti. Tu takođe vidimo da je maksimalna apsolutna vrednost dvostruko tačnih realnih brojeva po pravilu ista kao i kod običnih.

U potpunom FORTRANu IV postoje, kao posebna vrsta, i kompleksni brojevi. Atribut za te brojeve je COMPLEX. FORTRAN-kompleksni broj je uređeni par realnih brojeva; svaki broj u paru odgovara definiciji običnog realnog broja, a piše se u zagradama najpre realni i zatim imaginarni deo, ovaj poslednji bez broja *i;* delovi su međusobno odvojeni zarezom. Prilikom izvođenja aritmetičkih operacija, sam računar vodi računa o poznatim pravilima računanja sa kompleksnim brojevima. Kod većih računara mogu se čak i oba dela kompleksnog broja izraziti kao realni brojevi dvostruke tačnosti. Za računare čiji FORTRAN-prevodioci ne mogu da rade sa kompleksnim brojevima, sve operacije sa takvim broje-

* Ćelija ili polje u memoriji sastoji se od nekoliko susednih pozicija.

vima moraju se programirati. Kao podsetnik, niže su data aritmetička pravila za kompleksne brojeve:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$
$$(a + bi) - (c + di) = (a - c) + (b - d)i$$
$$(a + bi)\ (c + di) = (ac - bd) + (ad + bc)i$$
$$(a + bi) / (c + di) = \frac{(ac + bd)}{c^2 + d^2} + \frac{(bc - ad)}{c^2 + d^2}\, i$$

Veličina koja se u izvornom FORTRAN-programu pojavljuje eksplicitno, kao poseban broj, naziva se *konstantom*. Veličina predstavljena simbolom, a ne posebnim brojem, zove se promenljiva veličina ili prosto *promenljiva*. Na primer, u sledećim aritmetičkim naredbama, o kojima ćemo govoriti malo kasnije, imamo i konstante i promenljive:

```
I = 2
X = A + 12.7
P = (2.6, 1.3)
```

(Mada tačku u brojevima 12.7, 2.6 i 1.3 normalno zovemo decimalni zarez, po pravilima FORTRANa se za tu svrhu uvek upotrebljava tačka, pa ćemo se zato ubuduće i mi služiti nazivom *decimalna tačka*. Takođe, nije dozvoljeno ni odvajanje dekadnih klasa kod većih brojeva; na primer, nije pravilno ni 123.456,7 ni 123 456.7 nego samo 123456.7).

U gornjim primerima su 2, 12.7, 2.6 i 1.3 konstante, a I, X, A i P su promenljive.

Konstanta u obliku celog broja razlikuje se od konstante u obliku realnog broja samo po prisustvu ili odsustvu decimalne tačke. Tako je 6 jedna celobrojna konstanta, dok je 6.0 (ili 6. ili 6.000) realna konstanta. Zbog različitog internog predstavljanja ovih dveju vrsta brojeva (u memoriji računara), nije svejedno kako će se gornja konstanta pisati u naredbama FORTRAN-programa.

Ako je konstanta pozitivan broj, znak plus ispred broja može se po želji pisati ili izostaviti. Ako je pak konstanta negativna, znak minus se mora pisati.

Ovo su primeri pravilno napisanih celobrojnih konstanti:

```
   0
   6
 +400
 -1357
  99999
```

Slede primeri nepravilno napisanih celobrojnih konstanti:

12.75	(decimalna tačka nije dozvoljena)
672937458563	(apsolutna vrednost veća od maksimalno dozvoljene — videti podatak u tabeli 12.2)

Pravilno napisane realne konstante (obična tačnost):

0.0	
6.0	
6.	
—20000.	
—.0002784	(nula ispred decimalne tačke nije obavezna)
+15.016	

Vidimo da se kod realne konstante decimalna tačka može nalaziti na početku, na kraju broja ili između njegovih cifara. Realna konstanta obične tačnosti može se pisati sa proizvoljnim brojem decimalnih cifara ali će računar računati uvek sa onolikim brojem *značajnih* cifara koliko je navedeno u tabeli 12.2 (nule levo od prve značajne* cifre ne ulaze u taj broj!). Drugim rečima, ako se ne upotrebi oblik za dvostruku tačnost, kod većine računara realnu konstantu ima smisla pisati samo sa 7 ili 8 značajnih cifara.

Realna konstanta se u FORTRANu može pisati još na jedan način: jedan decimalni broj, za njim slovo E i za ovim jedan jednocifreni ili dvocifreni izložilac stepena osnove 10, sa kojim se množi taj decimalni broj. Za razliku od internog oblika pokretnog zareza, o kojem je bilo reči, ovakav oblik zovemo spoljnim ili eksternim oblikom pokretnog zareza. Maksimalna apsolutna vrednost tako napisane konstante ista je kao i za oblik pisanja bez izložioca. Ovakvim načinom uprošćava se pisanje vrlo velikih i vrlo malih realnih brojeva. Evo nekoliko primera ispravno napisanih konstanti na taj način:

5.0E+6	$(5.0 \cdot 10^{6})$
—7.E—12	$(-7.0 \cdot 10^{-12})$
6.205E11	$(6.205 \cdot 10^{11})$
—.1E7	$(-0.1 \cdot 10^{7})$

Za većinu računara, realna konstanta dvostruke tačnosti piše se samo u ovom obliku sa izložiocem, jedino se umesto slova E (exponent) upotrebljava slovo D (double precision). Samo kod nekih

* na pr. kod broja —0.000278400 ($= -0.2784 \cdot 10^{-4}$) beznačajne cifre su sve nule levo od cifre 2.

većih računara* takva konstanta se može pisati i u obliku bez izložioca ali tada broj značajnih cifara mora biti bar za jedan veći od maksimalnog broja kod obične tačnosti. Pogledajmo nekoliko primera pravilnog pisanja konstante u ovom obliku:

1.7D0	$(1.7 \cdot 10^0)$
6.0D6	$(6.0 \cdot 10^6)$
3.6D—4	$(3.6 \cdot 10^{-4})$
3.14159265359D0	$(3.14159265359 \cdot 10^0)$

(Broj u poslednjem primeru bio bi i bez izložioca ispravna konstanta dvostruke tačnosti, ali samo za neke veće računare*).

Maksimalna apsolutna vrednost realne konstante dvostruke tačnosti ista je kao i za običnu realnu konstantu, a zavisi od računara (vidi tab. 12.2). Ova povećana tačnost u radu sa decimalnim brojevima najčešće je potrebna da bi se umanjio efekat greške usled zaokrugljivanja ili odbacivanja cifara kada se izvodi dugi niz aritmetičkih operacija sa realnim brojevima.

Sada možemo pogledati i primere nepravilno napisanih realnih konstanti u obliku sa izložiocem E ili D:

25.3E105	(vrednost veća od maksimalno dozvoljene)
5.7642E2.5	(izložilac mora biti ceo broj)
E+6	(sam izložilac nije dozvoljen)
7.9D	(nedostaje vrednost izložioca)

Najzad, navedimo i nekoliko primera kompleksnih konstanti, mada se sa njima može operisati samo u FORTRAN-programima za veće računare:

(2.8, —0.73)	2.8 — 0.73i
(—3.0E+02, .22E+01)	—300.0 + 2.2i
(7.0E+04, .36E+02)	70000 + 36.0i
(2.6, 0.0)	2.6 + 0.0i (čisto realni broj)
(0.0, —4.8)	0.0 — 4.8i (čisto imaginarni broj)
(4.6D1, 1.864275319D0)	46.0 + 1.864275319i

Kao što je već rečeno za kompleksne konstante u FORTRANu, iz ovih primera se vidi da se one uvek pišu u zagradama, da se realni deo odvaja zarezom od imaginarnog i da se obe realne konstante (realni i imaginarni deo kompleksnog broja) definišu istom tačnošću.

* na pr. IBM 360, prevodilac H.

Poslednja, peta vrsta konstanti u FORTRANu, nema karakter brojne veličine. To su tzv. logičke konstante, čije su jedine dve vrednosti .TRUE. (»tačno« ili »jeste«) i .FALSE. (»pogrešno« ili »nije«). Logičke veličine i operacije sa njima u FORTRANu postoje kod najvećeg broja računara. O njima će biti reči kasnije.

2.2 *PROMENLJIVE I NJIHOVI SIMBOLI*

Rekli smo da se naziv *promenljiva* upotrebljava u FORTRANu za one veličine koje se u izvornom programu pojavljuju kao simboli, a ne kao posebne brojne vrednosti, i koje prema tome mogu u toku izvršavanja mašinskog programa u računaru imati sukcesivno više različitih vrednosti.

Promenljive u FORTRANu dele se na iste one vrste koje smo videli kod FORTRAN-konstanti. Vrednosti koje mogu imati promenljive u programu podležu pravilima i ograničenjima za odgovarajuću vrstu konstante. Međutim, kako se određuje vrsta svake promenljive, tj. kako se u programu zna da li se radi o celobrojnoj, običnoj realnoj, tačnijoj realnoj, kompleksnoj ili logičkoj promenljivoj — o tome će biti reči u ovom odeljku.

Simbol promenljive u FORTRANu, bilo koje vrste, sastoji se od jednog do najviše šest znakova i to isključivo slova i cifara, s tim da prvi (ili jedini) znak mora u svakom slučaju biti slovo. Tako su na primer A, B, IKS, GAMA, SUMA, X1, X2, YBAR — sve dozvoljeni simboli promenljivih. Mogu se upotrebljavati sva velika štampana slova internacionalnog alfabeta.

Prvo, univerzalno i tradicionalno pravilo FORTRANa omogućuje razlikovanje celobrojnih i običnih realnih promenljivih po prvom slovu njihovih simbola: ako simbol počinje slovom I, J, K, L, M ili N, promenljiva je celobrojna (INTEGER), dok svako drugo slovo na mestu prvog znaka simbola automatski deklariše promenljivu kao realnu, obične tačnosti (REAL). Dejstvo ovog pravila može biti poništeno naredbama za eksplicitno ili implicitno deklarisanje vrste promenljivih; ukoliko to nije učinjeno, ono i dalje važi za sve promenljive u progamu. Druge vrste promenljivih, DOUBLE PRECISION, COMPLEX i LOGICAL, ne mogu se deklarisati po ovom pravilu. Ovo pravilo se često naziva i *unutrašnja konvencija* FORTRANa.

Drugi način za određivanje ili deklarisanje vrste promenljivih pružaju tzv. eksplicitne naredbe za deklarisanje vrste. Takva naredba se sastoji od jednog atributa vrste (INTEGER, REAL, DOU-

BLE PRECISION, COMPLEX, LOGICAL) i niza simbola promenljivih, međusobno odvojenih zarezima. Na primer:

COMPLEX IKS, B, SUMA

znači da će IKS, B i SUMA biti kroz ceo program tretirane kao kompleksne promenljive, dok će za sve ostale promenljive i dalje važiti unutrašnja konvencija.

Treći način za deklarisanje vrste promenljivih, naredba IMPLICIT, postoji samo kod većih računara. Ovom naredbom deklarišu se vrste promenljivih na način vrlo sličan unutrašnjoj konvenciji, sa tom razlikom što se početna slova mogu slobodno birati. Na primer, ako hoćemo da sve promenljive čije prvo slovo pripada grupi A-H budu celobrojne, sve promenljive koje počinju slovima I-K realne, a sve promenljive koje počinju slovima L-N logičke, tada pišemo naredbu:

IMPLICIT INTEGER (A-H), REAL (I-K), LOGICAL (L-N).

Ako sada pretpostavimo da se poslednje dve naredbe, COMPLEX i IMPLICIT, nalaze u istom izvornom programu, tada će za taj program promenljive IKS, B i SUMA biti kompleksne, sve promenljive koje počinju slovima I-K i O-Z biće realne, celobrojne će biti promenljive koje počinju slovima A-H, dok će logičke promenljive biti sve one koje počinju slovima L-N. Na ovom primeru ujedno vidimo i rangove koje imaju tri načina za deklarisanje vrste promenljivih u FORTRAN-programu: »najstarija« je eksplicitna naredba, zatim naredba IMPLICIT, zatim unutrašnja konvencija.

Kod izbora simbola za promenljive najbolje je, gde god je to moguće, postupati po unutrašnjoj konvenciji FORTRANa. Kasnije ćemo videti da postoji još jedno ograničenje u izboru simbola: on ne sme biti identičan sa nazivom neke od standardnih matematičkih funkcija FORTRANa niti sa nazivom nekog potprograma koji se uključuje u rad našeg programa.

Radi bolje dokumentacije programa, korisno je da se simboli promenljivih biraju tako da podsećaju na fizički smisao predstavljenih veličina. Ako se taj princip dosledno primenjuje, funkciju programa će moći da razume i drugo lice, a ne samo autor. Na primer, za izračunavanje puta koji neko vozilo pređe za određeno vreme, vozeći datom brzinom, mogla bi se u izvornom programu napisati sledeća naredba:

X = Y*Z

gdje je zvezdica operator za množenje. Za onoga ko kasnije čita program, ista naredba bi imala više smisla ako bi bila napisana ovako:

PUT = BRZINA*VREME ili S = V*T

Treba, međutim, imati u vidu da FORTRAN-prevodilac ne pridaje nikakav smisao simbolima promenljivih; ukoliko to eksplicitnim i implicitnim naredbama nije izričito utvrđeno, prevodilac samo ispituje prvo slovo, da utvrdi da li je promenljiva celobrojna ili je realna. Tako na primer, simbol B7 *ne znači* B puta 7 niti B na sedmi stepen niti »B indeks sedam«. Isto tako, svaka konkretna kombinacija slova i cifara predstavlja poseban simbol, pa prema tome i novu promenljivu. Na primer, simbol ABC nije isto što i simbol BAC, a simboli A, AB i AB7 svi predstavljaju potpuno zasebne veličine, koje jedna s drugom nemaju nikakve veze.

2.3 *ARITMETIČKI OPERATORI I ARITMETIČKI IZRAZI*

U FORTRANu se mogu primenjivati operatori za pet osnovnih aritmetičkih operacija: sabiranje, oduzimanje, množenje, deljenje i stepenovanje. Simboli za te operatore su sledeći:

sabiranje	+
oduzimanje	—
množenje	*
deljenje	/
stepenovanje	**

Neuobičajeni simboli operatora za množenje, stepenovanje i donekle za deljenje, potiču otuda što se u FORTRANu iz tehničkih razloga izbegavaju komplikovani oblici pisanja (razlomačka crta u horizontalnom položaju, stepeni ili indeksi u gornjem odnosno donjem poluredu).

Kombinacija od dve zvezdice (operator stepenovanja) tretira se kao jedinstven simbol; ne može biti zabune između ** i * jer je, kao što ćemo videti, po pravilima FORTRANa zabranjeno pisati dva aritmetička operatora jedan do drugog. Navedene operacije su jedine moguće aritmetičke operacije u FORTRANu; svaka druga matematička operacija mora se definisati pomoću gornjih pet, koristeći još jedino standardne matematičke funkcije, o kojima će biti reči kasnije.

Najprostiji *aritmetički izraz* u FORTRANu ima samo jedan član, koji može biti konstanta, promenljiva ili standardna matematička funkcija (vidi 3.3). Složeniji izrazi imaju više takvih članova, koji su povezani aritmetičkim operatorima i grupisani pomoću zagrada. U tabeli 2.1 vidimo nekoliko primera za aritmetičke izraze u FORTRANu i njihova matematička značenja.

TABELA 2.1

Izraz	Značenje
K	Vrednost celobrojne promjenljive K
3.14159	Vrednost realne konstante 3.14159
A+2.1828	Zbir vrednosti A i 2.1828
ALFA—BETA	Razlika vrednosti ALFA i BETA
X*Y	Proizvod vrednosti X i Y
OMEGA/6.2832	Količnik vrednosti OMEGA i 6.2832
C**2	Kvadrat vrednosti C
(A+F)/(X+2.0)	Zbir vrednosti A i F podeljen zbirom vrednosti X i 2
1./(X**2+Y**3)	Recipročna vrednost izraza $(X^2 + Y^3)$

Kada se pišu aritmetički izrazi u izvornom programu, mora se voditi računa o sledećim pravilima FORTRANa za aritmetičke izraze:

1. Dva aritmetička operatora se ne smeju pisati jedan do drugog. Na primer, A*—B nije pravilan aritmetički izraz, dok A*(—B) jeste.
2. Kao i u matematici, grupisanje članova izraza vrši se pomoću zagrada. Tako izraz (X+Y)**3 znači $(x + y)^3$, dok X+Y**3 znači $x + y^3$. Isto tako, A—B+C i A—(B+C) su oba dozvoljeni izrazi ali imaju različita značenja. Zagrade, prema tome, daju veći prioritet operacijama koje su njima obuhvaćene.
3. Vrednost izraza se izračunava izvođenjem operacija s leva na desno, uzimajući pri tome u obzir sledeće rangove operatora:

izračunavanje vrednosti funkcije	1 (najviši)
stepenovanje	2
množenje i deljenje	3
sabiranje i oduzimanje	4 (najniži)

Ako je prvi operator višeg ili jednakog ranga u odnosu na drugi, izvršava se prva operacija. Ako nije, drugi operator se upoređuje sa trećim i tako dalje. Kada se dođe do kraja izraza, preostale operacije se izvršavaju obrnutim redom. Na primer, izračunavanje vrednosti izraza A*B + C*D **I ide sledećim redom:

A*B	vrednost je X (množenje)	(X + C*D**I)
D**I	vrednost je Y (stepenovanje)	(X + C*Y)
C*Y	vrednost je Z (množenje)	(X + Z)
X + Z	vrednost izraza (sabiranje)	

Drugi primer nam pokazuje kombinovani uticaj pravila 2 i 3. Izraz B + ((A + B)*C) + A**2 izračunavaće se sledećim redom:

(A + B)	vrednost je X	(B + (X*C) + A**2)
(X*C)	vrednost je Y	(B + Y + A**2)
B + Y	vrednost je W	(W + A**2)
A**2	vrednost je Z	(W + Z)
W + Z	vrednost izraza	

4. Kada u izrazu imamo niz uzastopnih stepenovanja, na pr. a^{b^c} (A**B**C), vrednost takvog složenog stepena izračunava se izvršavanjem operacija s desna na levo, dakle po obrascu $a^{(b^c)}$. Ako želimo $(a^b)^c$, moramo i u FORTRANu pisati (A**B)**C.
5. Vrsta kojoj pripada izračunata vrednost izraza, tj. da li će ta vrednost biti celobrojna, realna, dvostruko tačna ili kompleksna konstanta, zavisi od toga koje su vrste pojedini članovi izraza. FORTRAN-prevodioci za izvesne tipove računara ne dozvoljavaju mešanje veličina raznih vrsta u istom izrazu. Izuzetno od tog pravila mogu se izrazi, čiji su članovi realne veličine, stepenovati izložiocem koji je celobrojna veličina. Kod tih prevodilaca, vrednost izraza pripada jedinstvenoj vrsti svih njegovih članova, a u pomenutom izuzetnom slučaju vrednost je realna veličina.
Prevodioci mnogih savremenih računara (i većih starijih) dozvoljavaju, međutim, mešovite izraze, mada usvojeni standardi ni za osnovni ni za potpuni FORTRAN IV to ne zahtevaju. Vrsta izračunate vrednosti izraza može se odrediti primenom tabela 2.2 i 2.3 i pravila 3.

TABELA 2.2

+ — * /	Ceo broj	Realan broj	Dvostruka tačnost	Kompleksan broj
Ceo broj	ceo broj	realan broj	dvostruka tačnost	kompleksan broj
Realan broj	realan broj	realan broj	dvostruka tačnost	kompleksan broj
Dvostruka tačnost	dvostruka tačnost	dvostruka tačnost	dvostruka tačnost	kompleksan broj*
Kompleksan broj	kompleksan broj	kompleksan broj	kompleksan broj*	kompleksan broj*

TABELA 2.3

**	Ceo broj	Realan broj	Dvostruka tačnost	Kompleksan broj
Ceo broj	ceo broj	realan broj	dvostruka tačnost	—
Realan broj	realan broj	realan broj	dvostruka tačnost	—
Dvostruka tačnost	dvostruka tačnost	dvostruka tačnost	dvostruka tačnost	—
Kompleksan broj	kompleksan broj**	—	—	—

6. Aritmetički operatori se nikada ne podrazumevaju nego se uvek moraju pisati. Poznato je, naime, da se u matematici podrazumeva množenje kada se napiše na pr. ab ili $(a+b)(c+d)$. U FORTRANu se u ovakvim slučajevima mora pisati A*B odnosno (A + B)*(C + D).
7. Pri deljenju celih brojeva, kada su i deljenik i delilac celi brojevi, količnik je takođe ceo broj. Njegova vrednost se dobija kada se odbaci decimalni deo tačnog rezultata. Na

* i realni i imaginarni deo su realni brojevi dvostruke tačnosti.
** izložilac ceo broj.

primer, ako je I = 7 i J = 2, tada će izraz I/J imati vrednost 3 jer se decimali odbacuju.

U tabeli 2.4 dato je nekoliko primera pravilnog i nepravilnog pisanja aritmetičkih izraza na FORTRANu.

TABELA 2.4

Matematički izraz	Pravilno napisan izraz	Nepravilno napisan izraz
$a \cdot b$ ili ab	A*B	AB (nema operatora)
$a \cdot (-b)$	A*(—B) ili —A*B	A*—B (dva operatora jedan do drugog)
$a + 2$	A + 2.0	A + 2 (mešovit izraz)*
$-(a + b)$	—(A + B)	—A + B ili —+A + B
a^{i+2}	A**(I + 2)	A**I + 2 (= $A^I + 2$, mešovit izraz)*
$a^{b+2} \cdot c$	A**(B + 2.0)*C	A**B + 2.0*C (= $A^B + 2.0$*C)
$\frac{a \cdot b}{c \cdot d}$	A*B/(C*D) ili A/C*B/D	A*B/C*D $\left(= \frac{A \cdot B \cdot D}{C}\right)$
$\left(\frac{a + b}{c}\right)^{2.5}$	((A + B)/C)**2.5	(A + B)/C**2.5 $\left(= \frac{A + B}{C^{2.5}}\right)$
$a[x + b(x + c)]$	A*(X + B*(X + C))	A(X + B(X + C))

2.4 *SPECIFIČNOSTI ARITMETIČKIH OPERACIJA*

Bez obzira na to što FORTRAN-prevodioci nekih računara dozvoljavaju upotrebu raznih vrsta brojeva u istom izrazu, većinu aritmetičkih operacija sa podacima treba programirati kao operacije sa realnim veličinama, da bi se izbegli problemi oko određivanja mesta decimalne tačke. Ipak, često će se javljati i slučajevi gde je nužno da se aritmetičke operacije izvode sa celim brojevima. Pošto tu početnika očekuje najviše nezgoda, neophodno je da precizno izložimo kako se u računaru izvode aritmetičke operacije sa celim brojevima.

* ovi izrazi su nepravilno napisani samo za one FORTRAN-prevodioce koji ne dozvoljavaju mešovite izraze; za druge prevodioce važe tabele 2.2 i 2.3

Osnovna karakteristika je u tome što, kada se kao rezultat računanja dobije broj koji nije ceo, od takvog rezultata se zadržava samo celobrojna vrednost, dok se decimalni deo prosto *odbacuje*, bez zaokrugljivanja. Tako će, na primer, rezultat deljenja celog broja 5 sa celim brojem 3 biti 1, a ne 2. Ova karakteristika iskazana je kao pravilo 7 i odnosi se samo na deljenje; operacije sabiranja, oduzimanja i množenja celih brojeva daće uvek tačne rezultate u obliku FORTRAN-celog broja.

Izrazi u kojima se javljaju nizovi raznih operacija, među kojima ima i deljenja, zahtevaju posebnu pažnju što se tiče redosleda po kojem se operacije izvršavaju. To je jedna prilika kada dolazi do praktičnog izražaja značaj pravila 2 i 3. Na primer, ako napišemo izraz 5/3*6, rezultat će biti 6, a ne ni 10 niti 12. Niz operacija istog ranga, ako nema zagrada, obavlja se redom s leva na desno (izuzetak je stepenovanje). Tako će ovde najpre biti podeljen ceo broj 5 sa celim brojem 3, decimalni deo rezultata biće odbačen i ostaće samo 1, što pomnoženo sa 6 daje 6. S druge strane, da je izraz bio napisan na jedan od sledećih načina: 5*6/3, 6*5/3, 5*(6/3) ili (6/3)*5, rezultat bi bio 10. Sa realnim konstantama, ovaj izraz bi bio tačno izračunat na onoliko značajnih cifara koliko može imati jedan realan broj u datom računaru. Možda bi samo cifra na poslednjem mestu bila netačna. Isti rezultat bi se dobio u svim navedenim oblicima tog izraza.

Čak i kada se aritmetičke operacije izvode sa realnim brojevima, tačnost rezultata nije uvek apsolutna. Usled grešaka zaokrugljivanja i činjenice da mnogi razlomci nemaju tačne vrednosti kada se predstave kao decimalni brojevi, rezultati koji bi trebalo da budu celi brojevi ili konačni decimalni razlomci nisu uvek takvi. Na primer, neka se izračunava vrednost izraza 1.0/6.0*3.0; po pravilu 3, operacije će se izvoditi s leva na desno, pa će već rezultat prve operacije, na sedam značajnih cifara, biti 0.1666666. Kada se to pomnoži sa 3, dobićemo 0.4999998, a ne 0.5000000. Ako bi se negde u programu vršilo upoređivanje te vrednosti sa konstantom 0.5, rezultat upoređivanja bi bio »nije jednako«, što može odvesti dalji tok programa u pogrešnom pravcu.

Zbog ograničenog broja cifara za realne brojeve u računaru, sabiranje i oduzimanje malih i velikih brojeva može dovesti do potpunog gubitka uticaja malih brojeva na konačan rezultat. Na primer, kada se izračunava vrednost izraza

$$0.8000000 + 3589763.0 - 3589762.0$$

rezultat prve operacije (sedam značajnih cifara) biće 3589763.0; 0.8 nije uopšte ušlo u rezultat. Oduzimanjem 3589762.0 konačna vrednost izraza postaje 1.000000. Da je, međutim, isti izraz bio napisan sa zagradama:

$$0.8000000 + (3589763.0 - 3589762.0)$$

tada bi po pravilima 2 i 3 najpre bila izračunata vrednost izraza u zagradama, koja je jednaka 1.000000; sabiranjem sa 0.8 dobila bi se konačna vrednost 1.800000.

I na kraju još jedno upozorenje. Mada je u većini slučajeva dozvoljeni interval apsolutnih vrednosti brojeva sasvim dovoljan, uvek postoji mogućnost da neki rezultat premaši te granice. Iz tabele 12.2 vidimo da je apsolutna vrednost celih brojeva kod većine računara maksimalno $2^{31}-1$, dok je apsolutna vrednost za realne brojeve obično ograničena na interval $0-\dot{+}10^{\pm 75}$, uključiv nulu. Ako u toku rada neki rezultat premaši te granice, računar će o tome dati određenu indikaciju.

VEŽBE

(Zvezdica ispred broja zadatka znači da je za taj zadatak dato rešenje u posebnom prilogu, na kraju knjige).

1. Sledeće brojeve napiši u obliku realnih konstanti:

   16; 3.6802; —10.000; 10^{14}; 0,000000000637; —1; -10^{-10}

*2. Sve sledeće realne konstante nepravilno su napisane; zašto?

   2,768.035.0; +276; 1.8E86; 3E—7

*3. Da li svi sledeći parovi realnih konstanti predstavljaju iste brojeve?

a.	38.6	+38.6
b.	54000	5.4E4
c.	0.00006	.6E—4
d.	1.0	1.
e.	.906E5	+906.0E+2

4. Prema unutrašnjoj konvenciji FORTRANa, koji od sledećih simbola su pravilno napisani simboli za realne, a koji za celobrojne promenljive? Da li svi pravilno napisani simboli predstavljaju različite promenljive? Najpre obeleži nepravilno napisane simbole.

   X, I12G, ABS, X+2, XP2, NEPOZ, 42G, LOG, XKVADRAT, DELTA, MI, A*B, X1.4, (Y61), Y—6, GAMA81, AI, IA, X12, 1X2, GAMA, DELTA2

5. Za sledeće matematičke izraze napiši odgovarajuće aritmetičke izraze na FORTRANu:

*a. $x + y^3$

b. $(x + y)^3$

c. x^4

*d. $a + \frac{b}{c}$

e. $\frac{a + b}{c}$

*f. $a + \frac{b}{c + d}$

g. $\frac{a + b}{c + d}$

*h. $\left(\frac{a + b}{c + d}\right)^2 + x^2$

i. $\frac{a + b}{c + \frac{d}{f + g}}$

*j. $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$

*k. $\left(\frac{x}{y}\right)^{g-1}$

l. $\frac{\frac{a}{b} - 1}{g\left(\frac{g}{d} - 1\right)}$

6. Dato je nekoliko matematičkih izraza i odgovarajućih FORTRAN-izraza, u kojima je vrsta promenljivih deklarisana prema unutrašnjoj konvenciji FORTRANa. Svaki FORTRAN-izraz sadrži bar po jednu grešku, pri čemu se greškom smatra i mešanje vrsta promenljivih u istom izrazu. Treba pronaći greške i napisati pravilne izraze:

a.	$(x + y)^4$	X + Y**4
*b.	$\frac{x + 2}{y + 4}$	X + 2.0/Y + 4.0
c.	$\frac{a \cdot b}{c + 2}$	AB/(C + 2.0)
d.	$-\frac{(-x + y - 16)}{y^3}$	—(—X + Y — 16)/Y**3
*e.	$\frac{(x + a + \pi)^2}{2z}$	(X + A + 3.1416)/2.0*Z) **2
f.	$\left(\frac{x}{y}\right)^{n-1}$	(X/Y)**N — 1
*g.	$\left(\frac{x}{y}\right)^{r-1}$	(X/Y)**(R—1)
h.	$\frac{a}{b} + \frac{c \cdot d}{f \cdot g \cdot h}$	A/B + CD/FGH
i.	(a + b) (c + d)	A + B*C + D
*j.	$a + bx + cx^2 + dx^3 = a + x[b + x(c+dx)]$	A + X*(B + X*(C + D*X))
k.	$\frac{1{,}500.053x + 10^5}{3{,}706.663x + 10^5}$	(1,500.053X + 1E5)/(3,706.663X + 1E5)
l.	$\frac{1}{a^2}\left(\frac{r}{10}\right)^a$	1/A**2*(R/10) **A

7. Promenljive A, I i F deklarisane su kao celobrojne, promenljiva R kao realna, promenljiva D kao realna dvostruke tačnosti i promenljive C i E kao kompleksne. Odredi vrstu koju će imati izračunate vrednosti sledećih izraza ako FORTRAN-prevodilac dozvoljava mešovite aritmetičke izraze:

a. R*5 *b. A +3 *c. C + 2.9D10 d. D/F + 19 *e. C — 18.7E05 *f. A/I — E

8. Sa vrstama promenljivih koje su deklarisane kao u prethodnom zadatku, odredi vrstu koju će posle izračunavanja imati vrednosti sledećih izraza:

a. R**(A+2) *b. R**I c. I**F *d. 7.98E21**R *g. C**A e. R**2.1E5 f. A**D

Glava 3

ARITMETIČKE NAREDBE I STANDARDNE FUNKCIJE

3.1 *PISANJE I BUŠENJE PROGRAMA*

Već je rečeno da se FORTRAN-program sastoji od niza *naredbi*, od kojih svaka predstavlja ili jedan nalog da se izvrši neka operacija ili daje prevodiocu određenu informaciju o FORTRAN-programu. FORTRAN-program se piše na specijalnom obrascu (sl. 3.1), na kome svaka naredba zauzima jedan ili više redova. Zatim se podaci iz svakog reda buše u po jednu karticu (sl. 3.2); paket kartica stavlja se u ulaznu jedinicu računara, a u njegovu memoriju unosi se program-prevodilac. Računar sada »čita« kartice i pod kontrolom prevodioca razvija FORTRAN-naredbe u mašinske instrukcije, čijim će se izvršavanjem kasnije u računaru izvoditi postupak opisan naredbama.

Izgled programskog obrasca, koji vidimo na slici 3.1, poslužiće nam da objasnimo ulogu pojedinih delova obrasca. Sve kasnije primere u ovoj knjizi, bilo da se radi o pojedinim FORTRAN-naredbama ili o celim programima, čitalac treba da zamisli kao da su napisani na programskom obrascu, po pravilima koja su izložena niže.

Brojevi iznad prvog reda obrasca znače redne brojeve kolona u karticama, u koje će se bušiti sadržaj iz pojedinih redova obrasca. Prvo *polje* u obrascu (pod poljem se podrazumeva grupa susednih kolona), i to kolone 1 do 5, sadrži *broj naredbe*, ako naredba ima svoj poseban broj. Uloga ovih brojeva opisana je u Glavi 5.

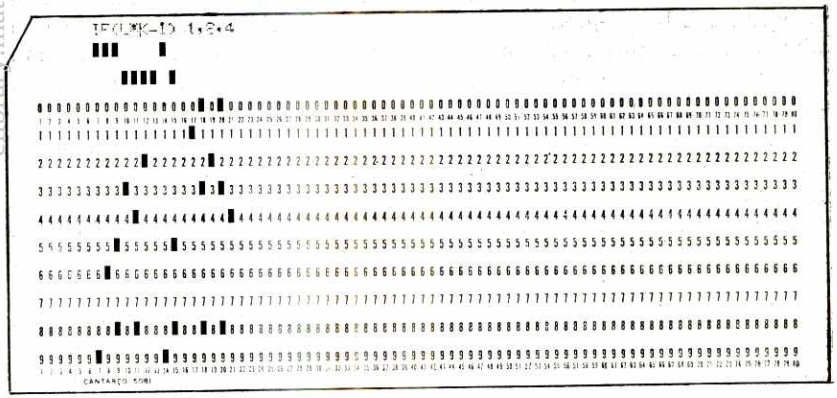
Kolona 1 ima još jednu ulogu, a to je da označi kada dotična kartica sadrži samo primedbu ili objašnjenje (komentar). Ako se u toj koloni nalazi slovo C, tada prevodilac »zna« da se u toj kartici ne nalazi nikakva naredba i sadržaj takve kartice neće prevoditi. Mada se kartice sa objašnjenjima ne obrađuju kao ostale FOR-

— C ZA NASTAVAK

BROJ NAREDBE	FORTRAN NAREDBE
C	PROGRAM ZA NALAZENJE PROSTIH BROJEVA
	WRITE (6,8)
8	FORMAT (40H OVO JE NIZ PROSTIH BROJEVA OD 1 DO 1000
	119X,1H1/19X,1H2/19X,1H3)
	I=5
3	A=I
	A=SQRT(A)
	J=A
	DO 1 K=3,J,2
	L=I/K
	IF(L*K-I) 1,2,4
1	CONTINUE
	WRITE (6,5) I
5	FORMAT (I20)
2	I=I+2
	IF(1000-I) 7,4,3
4	WRITE (6,9)
9	FORMAT (18H GREŠKA U PROGRAMU)
7	WRITE (6,6)
6	FORMAT (14H KRAJ PROGRAMA)
	STOP 999
	END

Slika 3.1 Programski obrazac (sa jednim FORTRAN-programom)

TRAN-naredbe, njihov sadržaj će biti odštampan u tzv. listi programa, koja se dobija pri procesu prevođenja FORTRAN-programa. Zato kartice sa slovom C u prvoj koloni treba koristiti za davanje naslova, podnaslova, primedbi i komentara u vezi sa programom, što će koristiti svakom ko kasnije bude imao potrebu da čita liste programa. Obilna upotreba C-kartica može biti od koristi i samom



Slika 3.2 Kartica sa jednom naredbom (iz programa na slici 3.1)

autoru programa, ako treba kasnije da modifikuje svoj program, a to je veoma čest slučaj uopšte kod svih programa za računare. FORTRAN-programi se i inače mnogo lakše čitaju nego oni koji su pisani na simboličkom ili mašinskom jeziku; ipak, ako nema objašnjenja, kasnije se teško može razumeti logika svakog složenijeg programa.

U koloni 6 stavlja se znak koji identifikuje tzv. *produžne kartice*. Ako naredba može cela da stane u jedan red na obrascu i samim tim i u jednu karticu, kolona 6 se ostavlja nebušena (tada sadrži znak »blanko«) ili se u nju buši nula. Ako je, međutim, za naredbu potrebno više od jedne kartice, tada se u koloni 6 svake kartice osim prve mora bušiti neki znak različit od znaka blanko ili nule, dok se u koloni 6 prve kartice mora nalaziti blanko ili nula. Radi preglednosti preporučuje se da se kartice svake duže naredbe numerišu u ovoj koloni brojevima i slovima u rastućem redosledu, počev sa nulom za prvu karticu, pa zaključno sa brojem 9 u desetoj kartici i dalje sa slovima A — J u karticama 11—20.

Jedna naredba može dakle da zauzme maksimalno 20 kartica, od kojih su 19 produžne*.

Tekst naredbe buši se u kolone 7 do 72. *FORTRAN-prevodilac ne vodi računa o znacima blanko u polju koje se sastoji od kolona 7 — 72.* To znači da ih možemo koristiti gde god je to potrebno, da bi tekst bio jasniji ili pregledniji. Sama naredba ne mora da počinje u koloni 7. Često se izvesni redovi programa namerno pišu sa početkom od kolone 9, 12 i sl., da bi kasnije odštampana lista programa bila preglednija. Iz istog razloga praktikuje se da se sa obe strane aritmetičkih operatora + i — ostavlja po jedan znak blanko. Takve i slične konvencije na potpunom su raspolaganju svakom ko piše program.

Pošto je programski obrazac direktno vezan za karticu, neophodno je da se na njemu jasno označi koliko se praznih mesta ili znakova blanko ostavlja na svakom mestu gde se to želi. Time se omogućuje naknadno verificiranje izbušenih kartica. Radi toga su na obrascima za pisanje FORTRAN-programa vertikalnim linijama odvojena mesta za upisivanje svakog znaka.

FORTRAN-prevodioci ne uzimaju u obzir tekst koji je bušen u polje od 73. do 80. kolone i to polje se može koristiti za bilo kakve identifikacione podatke, obično za kratak šifrovan naziv programa.

Bitno je da se obrasci popunjavaju veoma pažljivo i tačno u svim detaljima. Naredbe i podaci se uvek moraju pisati tačno u obliku koji je opisan u ovom ili sličnim udžbenicima; ako se na pogrešnom mestu napiše ili se izostavi samo jedan zarez, program uopšte neće biti ili će biti pogrešno preveden. Naročito se preporučuje da se pri pisanju koriste samo velika štampana slova i da se na nedvosmislen način ispisuju oni znaci koji se inače po svom obliku lako zamenjuju sa drugima. Postoji više različitih konvencija za bolje međusobno razlikovanje takvih znakova; jedan od načina citiran je niže.

Cifra jedan:	1	Slovo I:	I
Cifra nula:	0	Slovo O:	Ø
Cifra dva:	2	Slovo Z:	Z
Cifra pet:	5	Slovo S:	S
Cifra šest:	6	Slovo G:	G

Grčka slova kao i posebna slova jugoslovenske abecede, razume se, nisu ~~dozvoljena~~ moguća.

* izuzetke videti u tabeli 12.2

3.2 *ARITMETIČKE NAREDBE*

Svrha najvažnije vrste naredbi u FORTRAN-programu, aritmetičke naredbe, jeste da simbolu jedne promenljive dodeli ili pripiše jednu vrednost. Opšti oblik aritmetičke naredbe je

$$a = b$$

gde je *a* simbol promenljive, koji se uvek piše bez algebarskog znaka ispred, a *b* može biti bilo kakav aritmetički izraz, prema ranijoj definiciji takvog izraza u FORTRANu. Znak jednakosti u aritmetičkoj naredbi nema isti smisao kao u matematici. Nije, na primer, moguće pisati algebarske jednakosti oblika $X - Y = (A + B)*(C - D)$, gde su na levoj strani nepoznate, a na desnoj poznate veličine. Na levoj strani znaka jednakosti u aritmetičkoj naredbi može se nalaziti samo jedan simbol promenljive. Znak jednakosti dobija na taj način sledeći smisao: *izračunata vrednost izraza na desnoj strani postaje vrednost promenljive čiji je simbol na levoj strani.* Takva definicija očigledno pretpostavlja da u momentu izvršavanja aritmetičke naredbe moraju biti poznate sve veličine pomoću kojih se izračunava vrednost izraza na desnoj strani znaka jednakosti. Naredba $A = B + C$ znači prema tome nalog računaru (preko prevodioca) da sabere vrednosti promenljivih B i C i da dobijenu vrednost zbira stavi na mesto određeno za promenljivu A. Time se gubi prethodna vrednost promenljive A, jer njeno mesto sada zauzima novodobijena vrednost zbira. Isto tako, naredba ALFA = 3.259 jeste nalog računaru da raniju vrednost promenljive ALFA zameni sa konstantom 3.259.

Primer sledeće aritmetičke naredbe još bolje objašnjava specifičan smisao koji u FORTRANu ima znak jednakosti: $N = N + 1$. U matematici ovakva jednakost ne može postojati ni za jednu vrednost N, dok u FORTRANu to može biti bilo koja vrednost u granicama dozvoljenih veličina. Ovakva vrsta aritmetičke naredbe veoma je česta u FORTRAN-programima i znači: *zameniti staru vrednost promenljive N novom vrednošću, uvećanom za jedinicu.*

Ranije smo videli da za neke FORTRAN-prevodioce aritmetički izraz mora biti homogen u pogledu vrsta pojedinačnih njegovih članova (ceo, realan, dvostruko tačan ili kompleksan broj), dok za druge vrstu koju će imati izračunata vrednost izraza određuju pravilo 3 i tabele 2.2 i 2.3. Međutim, u oba slučaja potpuno je dozvoljeno da promenljiva na levoj strani znaka jednakosti u aritmetičkoj naredbi bude jedne vrste, a vrednost izraza na desnoj strani druge vrste.

Posle izračunavanja vrednosti izraza na desnoj strani, najpre se interni oblik vrste kojoj pripada izračunata vrednost pretvara u interni oblik vrste kojoj pripada promenljiva na levoj strani. Tako pretvorena vrednost smešta se zatim na mesto određeno za promenljivu. Pogledajmo na nekoliko prostih primera kako će biti izvršene aritmetičke naredbe po gornjim pravilima.

Neka su u trenutku izvršavanja aritmetičke naredbe promenljive I i J celobrojne, A i B realne, C i D dvostruko tačne realne i promenljiva E kompleksna. Aritmetičke naredbe imaće sledeći efekat:

A = B	Na mesto vrednosti A smešta se tekuća vrednost promenljive B
J = B	Odbacivanjem decimala dobijen je najpre celobrojni deo promenljive B, koja se vrednost zatim smešta na mesto rezervisano za promenljivu J
A = I	Najpre se vrednost promenljive I pretvara u realnu veličinu, a zatim smešta na mesto rezervisano za promenljivu A
I = I + 1	Vrednost I zamenjuje se vrednošću I+1
E = I**J + D	I se stepenuje izložiocem J i rezultat se pretvara u dvostruko tačnu realnu veličinu, kojoj se dodaje vrednost D. Konačna vrednost izraza smešta se na mesto realnog dela kompleksne promenljive E. Imaginarni deo kompleksne promenljive dobija vrednost nula, a iste je vrste kao i realni deo.
A = C*D	Značajniji deo cifara proizvoda C sa D zamenjuje tekuću vrednost promenljive A
A = E	Na mesto gde se nalazi tekuća vrednost promenljive A smešta se realni deo kompleksne promenljive E
E = A	Vrednost A zamenjuje tekuću vrednost realnog dela kompleksne promenljive E; imaginarni deo promenljive E dobija vrednost nula
E = (1.0,2.0)	Na mesto rezervisano za kompleksnu promenljivu E smešta se kompleksna konstanta (1.0,2.0). Naredba E = (A,B), gde su A i B realne promenljive, nije dozvoljena.

Sa prva četiri primera ilustrovani su slučajevi koje dozvoljavaju svi FORTRAN-prevodioci; šesti primer takođe pripada toj

vrsti. Ostali slučajevi mogu nastupiti samo kod većih računara, čiji FORTRAN-prevodioci obuhvataju operacije sa kompleksnim brojevima. Iz praktičnih razloga, mi ćemo se dalje ograničiti samo na slučajeve koji imaju opšti značaj. Takođe ćemo, ukoliko to nije izričito navedeno kod pojedinih primera, uvek smatrati da su kao celobrojne deklarisane sve promenljive čiji simboli počinju slovima I, J, K, L, M i N. Realnim promenljivima obične tačnosti smatraćemo sve ostale promenljive. Ovaj dogovor će pod istim uslovima važiti kroz sve odeljke ove knjige, kako u primerima tako i u vežbama.

3.3 *STANDARDNE MATEMATIČKE FUNKCIJE*

Najčešće matematičke funkcije kao što je kvadratni koren, prirodni i dekadni logaritam, eksponencijalna funkcija, trigonometrijske funkcije i druge, postoje u FORTRANu kao gotovi potprogrami, pa se njihovo izračunavanje ne mora programirati pomoću aritmetičkih naredbi. Izbor raspoloživih funkcija zavisi od tipa i veličine računara. One koje su ovde navedene postoje kod svih FORTRAN-prevodilaca. Takve funkcije zovemo standardnim matematičkim funkcijama FORTRANa, za razliku od drugih funkcija koje, kao što ćemo kasnije videti, možemo sami programirati i na određeni način uključiti u rad našeg FORTRAN-programa.

Svaka standardna matematička funkcija FORTRANa ima svoj strogo određeni simbol ili naziv. Tako na primer, SQRT je naziv za funkciju $y = \sqrt{x}$ (*square root* znači: kvadratni koren), EXP je naziv za funkciju $y = e^x$ itd. Kod najvećeg broja ovih funkcija i argument i vrednost funkcije su realni brojevi obične tačnosti*, međutim to mogu biti i brojevi neke druge vrste. To se obično može videti već po samom nazivu funkcije. Na primer, SQRT je naziv funkcije koja daje realnu vrednost od realnog argumenta, DSQRT je naziv iste funkcije kada daje dvostruko tačnu vrednost od dvostruko tačnog argumenta, CSQRT kada su i vrednost i argument kompleksne veličine itd. U tabeli 12.3 na kraju knjige dat je potpun pregled standardnih matematičkih funkcija, koje su ugrađene u skoro sve FORTRAN-prevodioce.

Da bi se u FORTRAN-programu izračunala vrednost neke od standardnih matematičkih funkcija, potrebno je samo da se napiše

---

* kod trigonometrijskih funkcija vrednost argumenta je uvek u radijanima

njen naziv i iza njega jedan aritmetički izraz u zagradama. Na tom mestu računar će sam izračunati vrednost funkcije, za onu vrednost argumenta koja se dobija izračunavanjem izraza u zagradama.

Kao primer za upotrebu standardnih matematičkih funkcija u programu, uzmimo da nam je za dalji rad potrebna apsolutna vrednost promenljive X. Ako je X obična realna promenljiva, upotrebićemo u naredbi simbol ABS (X) i računar će izvršiti programiranu operaciju sa realnom konstantom $|X|$. Ako je X celobrojna promenljiva, njenu apsolutnu vrednost daće nam funkcija IABS (X), dok ćemo dvostruko tačnu apsolutnu vrednost dobiti ako napišemo DABS (X), s tim da je sada i promenljiva X dvostruko tačna realna veličina. Ako bismo, na primer, hteli kvadratni koren izraza $B^2 - 4AC$, imali bismo jednostavno da napišemo SQRT (B**2 — 4.*A*C).

Naziv ili simbol funkcije može se u naredbi pisati na svakom mestu gde je potrebna vrednost te funkcije. Ako, na primer, koristimo funkciju za kvadratni koren da bismo izračunali jedan od dvaju korenova potpune kvadratne jednačine, možemo napisati sledeću naredbu:

```
X1 = (—B + SQRT(B**2 — 4.0*A*C))/(2.*A)
```

Korisno je da pogledamo čemu sve služe zagrade u ovom primeru. Par zagrada koje obuhvataju diskriminantu $B^2 - 4AC$ potreban je zato što je diskriminanta argument funkcije SQRT. Spoljni par zagrada potreban je da bi sa 2A bio podeljen ceo obuhvaćeni izraz, a ne samo vrednost kvadratnog korena. Zagrade oko 2.*A obezbeđuju da delilac bude 2A, a ne samo konstanta 2.

Nezavisno od upotrebe standardnih funkcija, na ovom primeru možemo istaći još jednu osobinu koja je opšta za sve FORTRAN-aritmetičke naredbe: nijedna od promenljivih na desnoj strani neće promeniti svoju vrednost izvršavanjem naredbe u računaru, dok će stara vrednost promenljive na levoj strani biti uništena.

Kao sledeći primer, izračunajmo vrednost sledećeg algebarskog izraza:

$$m = 2 \cdot 10^{-9} x \left(\ln \frac{2x}{d} - 1 + \frac{d}{x} \right)$$

Veličina m u elektrotehnici predstavlja tzv. uzajamnu induktivnost dvaju provodnika dužine x, na rastojanju d jedan od drugoga. Ako su u trenutku izvršavanja naredbe poznate vrednosti

svih promenljivih koje čine izraz na desnoj strani, veličinu m možemo izračunati po sledećoj naredbi:

```
EM = 2.E—9*X*(ALOG(2.*X/D) — 1. + D/X)
```

Simbol EM je upotrebljen za promenljivu m da bi vrednost izraza na desnoj strani bila dodeljena promenljivoj u istom obliku u kojem je izračunata, tj. kao realna konstanta. Unutrašnja konvencija FORTRANa, koje se po dogovoru držimo u svim primerima, smatrala bi simbol M celobrojnom veličinom, pa bi u tom slučaju vrednost izraza bila pretvorena u ceo broj. Time bismo dobili i netačan rezultat jer bi bile odbačene decimalne cifre.

Sasvim je dozvoljeno da se u argumentu jedne funkcije nalazi druga funkcija. Primer za to imamo u aritmetičkoj naredbi za izračunavanje jednog tipa određenog integrala:

$$\int \frac{dx}{\cos x} = v = \ln \operatorname{tg}\left(\frac{\pi}{4} + \frac{x}{2}\right) + C$$

```
V = ALOG(TAN(3.14/4.0 + X/2.0) ) + C
```

Ako naš FORTRAN-prevodilac ne raspolaže funkcijom za izračunavanje tangensa ugla, tada bismo morali izvršiti smenu

$$\operatorname{tg} x = \frac{\sin x}{\cos x},$$

pa bi naredba za izračunavanje vrednosti integrala glasila:

```
V = ALOG(SIN(3.14/4.+X/2.)/COS(3.14/4.+X/2.) ) + C
```

Da se obezbedimo od negativnog broja u argumentu logaritma, naredba bi u stvari trebalo da glasi:

```
V = ALOG(ABS(SIN(3.14/4.+X/2.)/COS(3.14/4.+X/2.)))+C
```

Isti rezultat bi se dobio kada bismo napisali sledeći niz naredbi, koje se izvršavaju jedna za drugom, onim redom kako su napisane:

```
Y = 3.14*0.25 + X*0.5
SINY = SIN (Y)
COSY = COS (Y)
TGY = SINY/COSY
ABSTGY = ABS (TGY)
V = ALOG (ABSTGY) + C
```

Vidimo da se na ovaj način dobijaju prostiji izrazi u argumentima funkcija. Kod nekih FORTRAN-prevodilaca ovakav način pisanja standardnih funkcija je jedino dozvoljen. Kod drugih, iako takvog uslova nema, ovakav način je koristan ako se želi da se ima bolja preglednost operacija u naredbi. U praktičnom radu pri programiranju na FORTRANu ovakve zamene, koje vode uprošćavanju izraza, često se upotrebljavaju i tamo gde nema funkcija, samo da bi naredbe bile jednostavnije i lakše za naknadnu kontrolu napisanog programa.

Sa primerima u tabeli 3.1 ilustrovano je ispravno pisanje matematičkih obrazaca u obliku aritmetičkih naredbi; neki od obrazaca sadrže poznate matematičke funkcije. Izbor simbola za promenljivu na levoj strani potpuno je slobodan ali su u primerima simboli birani tako da podsećaju na original. Pretpostavljeno je da su vrednosti promenljivih na desnoj strani već određene pomoću ranijih naredbi u programu.

TABELA 3.1

Matematički obrazac	Aritmetička naredba
$r = \frac{a + bx}{c + dx}$	R = (A + B*X)/(C + D*X)
$y = \sin ax$	Y = SIN (A*X)
$y' = a \cdot \cos ax$	YPRIM = A*COS (A*X)
$\beta = -\frac{1}{2x} + \frac{a^2}{4x^2}$	BETA = —1./(2.*X) + A**2/(4.*X**2)
$f(y) = x\,\frac{x^2 - y^2}{x^2 + y^2}$	FY = X*(X**2 — Y**2)/(X **2 + Y**2)
$c = 1{,}112d\,\frac{r_1 \cdot r_2}{r_1 - r_2}$	C = 1.112*D*R1*R2/(R1 — R2)
$y = (10^{-6} + ax^3)^{2/3}$	Y = (1.0E-6 + A*X**3)**(2.0/3.0)
$j = 4K - 6k_1k_2$	J = 4*K — 6*K1*K2
$a = b \cdot \cos\gamma + c \cdot \cos\beta$	A = B*COS (GAMA) + C*COS (BETA)
$i_{novo} = i_{staro} + 1$	I = I + 1
$k = 12$	K = 12
$\pi = 3.141593$	PI = 3.141593
$m_{novo} = 2m_{staro} + 10j$	M = 2*M + 10*J

Primeri u tabeli 3.2 dati su da bi još jedanput bila istaknuta važnost pisanja FORTRAN-izraza i aritmetičkih naredbi u tačno propisanom obliku. Za svaku grešku dato je objašnjenje.

TABELA 3.2

Aritmetičke naredbe sa greškom	Objašnjenje
Y = 2.X + A	Nedostaje *
3.14 =. Y — A	Na levoj strani može biti samo simbol jedne promenljive
A = ((X + Y)A**2 + 16.9	Nejednak broj levih i desnih zagrada; nedostaje *
Z = 3,625.084.*SIGMA	Dekadne klase se ne smeju odvajati pomoću , i .
DY/DX = COS(X)	Simbol promenljive se može sastojati samo od slova i cifara
CG8 = 1./—.5*B**3	Dva aritmetička operatora ne mogu se pisati jedan do drugog, bez obzira na to što ovde minus nije operator za oduzimanje
YPRIM = K*X **(I—1)	Ova naredba sadrži grešku samo za prevodioce koji ne dozvoljavaju upotrebu celobrojnih i realnih veličina u istom izrazu

VEŽBE

(Zvezdica ispred broja zadatka znači da je za taj zadatak dato rešenje u posebnom prilogu, na kraju knjige).

1. Sledeće aritmetičke naredbe napisane su u programu jedna za drugom. Odredi vrstu i vrednost svih promenljivih na levoj strani. Vrednosti realnih promenljivih piši u internom obliku pokretnog zareza ($O.XXXXXXX \cdot 10^n$).

a. A = 25.03
b. B = 2.02
* c. P = A*B
* d. K = A*B
e. L = 250
f. R = 3*L
* g. Z = K/L
h. F = 15.0E+05
i. T = 0.6E—15
j. W = —0.6E—15
k. X = 3.0E+04 + A
l. Y = .0009874327
*m. J = Y

2. Isto kao u prethodnom zadatku, za sledeće naredbe:

* a. B = 2*6 + 1
* b. B = 2/3
c. B = 2.*6./4.
d. L = 2*10/4
* e. L = 2*(10/4)
* f. B = 2*(10/4)
g. B = 2.*(10./4.)
h. B = 2.0*(1.0E1/4.0)
i. B = 6.0*1.0/6.0
j. B = 6.0*(1.0/6.0)
* k. B = 1./3. + 1./3. + 1./3.
l. B = (4.0)**(3/2)
* m. B = (4.0)**(3./2.)
n. B = (4.0)**3./2.
* o. L = 19/4 + 5/4
p. B = 19/4 + 5/4
q. L = 100*(99/100)

3. Pronađi greške u sledećim aritmetičkim naredbama. FORTRAN--prevodilac ne dozvoljava upotrebu celobrojnih i realnih veličina u istom izrazu.

a. +Y = A +B
b. 84 = J
c. U — 7.25 = Z**0.33
d. X = (H + 5)**2
e. R*Z**2 + S*Z + T
f. IKS = I**G
g. W2 = D*— E + F**3
h. U = V + 3.5 = W + 5.2
i. S = 2.54X + PQ

* 4. Odredi vrstu i vrednost koje će promenljiva A dobiti po sledećem nizu FORTRAN-naredbi (uz pomoć tabela 2.2 i 2.3):

```
INTEGER Z
X = 2.0
Y = 7.0
Z = 4.69
A = Y + ((X + Y)/Z) + X**2
```

5. Napiši aritmetičke naredbe za sledeće matematičke obrasce. Izborom simbola za promenljive na levoj strani, obezbedi da njihove vrednosti budu realne veličine. FORTRAN-prevodilac ne dozvoljava upotrebu celobrojnih i realnih veličina u istom izrazu. Standardne funkcije date su u tabeli 12.3 na kraju knjige.

* a. $S = 2 \cdot P \cdot R \cdot \sin\left(\frac{\pi}{\varrho}\right)$

b. $k = 2 \cdot R \cdot \sin\left(\frac{\alpha}{2}\right)$

c. $l = 2\sqrt{y^2 + \frac{4x^2}{3}}$

d. $s = -\frac{\cos^4 x}{x}$

* e. $u = -\frac{\cos^{p+1} x}{p+1}$

* f. $m_2 = \frac{1}{2} \ln \frac{1+\sin x}{1-\sin x}$ (piši: EM2)

g. $R = \frac{\sin^3 x \cdot \cos^2 x}{5} + \frac{2}{15}\sin^3 x$

h. $N = \ln|\operatorname{tg} x + \operatorname{ctg} x|$

* i. $e = x \cdot \operatorname{arctg}\frac{x}{a} - \frac{a}{2}\ln(a^2 + x^2)$

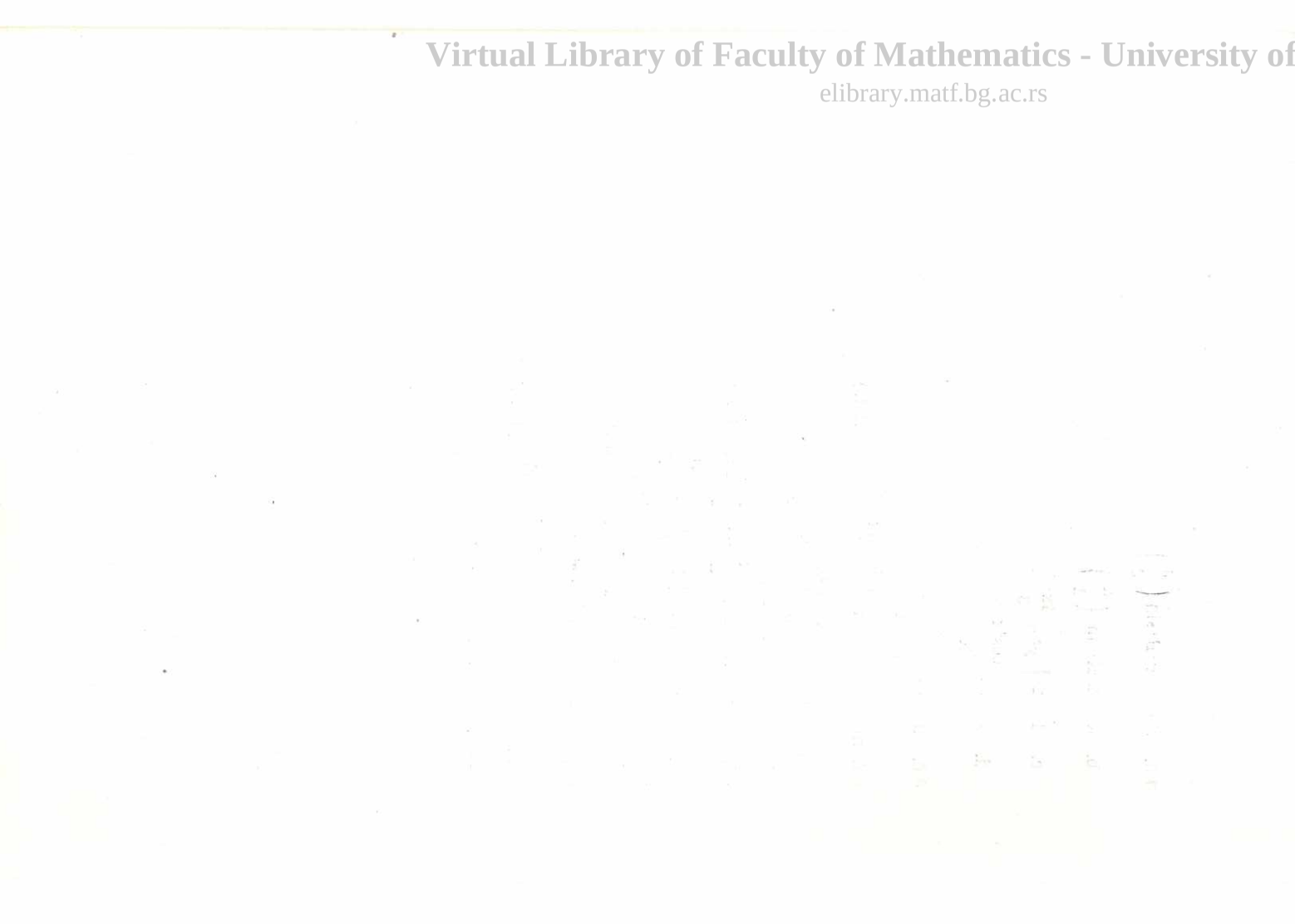
j. $n = -\frac{\pi}{2}\ln|x| + \frac{a}{x} - \frac{a^3}{9x^3}$

k. $W = -\frac{1}{\sqrt{x^2 - a^2}} - \frac{2a^2}{3(\sqrt{x^2 - a^2})^3}$

* l. $q = \left(\frac{2}{\pi x}\right)^{1/2} \cdot \sin x$

m. $j_1 = \frac{e^{x\sqrt{2}} \cos(\sqrt{x/2} + \pi/8)}{\sqrt{2\pi x}}$ (piši: AJ1)

n. $y = (2\pi)^{1/2} x^{x+1} e^{-x}$



Glava 4

ELEMENTARNI OPIS NAREDBI ZA ULAZ I IZLAZ

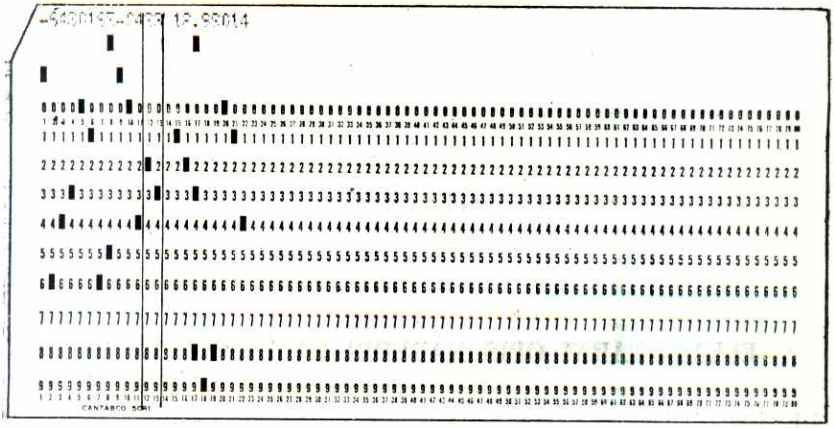
4.1 *PODACI*

Kada se pristupa sastavljanju programa za rešavanje nekog problema na računaru, treba ga zamisliti tako da omogućava rešavanje više zadataka iste vrste. To je princip na kome se zasniva logika svakog programa, bez obzira na to o kakvoj vrsti zadatka se radi.

Od jedne do druge upotrebe programa na računaru, u njemu se dakle moraju menjati *podaci* sa kojima se obavljaju računske i druge operacije. Za FORTRAN-program, podaci su obično nizovi konstanti i te konstante najčešće predstavljaju početne vrednosti određenih promenljivih u programu.

Videli smo da svaka promenljiva zauzima u memoriji računara jedno mesto, koje je programom pripremljeno za prijem konstante određene vrste (celobrojna, realna, dvostruko tačna, kompleksna ili logička konstanta). Izvan memorije računara, podaci u vidu konstanti nalaze se na spoljnim nosiocima, koji kod savremenih računara mogu biti perforirane kartice ili papirne trake, magnetne trake ili magnetni diskovi. Na tim nosiocima svaki podatak zauzima po jedno *polje;* više polja čine veće jedinice — *slogove,* a više slogova formiraju *datoteku* ili skup organizovanih podataka koji logički pripadaju jednom zadatku.

Najmanja jedinica, polje, zauzima na perforiranoj kartici jednu ili više susednih kolona (slika 4.1). Prelazeći na naredbe za ulaz i izlaz, u početku ćemo govoriti samo o ulazu podataka sa bušenih kartica i o izlazu u obliku štampanog teksta. Kasnije ćemo videti da se isti pojmovi ponavljaju i kod drugih nosilaca podataka.



Slika 4.1 Kartica sa podacima

4.2 *NAREDBE READ I FORMAT*

Početne vrednosti za rad programa najčešće se unose u program mašinskim »čitanjem« bušenih kartica. U tom momentu, u memoriji računara se već nalazi naš program, čijim izvršavanjem i dolazi do čitanja ili unošenja podataka. FORTRAN-naredba, koja se prevodi u mašinske instrukcije za čitanje ili ulaz podataka, jeste naredba READ (*read* znači čitati). Naredba READ odnosi se uvek na jedan ceo slog; kada su podaci na karticama, jedan slog čine sva polja jedne kartice.

Naredba READ sadrži u opštem slučaju i jedan niz ili *listu* simbola promenljivih, za koje se uzimaju vrednosti s jedne kartice. Te vrednosti moraju u kartici biti izbušene istim redom kojim su njihovi simboli navedeni u listi naredbe READ. Vidimo, dakle, da postoji sledeća veza između simbolâ i vrednosti na kartici: prvom simbolu odgovara prva vrednost na kartici, drugom simbolu druga vrednost i tako dalje, sve dok ima simbola i novih vrednosti. Unošenje pročitanih vrednosti u memoriju računara, na mesta koja odgovaraju navedenim simbolima promenljivih u naredbi READ, vrši se s leva na desno, počev sa prvim poljem na levoj strani. Osim toga, svakim izvršavanjem naredbe READ pročita se *uvek* samo po jedna nova kartica. Ako je, na primer, u jednu karticu bušeno šest vrednosti (šest polja), tih šest vrednosti se ne mogu uneti u računar pomoću dve naredbe READ, tako što bi druga nastavila onde gde je prva stala. Jedini način da se tih šest brojeva pročitaju sa jedne kartice jeste da u programu bude naredba READ, u čijoj

će listi biti napisani simboli svih šest promenljivih kojima te vrednosti pripadaju.

Ulazni podaci za program mogu biti organizovani u više datoteka, koje se fizički nalaze na svojim nosiocima u raznim uređajima računara. Tako, jedan deo podataka može biti u datoteci u čitaču kartica, druge dve datoteke na jednom disku i sledeće dve datoteke na jednom koturu u uređaju magnetne trake. Zato naredba READ mora da sadrži i podatak o kojoj se datoteci radi. U tu svrhu služi tzv. *pozivna šifra* datoteke u naredbi READ, koja se kod starijih FORTRAN-sistema zvala *simbolički broj* uređaja računara. Ta pozivna šifra može biti jedan pozitivan ceo broj ili odgovarajući simbol promenljive. Za sada ćemo uzeti da su sve datoteke numerisane celim brojevima.

Pisanjem, dakle, jedne naredbe READ, prema do sada izloženom opisu ove naredbe, daju se sledeće informacije o operaciji ulaza podataka:

1. da je to operacija ulaza podataka, zna se po reči READ;
2. da se radi o ulazu podataka sa bušenih kartica, zna se na osnovu pozivne šifre datoteke*;
3. kojim promenljivima u programu se dodeljuju ulazne vrednosti: to je precizirano nabrajanjem njihovih simbola u listi naredbe READ;
4. kojim redom, s leva na desno, su vrednosti bušene u karticu: to je onaj isti redosled po kome su pripadajući simboli citirani u listi naredbe READ.

Za potpuno definisanje operacije ulaza potrebne su još i sledeće informacije: u kom obliku su vrednosti bušene u karticu? Koliko kolona u kartici pripada svakoj pojedinoj vrednosti? Je li pojedina vrednost celobrojna, realna, dvostruko tačna, kompleksna ili logička veličina? Ako je to realan broj, da li je u normalnom obliku ili sa izložiocem (slovo E odnosno D)? Ako u samom broju nije eksciplitno sadržana decimalna tačka, gde se ona podrazumeva?

Sve ove informacije daju se naredbom FORMAT. Detaljnije o naredbi FORMAT govorićemo u Glavi 8, gde ćemo videti sve mogućnosti koje ona pruža u vezi sa ulazom i izlazom. Ovde ćemo, na jednom primeru, razmotriti samo ono što je neophodno za razumevanje naredbe READ. Uzmimo da treba uneti početne vrednosti za tri promenljive, B, J i Y. Te vrednosti su bušene u prva tri

* svaki računski centar povezuje, po specifičnim pravilima svog računara, naše pozivne šifre datoteka sa konkretnim fizičkim uređajima računara

polja kartice na slici 4.1. Čitanje te kartice vrši se na osnovu sledećih dveju naredbi:

```
      READ (5,69) B, J, Y
69 FORMAT (E11.5, I2, F9.0)
```

Broj 69 u četvrtoj i petoj koloni kartice sa naredbom FORMAT (vidi programski obrazac, sl. 3.1) predstavlja *broj naredbe;* taj broj je izabran sasvim proizvoljno. Kao što ćemo videti u Glavi 5, brojevi se naredbama daju po potrebi, da bi se u okviru programa mogla uspostaviti veza jedne naredbe sa drugom. Ovde je broj 69 upotrebljen da se iskaže veza naredbe FORMAT sa naredbom READ, na koju se ona odnosi.

Broj 69 vidimo u zagradama naredbe READ, na drugom mestu. Ispred tog broja je broj 5, koji predstavlja pozivnu šifru datoteke u kojoj se nalazi kartica sa slike 4.1. Datoteka ulaznih podataka na karticama nalazi se, za naš program, u čitaču kartica čiji je simbolički broj 5.

Sada vidimo onu ranije pomenutu vezu između simbola promenljive u listi naredbe READ i njoj pripadajuće vrednosti u ulaznoj kartici: ta veza se ostvaruje preko naredbe FORMAT. Definicija ili *specifikacija* E11.5 u naredbi FORMAT, pošto je prva po redu, automatski se povezuje sa prvim poljem na kartici; FORMAT-specifikacija I2 povezana je sa drugim poljem kartice; analogno važi i za treću specifikaciju i za treće polje.

Ako sada pogledamo na karticu, vidimo da su na njoj vertikalnim linijama ograničena tri polja. Vertikalne linije su povučene samo radi ilustracije, inače one nemaju nikakvog značaja za program. Prvo polje se sastoji od prvih 11 kolona na kartici, što nam pokazuje broj 11 u specifikaciji E11.5. Vrednost u tom polju je realan broj, pisan u obliku sa izložiocem (E-04), što vidimo po slovu E u FORMAT-specifikaciji. Desno od zamišljene decimalne tačke (ona u polju nije bušena) treba podrazumevati pet decimalnih cifara, tj. ovaj broj je ustvari —6.43016E—04 ($= -6.43016 \cdot 10^{-4} = -0.000643016$); to nam pokazuje broj 5 u specifikaciji E11.5. Ukratko, FORMAT-specifikacija E11.5 znači da se vrednost za promenljivu B nalazi u polju ulazne kartice koje ima 11 kolona, da je to realan broj pisan na način sa izložiocem, kao i da su njegovih pet cifara decimale. Ovde vidimo da smo izostavljanjem decimalne tačke pri bušenju dobili mesta za jednu cifru više.

Specifikacija za drugo polje je jednostavnija, pošto se tu radi o celom broju, kod koga nema decimalne tačke. Slovo I znači da se radi o celom broju, a broj 2 da polje zauzima dve kolone. Na

slici 4.1 se vidi da je vrednost bušena bez algebarskog znaka: ako je vrednost pozitivna, znak plus se ne mora bušiti.

Specifikacija trećeg polja, F9.0, je još jedna vrsta specifikacije. Slovo F pokazuje da se radi o realnom broju ali u normalnom obliku, bez izložioca E ili D. Kao što već možemo i sami pretpostaviti, broj 9 znači polje od devet kolona, a 0 da nema nijednog decimalnog mesta. Međutim, ova konvencija o decimalnim mestima važi samo kada decimalna tačka *nije* bušena. U našem primeru, naprotiv, vidimo da u trećem polju kartice *postoji* decimalna tačka; u takvom slučaju, bušena decimalna tačka ima prednost nad specifikacijom, pa će broj koji se unosi u memoriju računara biti realan broj +12.98014, dakle tačno onakav kako je bušen.

Mi smo na ovom primeru videli samo mali deo mogućnosti koje pruža naredba FORMAT ali se već i iz ovog uvoda vidi da FORTRAN daje potpunu slobodu u izboru plana kartice. Kasnije, u Glavi 9, videćemo i druge FORMAT-specifikacije, pomoću kojih se postiže potpuna fleksibilnost u oblikovanju ulaznog i izlaznog sloga.

4.3 *NAREDBE WRITE I FORMAT*

Pomoću naredbe WRITE (*write* znači pisati) mogu se u jednom redu odštampati rezultati računanja, a isto tako i narediti izlaz podataka iz memorije računara na bilo koji izlazni medijum. Naredba WRITE deluje analogno naredbi READ, samo u obrnutom smeru. Posle reči WRITE, u zagradama se piše pozivni broj datoteke koja prima izlazne podatke i broj pripadajuće naredbe FORMAT. Simboli promenljivih, čije vrednosti treba da se štampaju, navode se u vidu jedne liste posle zagrada. Isto kao kod naredbe READ, simboli se u listi međusobno odvajaju zarezima. Kao što je izvršavanje naredbe READ značilo uvek ulaz jednog sloga, tj. jedne kartice, tako se i pri svakom izvršavanju naredbe WRITE, ako se radi o direktnom izlazu na štampač, štampa uvek samo jedan red.

Vrednosti promenljivih biće u jednom štampanom redu poređane istim redom kojim su njihovi simboli navedeni u naredbi WRITE. Oblik tih vrednosti, tj. oblik celog ili realnog broja, sa ili bez izložioca, zavisiće od specifikacija u naredbi FORMAT.

U vezi sa naredbom WRITE, naredba FORMAT služi za određivanje oblika i rasporeda podataka u izlaznom slogu, na primer u štampanom redu. Značenje pojedinih specifikacija je isto kao i ranije, s tim što treba voditi računa da se sada radi o izlazu, a ne o ulazu podataka. Slovo I opet znači da se radi o celobrojnoj

vrednosti, koja prema tome treba da bude odštampana bez decimalne tačke. Slova E i D diktiraju štampanje u obliku realnog broja, sa odgovarajućim izložiocem, dok slovo F znači štampanje realnog broja bez izložioca. Prvi broj iza slova I, E, D ili F uvek određuje broj mesta ili pozicija u polju gde će biti odštampana vrednost, a sledeći broj (izuzev kod specifikacije I, gde drugog broja nema) određuje još i koliko ima mesta iza decimalne tačke. Ako se štampaju po specifikacijama E, D ili F, odštampani brojevi će uvek sadržati i decimalnu tačku.

Pokazaćemo primenu ovih pravila na nekoliko jednostavnih primera. Uzmimo da četiri promenljive, I, J, X i Y, imaju redom sledeće vrednosti: —1234, +32767, —0.0830 i +733.5678. Po naredbama

```
      WRITE (3,40) I, J, X, Y
   40 FORMAT (I10, I10, F10.4, F12.4)
```

te vrednosti će biti odštampane kao u prvom redu na slici 4.2. Vidimo da su pozitivni brojevi odštampani bez znaka plus, što je uvek slučaj kada se radi o izlazu na štampač. Prva tri polja zauzimaju po deset pozicija, a četvrto dvanaest; poslednja dva imaju po četiri decimalna mesta. U svim poljima brojevi zauzimaju mesta do desne granice polja, dok su preostale pozicije do leve granice ispunjene znacima blanko.

```
-1234      32767    -0.0830      733.5678
-1234      32767  -0.8300000E-01    0.7335678E 03
-1234      32767   -8.300000E-02     7.335678E 02
```

Slika 4.2 Štampani redovi

FORMAT-specifikacija F u vezi sa naredbom WRITE može se upotrebiti uvek kada su nam poznate maksimalne i minimalne veličine brojeva koji će se štampati. Ako ne možemo pretpostaviti veličinu, tada je bolje da upotrebimo specifikaciju E (ili D), pa će vrednost pre štampanja biti pretvorena u oblik realnog broja sa izložiocem. Na primer, ako za iste vrednosti promenljivih kao u prethodnom primeru upotrebimo naredbe

```
      WRITE (3,125) I, J, X, Y
  125 FORMAT (I10, I10, E16.7, E16.7)
```

tada će vrednosti biti odštampane kao u drugom redu na slici 4.2. I kod mantise i kod izložioca u karakteristici, ako su to pozitivne

veličine, ne štampa se znak plus. Apsolutna vrednost mantise će uvek biti između 0 i 1, dakle u internom ili normalizovanom obliku pokretnog zareza.

Ako želimo da brojevi budu odštampani tako da im apsolutne vrednosti mantisa budu između 1 i 10, tada ćemo za iste vrednosti promenljivih pisati naredbe

```
      WRITE (3,1547) I, J, X, Y
1547  FORMAT (I10, I10, 1PE16.6, 1PE16.6)
```

pa ćemo dobiti oblik štampanog reda kao što je pokazano u trećem redu na slici 4.2. Faktor 1 sa slovom P ispred specifikacije E (ili D) zove se *faktor razmere* i njegovo dejstvo vidimo kada na slici 4.2 uporedimo odštampane vrednosti realnih brojeva u drugom i trećem redu; o njegovoj upotrebi detaljnije se govori u Glavi 9.

Kada u naredbi FORMAT imamo uzastopno nekoliko istih specifikacija, tada specifikacije možemo prostije pisati upotrebljavajući tzv. *faktor ponavljanja*. Naredbe iz prethodna dva primera mogli smo, znači, pisati

```
      WRITE (3,125) I, J, X, Y
125   FORMAT (2I10, 2E16.7)
```

odnosno

```
      WRITE (3,1547) I, J, X, Y
1547  FORMAT (2I10, 1P2E16.6)
```

Oblici štampanih redova će biti potpuno isti kao i bez faktora ponavljanja, sa pojedinačno napisanim specifikacijama.

4.4 *JEDAN PROGRAM*

Sa pravilima koja smo do sada naučili, moguće je već pisati jednostavnije FORTRAN-programe. Da se pri tome ne bismo suviše zadržavali oko odredivanja oblika ulaznih i izlaznih podataka, pridržavaćemo se u ovom i sličnim primerima izvesne jednoobraznosti, koja se sastoji u sledećem. Za sve vrednosti koje se nalaze na karticama i koje će služiti kao ulazni podaci, usvojićemo jedinstvenu dužinu polja od 10 kolona. To znači da će za celobrojne konstante važiti specifikacija I10. Za realne konstante je najbolje usvojiti specifikaciju F10.0, a u karticama bušiti njihove vrednosti sa decimalnom tačkom; broj decimalnih mesta određivaće u tom slučaju sama bušena decimalna tačka, bez obzira što u specifikaciji

stoji da je broj decimala nula. Podaci će veoma retko biti toliko veliki ili toliko mali brojevi da bi ovoliko polje bilo nedovoljno; ako se ipak računa sa takvim slučajevima, tada treba usvojiti neku drugu jedinstvenu specifikaciju. Za izlaz celobrojnih vrednosti biće takođe dovoljno ako se polje specificira sa I10. Vrednosti koje izlaze u obliku realnih brojeva treba štampati sa specifikacijom E20.7. Tolika dužina polja daće dovoljan razmak između susednih brojeva, jer smo u primerima na slici 4.2 videli da realni brojevi sa izložiocem E zauzimaju maksimalno četrnaest mesta*.

Pretpostavimo da treba da se izračuna vrednost promenljive R po niže datom obrascu:

$$R = 16.78 \cos 2\pi x + (y + 1.667)^I$$

pri čemu se vrednosti za promenljive X, Y i I čitaju sa kartice na slici 4.3.

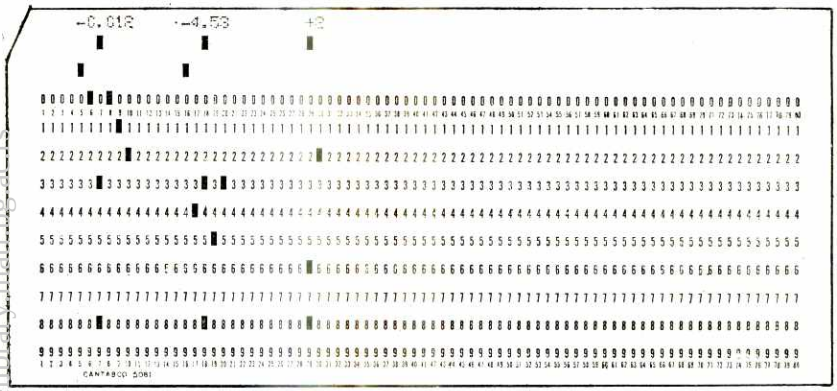
Ovo je, razume se, problem za čije rešavanje nije potrebno koristiti računar; on je ovde uzet samo radi ilustracije naredbi za ulaz i izlaz u jednom FORTRAN-programu. Dalje u tekstu vidimo kako bi se mogao napisati program za ovaj zadatak, program koji će pročitati ulazne podatke, izvršiti računanje i odštampati rezultat tog računanja.

```
      READ (3,10) X, Y, I
      R = 16.78*COS(6.2832*X) + (Y + 1.667)**I
      WRITE (5,20) X, Y, I, R
   10 FORMAT (2F10.0, I10)
   20 FORMAT (2E20.7, I10, E20.7)
```

Naredbom READ zahteva se čitanje jedne kartice, koja se nalazi u datoteci sa pozivnim brojem 3 (čitač kartica). U karticu su bušene vrednosti za tri promenljive, X, Y i I; ta kartica je prikazana na slici 4.3, gde vidimo da su te vrednosti X = —0.012, Y = —4.53 i I = +2.

Specifikacije u naredbama FORMAT usvojene su prema uprošćenom metodu, o kome je napred bilo reči. Same naredbe FORMAT napisane su u ovom programu na kraju i povodom toga treba reći da se one u svakom programu mogu pisati bilo gde, tj. nije obavezno da se pišu neposredno iza naredbe za ulaz odnosno izlaz, sa kojima su u vezi. To je i logično jer su to naredbe koje se ne izvršavaju nego samo daju FORTRAN-prevodiocu informaciju o

* kod računara gde mantisa običnog realnog broja ima 7 značajnih cifara



Slika 4.3 Kartica sa podacima

obliku ulaznih i izlaznih podataka. Takvih naredbi, koje imaju samo informativan karakter, ima još i o njima će biti reči kasnije. Ovde još treba zapaziti da je u naredbama FORMAT korišćen faktor ponavljanja za specifikacije susednih vrednosti u slogovima, pošto su istog oblika.

Po naredbi WRITE izlaze u datoteku sa pozivnim brojem 5 (štampač) ulazne vrednosti, a zajedno sa njima u istom slogu (redu) i rezultat računanja. Ovakav način se preporučuje jer inače, ako se sa istim programom reši više zadataka, neće se uvek lako znati kojim ulaznim podacima odgovara koji rezultat. Na slici 4.4 prikazan je štampani red, dobijen na osnovu naredbe WRITE u ovom primeru.

```
-0.1200000E-01     -0.4530000E 01     2     0.2492980E 02
```

Slika 4.4 Rezultat programa iz odeljka 4.4

Ovakav program nije još uvek potpun FORTRAN-program i to iz dva razloga. Posle ulaza podataka, izračunavanja i izlaza rezultata, program bi trebalo da se automatski vrati na početak i da ponavlja programirani postupak sve dok ima podataka u ulaznoj datoteci; kako se to postiže videćemo u Glavi 5. Osim toga, nedostaje informacija FORTRAN-prevodiocu o tome gde je kraj ovog programa, da bi se prevođenje moglo završiti. To je funkcija naredbe END, jedne od naredbi za upravljanje, o kojima ćemo govoriti

u Glavi 5. Sada ćemo, međutim, rezimirati ceo postupak kojim se od izvornog programa dolazi do rešenja zadatka na računaru.

4.5 *PREVOĐENJE I IZVRŠAVANJE FORTRAN-PROGRAMA*

Onaj ko želi da programira rešavanje svog zadatka na FORTRANu, da bi zatim program i podatke za računanje predao na realizaciju nekom računskom centru, mora da ima jasnu predstavu o daljim fazama rada u centru, od prijema izvornog programa pa do dobijanja rezultata. Naročito je važno da se jasno shvati kako se dolazi do faze u kojoj naš program može da primi podatke sa kojima će vršiti računanje. Ne ulazeći ponovo u prethodni deo rada na stvaranju programa, opisan u Glavi 1, ovde ćemo poći od trenutka kada je naš FORTRAN-program napisan na obrascima za programiranje. Dalji postupak je sledeći:

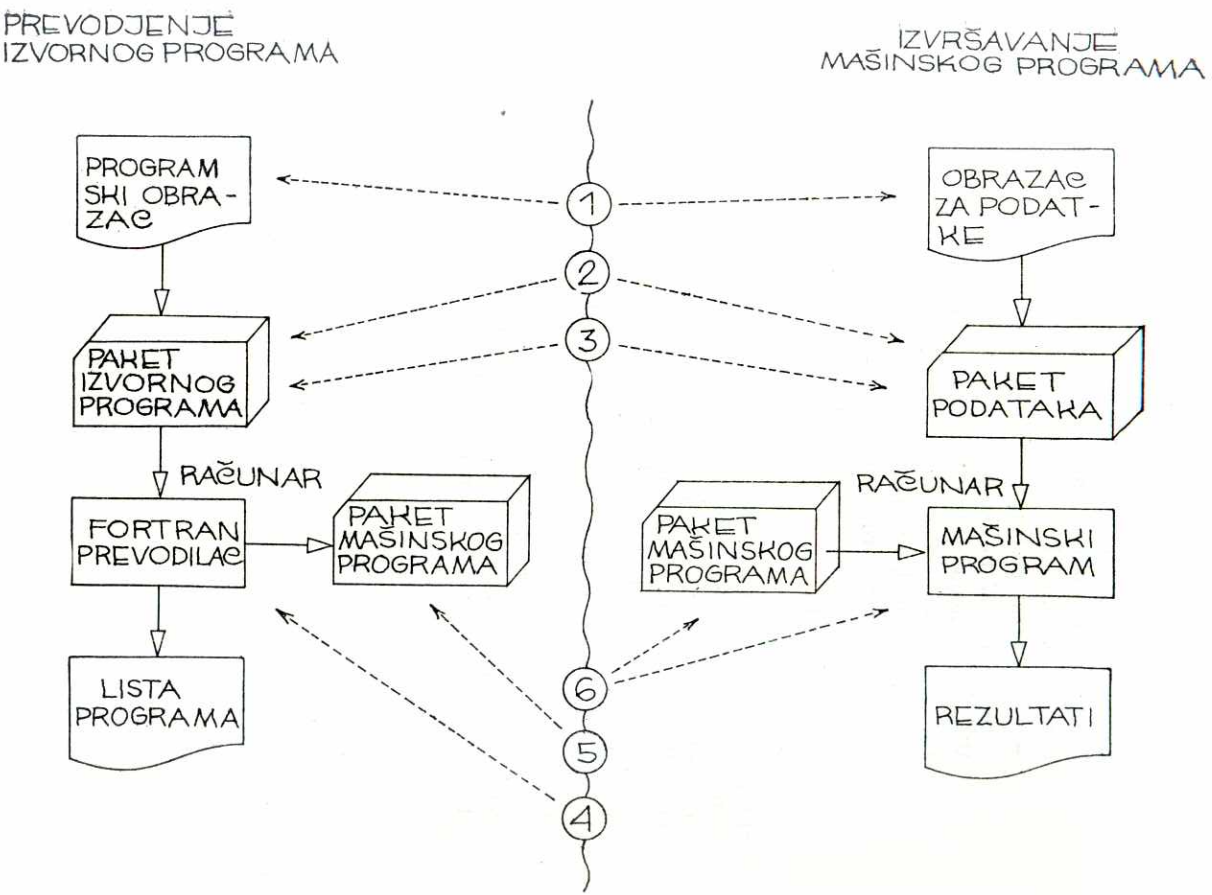
1. Računski centar prima program, napisan na obrascima kao što su oni na slici 3.1. To je izvorni program, u kojem se, pored ostalih, obično nalazi i više naredbi READ i WRITE, pomoću kojih će se unositi podaci i štampati rezultati.
2. Program se buši u kartice. Proizvod bušenja je *paket kartica izvornog programa*. Kartice sa ulaznim podacima *ne* nalaze se u tom paketu. U nastavku se zatim buše kartice koje sadrže naše podatke za ulaz u računar i od njih se formira poseban *paket ulaznih kartica*. Podatke za ove kartice predajemo centru na posebnim obrascima, različitim od onih za izvorni program.
3. Zavisno od organizacije rada u centru, naš program i ulazni podaci se prenose na druge medijume (magnetnu traku ili magnetni disk) ili se u vidu kartica stavljaju u čitač. Pri tome je važno da se na ulaz računara stavi naš izvorni program, dok ulazni podaci za sada još nisu potrebni.
4. U memoriju računara unosi se FORTRAN-prevodilac, obično sa diska ili magnetne trake, gde se stalno nalazi.
5. Izvršavanjem programa-prevodioca počinje čitanje kartica našeg izvornog programa. Kartice FORTRAN-programa sada su ustvari nosioci podataka potrebnih prevodiocu za njegov rad. Mašinske instrukcije, dobijene prevođenjem izvornog programa na interni jezik računara, izlaze na neki spoljni medijum (disk ili traku), eventualno i na nove kartice koje se nalaze u bušaču. Tako se dobija *paket kartica mašinskog programa* ili mašinski program zapisan na magnetnoj traci

ili disku. Pri tome, *mašinski program se još ne izvršava niti se još čitaju naši ulazni podaci.*

6. Umesto programa-prevodioca ili kasnije, posle nekog drugog radnog programa, u memoriju računara se unosi naš *mašinski program.* Pre početka rada tog programa, stavljaju se naši ulazni podaci i ostale potrebne datoteke u one jedinice računara, na koje se pozivaju naše naredbe READ i WRITE pomoću pozivnih šifara datoteka koje smo pisali u tim naredbama. Tek kada počne rad ovog programa *dolazi do čitanja naših ulaznih podataka* i obavljanja izlaznih operacija koje smo programirali u izvornom programu. Tada se u računaru odvija uopšte cela logika našeg programa, od početka do kraja.

Neobično je važno da se ove faze stvaranja jednog FORTRAN-programa imaju stalno u vidu. Ovo je pogotovu potrebno zato što ćemo se i nadalje kao i do sada služiti skraćenim načinom izražavanja, da bi izlaganje bilo jasnije. Kada kažemo »izvršavanje naredbe WRITE«, tu se uvek misli na rad onih mašinskih instrukcija u koje je prevedena naredba WRITE. Isto tako, »po naredbi READ čitaju se ulazni podaci« ne treba shvatiti doslovno, nego zamisliti sve faze koje su već obavljene od pisanja izvorne naredbe READ pa do stvarnog izvršavanja instrukcija za ulaz podataka u memoriju računara. Između prevođenja programa (faza 5) i njegove upotrebe u računaru za rešavanje konkretnog zadatka (faza 6) može biti vremenski interval od nekoliko minuta do nekoliko godina. To je i razumljivo kada se zna da se prevođenje obično obavi dva do tri puta (dok se ne otkriju sve greške u logici izvornog programa), a gotov program se često posle koristi i više stotina puta.

Na slici 4.5 šematski je prikazan ceo postupak, koji je napred opisan pq fazama 1 do 6. Na slici nisu, radi preglednosti, naznačeni mogući drugi oblici u kojima se javljaju izvorni program, mašinski program ili ulazni podaci; te oblike pomenuli smo u opisu postupka. Brojevi na dijagramu predstavljaju opisane faze.



Slika 4.5 Prevođenje i izvršavanje FORTRAN-programa

VEŽBE

U svim sledećim zadacima traži se da se napiše program po ugledu na onaj u odeljku 4.4. Ulazni podaci nalaze se uvek na jednoj kartici i pošto su svi realne veličine, bez izložioca, i svi zauzimaju polja dužine 10 kolona, treba sa naredbom za ulaz primeniti specifikaciju F10.0. Izlazne veličine treba štampati po specifikaciji E20.7.

1. Ulaz: a, b, c
   Izlaz: a, b, c, d
   Izračunati:

$$d = \frac{1 + a}{1 + \frac{b}{c + 8}}$$

* 2. Ulaz: a, b, c
   Izlaz: a, b, c, s, P
   Izračunati:

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

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

3. Ulaz: s, x
   Izlaz: s, x, g
   Izračunati:

$$g = (12.7 - x)^{s+2}$$

4. Ulaz: x, y
   Izlaz: x, y, h
   Izračunati:

$$h = \frac{x \cdot \cos^4 x}{2y}$$

* 5. Ulaz: a, b, c
   Izlaz: a, b, c, x_1, x_2
   Izračunati:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

6. Ulaz: D, d
Izlaz: D, d, J, W
Izračunati:

$$J = \frac{\pi}{64}(D^4 - d^4)$$

$$W = \frac{\pi}{32} \cdot \frac{D^4 - d^4}{D}$$

7. Ulaz: a, b, c, x
Izlaz: a, b, c, x, r
Izračunati:

$$r = \frac{b \cdot c}{12}\left[6x^2\left(1 - \frac{x}{a}\right) + b^2\left(1 - \frac{x}{a}\right)^3\right]$$

8. Ulaz: a, b, g_1, g_2, h_1, h_2
Izlaz: a, b, g_1, g_2, h_1, h_2, v
Izračunati:

$$v = \frac{h_2 \cdot g_1 \cdot g_2}{a \cdot b} \operatorname{arctg} \frac{a \cdot b}{h_1\sqrt{h_1^2 + a^2 + b^2}}$$

* 9. Ulaz: a, x, s
Izlaz: a, x, s, y, z
Izračunati:

$$y = \sqrt{x^2 - a^2}$$

$$z = \frac{x \cdot s}{2} - \frac{a^2}{2} \ln|x + s|$$

10. Ulaz: f, t, c, h, r
Izlaz: f, t, c, h, r, a, s
Izračunati:

$$a = \operatorname{arctg} \frac{2\pi \cdot f \cdot h - \frac{1}{2\pi \cdot f \cdot c}}{r}$$

$$s = \sin(2\pi \cdot f \cdot t - a)$$

Glava 5

NAREDBE ZA UPRAVLJANJE

5.1 BROJ NAREDBE

U svim dosadašnjim primerima i vežbama videli smo da se FORTRAN-naredbe izvršavaju onim redom kako su u programu napisane. Nama je, međutim, često potrebno da program ne »teče« tim utvrđenim tokom nego da, zavisno od nekog uslova ili bezuslovno, pređe na bilo koju drugu naredbu umesto na sledeću po redu. Takvu potrebu smo videli na primeru jednostavnog programa u odeljku 4.4, gde smo rekli da bi posle završenog ciklusa računanja program trebalo da se automatski vrati na početak i da ponavlja programirani postupak sve dok ima podataka u ulaznoj datoteci.

Da bismo takvu potrebu u programu mogli realizovati, naredba na koju želimo da program u svom radu pređe mora biti na neki način obeležena. U tu svrhu služe *brojevi naredbi* koje možemo ali i ne moramo, dati svakoj naredbi. Jednu takvu vezu dveju naredbi pomoću broja videli smo kod parova READ i FORMAT odnosno WRITE i FORMAT, samo što se tu nije radilo o prenošenju akcije na drugu naredbu jer naredba FORMAT ne vrši nikakvu akciju.

Broj naredbe može biti bilo koji pozitivan ceo broj, manji od 99999. (Za FORTRAN-prevodioce nekih tipova računara on mora biti manji od 32768). Taj broj se upisuje u obrascu u polje od 1. do 5. kolone dotične naredbe i u tom polju on se može nalaziti bilo gde. Ne mora se paziti na redosled tih brojeva u okviru jednog programa, samo dve ili više naredbi jednog programa ne smeju nositi isti broj. Da ovo bolje istaknemo, evo jednog dozvoljenog niza brojeva za naredbe: 300, 9, 9023, 9022, 9024, 180 i 213. Između naredbi koje nose ove brojeve, mogu se nalaziti i naredbe bez brojeva.

Ukratko, brojevi naredbi omogućuju da jedna naredba poziva drugu ili prenosi akciju na drugu naredbu.

5.2 *NAREDBA GO TO*

Naredba GO TO ima tri oblika, koji se međusobno logički razlikuju i koji se zato u programima različito upotrebljavaju. Ovde ćemo govoriti o tzv. bezuslovnoj naredbi GO TO, čiji je opšti oblik:

GO TO n

Dejstvo ove naredbe je da se akcija prenosi na naredbu sa brojem n u istom programu. Da je ona izostavljena, rad programa bi tekao normalnim redosledom od naredbe do naredbe, onako kako su složene kartice sa naredbama u paketu izvornog programa. Naredba sa brojem n na koju se akcija prenosi, mora da bude jedna od naredbi koje izražavaju neku akciju (ne sme biti npr. FORMAT, REAL, COMPLEX i tsl.), a može se u programu nalaziti ispred ili iza naredbe GO TO n.

Vrlo česta primena bezuslovne naredbe GO TO jeste kad želimo da program počne rad ispočetka, sa novim početnim podacima, pošto je već dobijen rezultat računanja sa prethodnom grupom podataka. Ovakvu primenu ćemo ilustrovati jednim primerom, u kome ćemo ujedno videti bolje korišćenje mogućnosti računara od onoga što smo imali u jednostavnom programu u odeljku 4.4.

Pretpostavimo da treba da se izračuna jačina naizmenične struje u kolu koje sadrži elemente otpornosti, kapacitivnosti i induktivnosti. Izraz za jačinu struje, I, je:

$$I = \frac{E}{\sqrt{R^2 + \left(2\pi fL - \frac{1}{2\pi fC}\right)^2}}$$

gde je I u amperima, E napon u voltima, R otpor u omima, L induktivnost u henrima, C kapacitet u faradima i f frekvencija u hercima.

Svrha programa je da se dobiju podaci za crtanje dijagrama, koji će pokazivati zavisnost struje od frekvencije pri datim vrednostima E, R, L i C. Zato ćemo program napisati tako da najpre pročita sa jedne kartice vrednosti za sve te veličine, a zatim da redom čita sa niza kartica vrednosti za frekvenciju. Kada izračuna jačinu struje za datu frekvenciju, program treba da štampa vrednost frekvencije i izračunatu jačinu struje, da pročita novu vred-

nost frekvencije, da izračuna novu jačinu struje itd. (Kasnije ćemo videti da se isti problem može rešiti i bez čitanja svih vrednosti frekvencije sa kartica, ako su željene vrednosti frekvencije raspoređene po nekom poznatom zakonu, npr. ekvidistantne).

Držeći se načela da simbole promenljivih biramo tako da budu što bliži izvornim oznakama, a istovremeno da po mogućstvu budu saglasni sa unutrašnjom konvencijom FORTRANa, vidimo da za dve veličine, I i L, moramo potražiti nove simbole. Pošto iz poznatih razloga želimo da sve veličine budu realne (dvostruka tačnost nam ovde nije potrebna), usvojićemo za jačinu struje simbol A (amper), a za induktivnost H (henri), dakle početna slova naziva jedinica. Drugo rešenje bi bilo da jednom naredbom na početku programa deklarišemo promenljive I i L kao realne, tj. da napišemo REAL I, L. Napišimo sada program prema postavljenom zadatku:

```
      READ (4,200) E, R, H, C
  200 FORMAT (4F10.0)
   68 READ (4,200) F
      A = E/SQRT (R**2 + (6.2832*F*H — 1./(6.2832*F*C))**2)
      WRITE (7,300) F, A
  300 FORMAT (2E20.7)
      GO TO 68
      END
```

Prvom naredbom READ unosimo početne vrednosti za četiri promenljive, koje se ustvari neće više menjati u toku rada programa. Druga naredba READ čita sledeću karticu i sa nje se dodeljuje prva vrednost za promenljivu F. Ovoj naredbi smo dali broj da bi program kasnije automatski tražio novu karticu, sa novom vrednošću za F; vrednosti za E, R, H i C čitaju se prema tome samo jedanput u toku celog programa. Zatim dolazi aritmetička naredba, kojom je programiran glavni cilj zadatka. Posle štampanja vrednosti F i A, izvršiće se naredba GO TO 68, koja zahteva čitanje sledeće kartice sa vrednošću za F i tako započinje ponavljanje celog postupka. Ponavljanje će se vršiti sve dotle dok postoje novi slogovi u datoteci sa pozivnim brojem 4. Kada više ne bude bilo slogova u toj datoteci, računar neće moći da izvrši operaciju čitanja i daće o tome poruku operatoru preko posebne izlazne jedinice. (Ovakav način zaustavljanja rada programa ne preporučuje se kao redovna praksa; time se može izgubiti mnogo dragocenog vremena računara. Na kraju ove glave opisane su FORTRAN-naredbe sa kojima se operatoru računara nedvosmisleno signalizira da je program došao

do svog normalnog završetka. Pre pisanja tih naredbi, treba saznati i kakav je standardni postupak koji se praktikuje u konkretnom računskom centru. Ovde je ovakav način usvojen samo zato što je programerski jednostavniji, pa je za ovu svrhu opravdan).

5.3 *ARITMETIČKA NAREDBA IF*

Naredba GO TO omogućava nam da eksplicitno ili *bezuslovno* menjamo redosled izvršavanja naredbi u jednom FORTRAN-programu. Tu naredbu smo nazvali bezuslovnom upravo zato što ona prenosi akciju na drugu naredbu u programu ne uslovljavajući to ni stanjem podataka ni stanjem računara niti bilo čim drugim. Mada se ta naredba veoma mnogo koristi u programima, nama je takođe potreban i neki način da menjamo redosled izvršavanja naredbi *u zavisnosti od toga šta se desilo u toku rada programa* ili, opštije rečeno, *uslovno*. Drugim rečima, potreban nam je način za uslovno menjanje redosleda izvršavanja, pri čemu bi tu uslovnost određivali prethodno uneti podaci ili prethodno izračunate veličine. Takvu mogućnost pruža nam aritmetička naredba IF.

Opšti oblik aritmetičke naredbe IF je

$$\text{IF (a) } n_1, n_2, n_3$$

gde je a bilo kakav aritmetički izraz, a n_1, n_2 i n_3 su brojevi drugih naredbi u istom programu. Dejstvo ove naredbe je sledeće: ako je aritmetički izračunata vrednost izraza a manja od nule, kao sledeća će se izvršiti naredba sa brojem n_1; ako je vrednost tog izraza nula, sledeća će biti naredba sa brojem n_2; ako je izraz pozitivan, akcija se prenosi na naredbu sa brojem n_3.

Kao jednostavan primer za praktičnu upotrebu aritmetičke naredbe IF, rasmotrimo problem izračunavanja vrednosti jednog određenog integrala. Iz matematičke analize je poznato da vrednost tog integrala, koju ćemo označiti sa v, zavisi od toga da li je konstanta a negativna, nula ili pozitivna.

$$v = \int_0^\infty \frac{a}{a^2 + x^2}\, dx = \begin{cases} -\pi/2 & \text{za } a < 0 \\ 0 & \text{za } a = 0 \\ +\pi/2 & \text{za } a > 0 \end{cases}$$

Neka su nam iz prethodnog dela programa već poznate vrednosti za konstante a i π. Tada možemo napisati deo programa kojim se izračunava vrednost integrala:

```
      IF (A) 7, 8, 9
    7 V = —PI/2.0
      GO TO 6
    8 V = 0.0
      GO TO 6
    9 V = +PI/2.0
    6 . . . . . .
```

Naredba sa brojem 6 je sledeća po redu naredba u tom programu, koja po prirodi zadatka sledi posle izračunavanja vrednosti integrala.

Naredbom IF naređuje se izračunavanje vrednosti izraza a, koji je u ovom slučaju jedna konstanta (podsetimo se da i jedna sama konstanta ili promenljiva može, po definiciji, da čini aritmetički izraz u FORTRANu). Ako je a negativno, kao sledeća će se izvršiti naredba 7, pa će v dobiti vrednost $-\pi/2$. Posle toga nailazi naredba GO TO 6. Kada na ovom mestu ne bi bilo te naredbe, program bi normalno prešao na naredbu 8, čime bi v dobilo pogrešnu vrednost nula. Drugim rečima, naredbom GO TO 6 zaobilaze se ili se »preskaču« sve ostale naredbe ispred one sa brojem 6. Ako se pak naredbom IF utvrdi da je a nula, akcija prelazi na naredbu 8, gde se v izjednačuje sa nulom, pa se po sledećoj naredbi GO TO 6 opet zaobilazi ostali deo kao i ranije. Ako je a pozitivno, posle IF se direktno ide na naredbu 9, gde v dobija vrednost $\pi/2$, i zatim normalno nailazi naredba 6.

U drugom primeru za primenu aritmetičke naredbe IF uzmimo da se traži y kao funkcija od x, po sledeća dva obrasca:

$$y = 0.7\,x + 0.62 \quad \text{ako je } x \leqslant 1.7$$
$$y = 0.9\,x + 0.41 \quad \text{ako je } x > 1.7$$

Niže je dat deo programa po kome se vrši izračunavanje vrednosti funkcije. Naredba IF tu diktira prelaz na osnovu vrednosti izraza (x — 1.7).

```
      IF (X — 1.7) 32, 32, 8
   32 Y = 0.7*X + 0.62
      GO TO 50
    8 Y = 0.9*X + 0.41
   50 . . . . . .
```

Ako je (x — 1.7) negativno, tada je x očigledno manje od 1.7 i program prelazi na naredbu 32, gde se vrednost funkcije računa po obrascu koji važi za taj slučaj. Ako je (x — 1.7) = 0, tada je

$x = 1.7$, pa opet dolazimo na isti obrazac, kao što se i traži u postavljenom zadatku. Ako je (x — 1.7) pozitivno, to znači da je x veće od 1.7 i program prelazi na naredbu 8, gde će vrednost funkcije biti izračunata po odgovarajućem obrascu. Od naredbe sa brojem 50 teče ostali deo programa, u kome se koristi vrednost y, izračunata po jednom od prethodnih obrazaca.

U primerima za aritmetičku naredbu IF, koje smo videli, mogli smo zapaziti da ta naredba kao neka skretnica skreće tok programa u najviše tri, a najmanje dva moguća pravca. Dalje ćemo videti još dve naredbe koje imaju slično dejstvo, samo što je broj pravaca veći.

5.4 *NAREDBA GO TO SA IZRAČUNATIM PRELAZOM*

Kod aritmetičke naredbe IF imali smo skretanje toka programa u tri moguća pravca, na osnovi ispitivanja vrednosti jednog aritmetičkog izraza. Kod naredbe GO TO sa izračunatim prelazom broj mogućih pravaca skretanja je veći, a izbor zavisi od trenutne vrednosti jedne celobrojne promenljive. Opšti oblik ove naredbe je:

GO TO (n_1, n_2, ..., n_m), i

gde je i jedna celobrojna promenljiva sa vrednošću $1 \leqslant i \leqslant m$, pisana bez algebarskog znaka, a n_1, n_2, ..., n_m su brojevi nekih drugih naredbi u istom FORTRAN-programu. Ako je u trenutku izvršavanja ove naredbe vrednost promenljive i jednaka j, doći će do prelaza na naredbu sa brojem n_j. Na primer, neka naredba u konkretnom slučaju glasi

GO TO (227, 3, 7, 678, 2), IKS

Ako je u momentu izvršavanja ove naredbe vrednost promenljive IKS jednaka 1, program će preći na naredbu sa brojem 227; ako je IKS = 2, doći će do prelaza na naredbu sa brojem 3; ako je IKS = 3, izvršiće se prelaz na naredbu sa brojem 7 itd. Međutim, ako je vrednost promenljive IKS izvan dozvoljenog opsega, program će normalno nastaviti sa sledećom naredbom po redu.

Razmotrimo sada, primera radi, jedan slučaj koji se u programu može rešiti primenom naredbe GO TO sa izračunatim prelazom. Na nekom mestu u programu potrebna nam je vrednost jednog od prvih pet Ležandrovih (Legendre) polinoma:

$$P_0(x) = 1$$
$$P_1(x) = x$$
$$P_2(x) = \frac{3}{2}\,x^2 - \frac{1}{2}$$
$$P_3(x) = \frac{5}{2}\,x^3 - \frac{3}{2}\,x$$
$$P_4(x) = \frac{35}{8}\,x^4 - \frac{15}{4}\,x^2 + \frac{3}{8}$$

Pretpostavimo da je vrednost promenljive x određena u prethodnom delu programa, kao i da je za celobrojnu promenljivu NUM računski dobijena jedna od vrednosti 0 do 4. Promenljiva NUM će svojom vrednošću odrediti koji od pet Ležandrovih polinoma treba da bude izračunat: ako je NUM = 0, izračunaće se $P_0(x)$, ako je NUM = 1, izračunaće se $P_1(x)$ itd. Taj deo programa možemo napisati ovako:

```
      NUM = NUM + 1
      GO TO (11, 22, 33, 44, 55), NUM
   11 P = 1.0
      GO TO 66
   22 P = X
      GO TO 66
   33 P = 1.5*X**2 — 0.5
      GO TO 66
   44 P = 2.5*X**3 — 1.5*X
      GO TO 66
   55 P = 4.375*X**4 — 3.75*X**2 + 0.375
   66 . . . . . .
```

Pošto je u prethodnom delu programa promenljiva NUM mogla dobiti i vrednost 0, što definicijom naredbe GO TO sa izračunatim prelazom nije dozvoljeno, to naredbom NUM = NUM + 1 najpre pretvaramo konstante 0 do 4 u konstante 1 do 5, posle čega program nastavlja sa naredbom GO TO (11, 22, 33, 44, 55), NUM.

5.5 *NAREDBA ASSIGN I NAREDBA GO TO SA ZADATIM PRELAZOM*

Kod FORTRAN-prevodilaca koji odgovaraju standardu za potpuni FORTRAN IV postoji još jedan oblik naredbe GO TO, tzv. naredba GO TO sa zadatim prelazom. Za razliku od prethodne, gde se vrednost celobrojne promenljive dobija aritmetičkom naredbom, kod naredbe GO TO sa zadatim prelazom ta vrednost se *zadaje* ili dodeljuje jednom prethodnom naredbom ASSIGN (*assign* znači dodeliti). Druga razlika je u značenju koje imaju te dve vrednosti, izračunata i zadata. Izračunati ceo broj ukazuje na *mesto* koje broj naredbe zauzima u zagradama, počev od leve zagrade odgovarajuće naredbe GO TO sa izračunatim prelazom. Zadata celobrojna konstanta kod naredbe GO TO sa zadatim prelazom predstavlja *broj* naredbe na koju treba izvršiti prelaz, pri čemu to mora biti jedan od brojeva navedenih u zagradama.

Opšti oblik ovih dveju naredbi je

```
ASSIGN i TO n
        .
        .
GO TO n, (n1, n2, . . ., nm)
```

U vezi s jednom naredbom GO TO sa zadatim prelazom, normalno se upotrebljava više naredbi ASSIGN. Pogledajmo jedan primer:

```
    B = . . .
1   A = . . .
    ASSIGN 3 TO J
          .
          .
    IF (A — B) 5, 5, 6
5   ASSIGN 1 TO J
          .
          .
6   GO TO J, (1, 3)
3   . . . . . .
```

Kada se u ovom programu bude izvršavala naredba 6, dalji tok programa će zavisiti od toga koja od dve naredbe ASSIGN je bila *poslednja* izvršena, a to opet zavisi od trenutnog odnosa veličina A i B.

Treba još napomenuti da ASSIGN 3 TO J nije isto što i J = 3, drugim rečima celobrojnoj promenljivoj *n* u naredbi GO TO sa zadatim prelazom vrednost se može dati *samo* pomoću naredbe ASSIGN.

Kod svih naredbi kojima se menja redosled izvršavanja naredbi u FORTRAN-programu važi jedno opšte pravilo, koje smo do sada pomenuli samo za bezuslovnu naredbu GO TO (vidi 5.2): naredba *na koju se prenosi* akcija mora da bude jedna od naredbi koje izražavaju neku akciju (ne sme biti npr. FORMAT, REAL, COMPLEX i tsl.), a u programu se može nalaziti ispred ili iza naredbe *koja prenosi* akciju.

Videli smo do sada četiri naredbe pomoću kojih se u programu obavljaju tzv. prelazi, tj. prenošenje akcije sa jedne na bilo koju drugu naredbu. To su bile: bezuslovna naredba GO TO, aritmetička naredba IF, naredba GO TO sa izračunatim prelazom i naredba GO TO sa zadatim prelazom. Još jedna naredba sa istom funkcijom, logička naredba IF, biće opisana u Glavi 6 »Logički izrazi i naredbe« jer ta cela oblast jezika FORTRAN postoji samo u standardu za potpuni FORTRAN IV.

5.6 *NAREDBE PAUSE, STOP I END*

Naredba PAUSE (zastoj) upotrebljava se u programu na onim mestima gde je logički potrebno da se privremeno obustavi rad programa, radi neke intervencije operatora. Opšti oblik te naredbe je:

PAUSE n

gde *n* može biti znak blanko ili celobrojna konstanta, dužine 1 do 5 dekadnih cifara. Kada se u programu izvrši naredba PAUSE, na izlaznoj jedinici operatora (pisaćoj mašini ili dr.) pojaviće se poruka PAUSE 00000, ako u naredbi nije navedena nikakva konstanta, odnosno poruke PAUSE sa odgovarajućim brojem n, ako je taj broj-konstanta napisan u samoj naredbi. Ove brojeve treba pisati ako u progamu postoji više naredbi PAUSE, da bi se znalo na kojoj od njih je program zastao. U svakom slučaju, operator treba da ima spisak zastoja sa njihovim značenjem i opisom potrebne intervencije. Posle izvršene intervencije, operator pritiskom na jedan taster nastavlja rad programa i to od naredbe koja neposredno sledi iza PAUSE.

Naredba STOP (kraj programa) upotrebljava se u programu na onim mestima gde logički treba da bude kraj programa. Takvih naredbi može u jednom programu takođe biti više od jedne (razni

uzroci zbog kojih rad po programu više nije potrebno nastaviti). Opšti oblik naredbe je:

STOP n

gde je *n* konstanta istog oblika i veličine kao kod naredbe PAUSE. Posle izvršavanja naredbe STOP program se više ne može nastaviti. Iz istog razloga kao naredbe PAUSE, i naredbe STOP treba u nekim slučajevima razlikovati, što se postiže pisanjem odgovarajućih celobrojnih konstanti desno od reči STOP, kao i kod naredbe PAUSE.

Obe ove naredbe, PAUSE i STOP, deluju u vreme *izvršavanja* programa u računaru. Nasuprot tome, naredba END, koju smo već pomenuli i videli u jednom programu (vidi 4.4 i 5.2), deluje samo za vreme *prevođenja* programa, pokazujući prevodiocu gde je fizički (a ne logički) kraj izvornog programa.

Sada možemo dopuniti program za izračunavanje jačine struje u funkciji frekvencije, koji smo naveli kao primer u odeljku 5.2, i za koji smo rekli da nije potpun baš zbog nepostojanja ovih naredbi.

Istovremeno ćemo i sam zadatak postaviti opštije.

U ulaznoj datoteci nalazi se sada više grupa kartica, pri čemu u svakoj grupi prva nosi podatke-vrednosti za promenljive E, R, H i C, dok ostale kartice u grupi sadrže svaka po jednu vrednost za promenljivu F. Poslednja kartica F svake grupe, izuzev one koja se nalazi na kraju cele datoteke, sadrži negativnu vrednost, dok kartica F na kraju datoteke (znači, kartica koja je poslednja u poslednjoj grupi) nosi takvu vrednost za F, koja je sigurno veća od svake moguće vrednosti (npr. 1000000 ili još bolje, 9999999).

U ovom slučaju FORTRAN-program izgledaće ovako:

```
 12 READ (4, 200) E, R, H, C
200 FORMAT (4F10.0)
    WRITE (7, 300) E, R, H, C
300 FORMAT (4E20.7)
 68 READ (4, 200) F
    IF (F — 9999999.0) 8, 9, 9
  9 STOP 111
  8 IF (F) 4, 5, 5
  4 PAUSE 222
    GO TO 12
  5 A = E/SQRT(R**2 + (6.2832*F*H — 1./(6.2832*F*C)) **2)
    WRITE (7, 300) F, A
    GO TO 68
    END
```

Posle čitanja kartice sa vrednostima za promenljive E, R, H i C biće odmah i odštampan jedan red sa tim vrednostima. Dalje se uzastopno ponavlja čitanje vrednosti za promenljivu F, izračunavanje A i štampanje F i A, sve dok ne naiđe kartica sa negativnom vrednošću za F, kada se program zaustavlja i daje poruku PAUSE 222. Pritiskom na taster za nastavak rada po programu, čita se i štampa novi slog vrednosti za E, R, H i C i dalje se ponavlja prethodni postupak. Kada naiđe poslednja kartica datoteke (F = = 9999999.0), pojaviće se poruka STOP 111, što znači kraj rada po programu.

Pored ilustracije upotrebe naredbi PAUSE, STOP i END, ovde vidimo i koliko su tri nove naredbe za upravljanje doprinele povećanju mogućnosti programa: umesto crtanja samo jednog dijagrama zavisnosti struje od frekvencije, sada možemo istraživati i crtati više takvih dijagrama, za razne vrednosti parametara E, R, H i C. Naredbom PAUSE 222 omogućen je prekid rada i pre normalnog kraja programa (STOP 111), s tim što tada moramo dati uputstvo operatoru da rad prekine posle određenog broja poruka PAUSE 222.

Više naredbi za ulaz i izlaz može biti povezano sa samo jednom naredbom FORMAT. U našem programu vidimo da se dve naredbe READ pozivaju na naredbu 200 FORMAT, dok se dve naredbe WRITE pozivaju na naredbu 300 FORMAT. Pored toga, lista simbola u naredbama za ulaz i izlaz može da bude i kraća od broja odgovarajućih FORMAT-specifikacija. O tome će, međutim, biti više reči u Glavi 9.

Kod FORTRAN-prevodilaca za veće računare, naredba PAUSE može umesto konstante štampati i tekst. Tada se u samoj naredbi u izvornom programu željeni tekst piše između dva apostrofa, na primer PAUSE 'ZABELEŽI BROJ — PRITISNI START'.

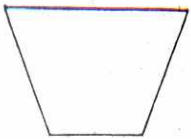
Treba napomenuti da se naredbe PAUSE i STOP moraju opravdano koristiti u programima, jer one često smanjuju efikasnost korišćenja računara u računskim centrima. Specijalno u velikim centrima, izbegava se upotreba naredbe PAUSE da se ne bi gubilo vreme usled čestog prekida rada po programu.

5.7 *GRAFIČKO PREDSTAVLJANJE PROGRAMA*

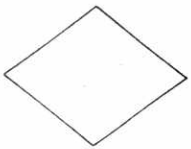
Svi problemi koje smo do sada pisali na FORTRAN-jeziku mogli su se izraziti manjim brojem naredbi, sa logički relativno jednostavnim vezama među njima. Kod dužih i složenijih zadataka, međutim, čovek više nije u stanju da u mislima drži pregled svih unutrašnjih veza u programu, da bi mogao FORTRAN-naredbe pisati

direktno u obrazac. Zato se uopšte u programiranju, pre pisanja izvornih naredbi, praktikuje grafičko predstavljanje logičkog toka programa pomoću tzv. *dijagrama toka* (česti nazivi su još *blok-dijagram* i *organigram*). Dijagram toka omogućuje da se u celini sagleda redosled operacija ili grupa operacija u programu, pre nego što se pristupi pisanju FORTRAN-naredbi. Postupajući na taj način, često se unapred izbegavaju komplikovana rešenja, a pregledani dijagram toka služi i kasnije kao dokumentacija programa i olakšava prezentaciju programa korisnicima.

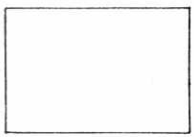
Za crtanje dijagrama toka usvojen je kao standard izvestan broj prostih geometrijskih figura, čiji oblik ukazuje na prirodu



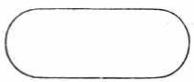
Trapezoid znači operaciju ulaza ili izlaza.



Romb predstavlja operaciju logičkog odlučivanja na osnovu vrednosti izraza.

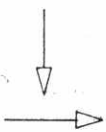


Pravougaonik označava sve ostale operacije računara.



Ovalni oblik označava početak i kraj programa.

Mali krug se upotrebljava da se izbegne ukrštanje linija na dijagramu ili da se povežu delovi dijagrama, crtani na posebnim listovima.



Linije sa strelicama povezuju figure dijagrama i pokazuju redosled izvodjenja operacija.

Slika 5.1 Figure za crtanje dijagrama toka

operacija koje su u njih upisane. Te figure se na dijagramu spajaju linijama i strelicama, tako da se jasno vidi tok akcije iz jedne operacije u drugu. Na slici 5.1 vidimo najčešće upotrebljavane oblike, zajedno sa objašnjenjem njihove upotrebe.

Pogledajmo najpre kako bi izgledali dijagrami toka za dva programa koje smo u odeljcima 5.2 i 5.6 uzimali kao primere. Ti dijagrami su dati na slikama 5.2 i 5.3.

Dijagram za zadatak izračunavanja jačine naizmenične struje, postavljen u odeljku 5.2, vidimo na slici 5.2. Grafička predstava je jednostavna kao što je jednostavan i sam zadatak: operacije ulaza vrednosti F, računanja i izlaza, ponavljaju se dok ima podataka u ulaznoj datoteci. Dijagram za isti zadatak, posle proširenja u odeljku 5.6, već je dosta složeniji. Prateći sliku toka operacija, sada već mnogo jasnije uočavamo njihove uzajamne veze pri raznim uslovima koje određuju ulazni podaci (slika 5.3).

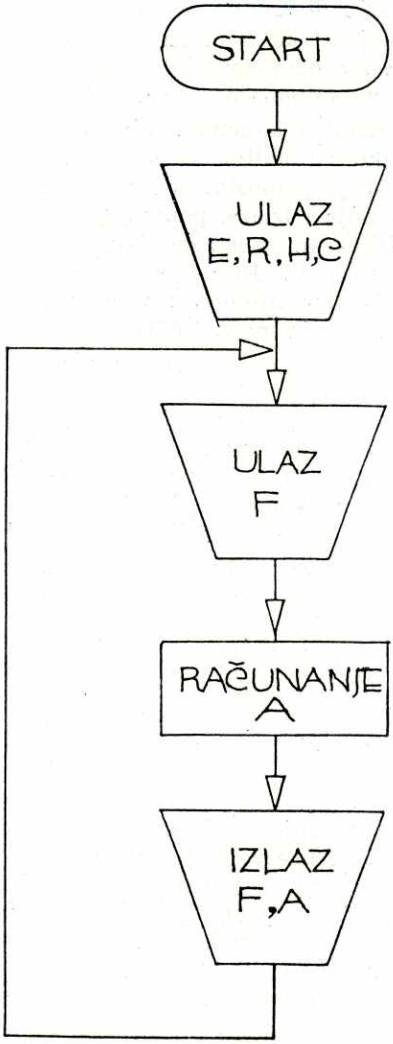
Kada se prema datom opisu zadatka nacrta dijagram toka, najpre treba proveriti njegovu logičku ispravnost, prateći u mislima kretanje ulaznih podataka kroz sve operacije. Pri tome treba ispitati rešenja ne samo za normalan slučaj ($0.0 < F < 9999999.0$) nego i za sve izuzetke ($F < 0.0$, $F > 9999999.0$). Tek kada je dijagram proveren i konačan, pristupa se pisanju programa na FORTRAN-jeziku, upoređujući ga stalno sa dijagramom toka.

Da na jednom primeru ilustrujemo crtanje još jednog složenijeg dijagrama toka, uzećemo zadatak koji zahteva upotrebu više aritmetičkih naredbi IF.

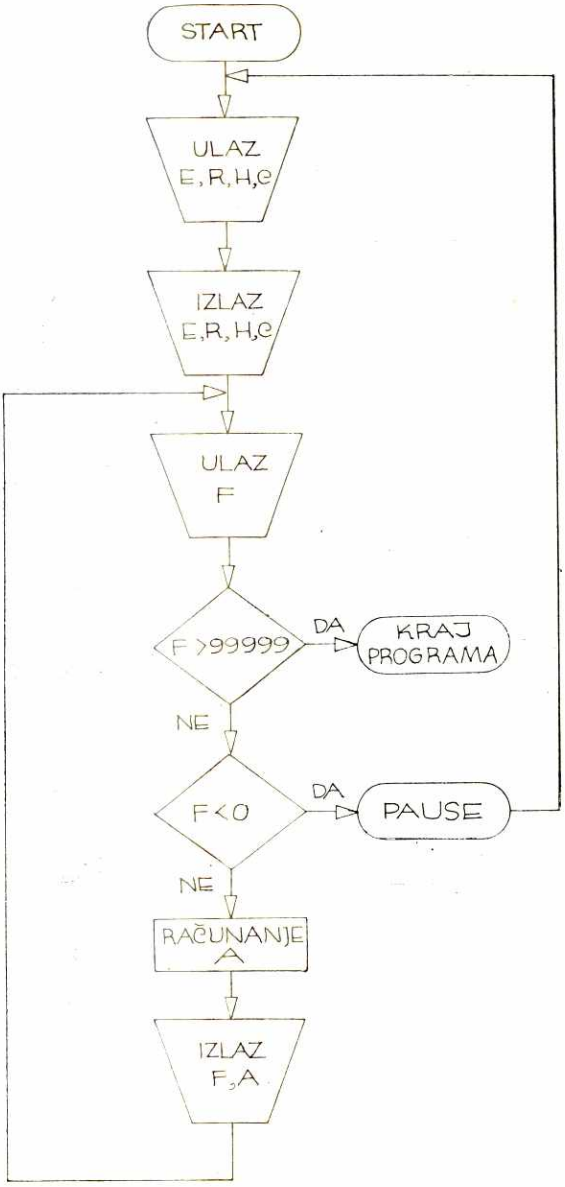
Ulazna datoteka sadrži 1200 kartica. U svaku karticu bušen je jedan broj oblika $\pm$0.XXXXXXXE$\pm$XX, u polju koje zauzima kolone 1—14. Treba prebrojati sve vrednosti koje su negativne, koje su nula i koje su pozitivne i te brojeve odštampati u jednom redu, zauzimajući za svaki broj po 10 pozicija. Dijagram toka dat je na slici 5.4.

Prvi korak je davanje vrednosti nula promenljivima NN, NO i NP, koje će na kraju programa ukazivati na broj negativnih, nula i pozitivnih vrednosti ulaznih podataka V. Takođe, početnu vrednost nula moramo dati i promenljivoj NK, koja predstavlja broj pročitanih kartica. U sledećem koraku ispitujemo da li je pročitana vrednost V negativna, nula ili pozitivna, pa na osnovu toga povećavamo za 1 vrednost odgovarajuće promenljive. Posle ovog povećavamo za 1 i broj do tada pročitanih kartica, NK. Sada ispitujemo da li su već pročitane sve kartice, pa ako nisu vraćamo se na čitanje sledeće kartice, a ako jesu štampamo rezultat, posle čega se pro-

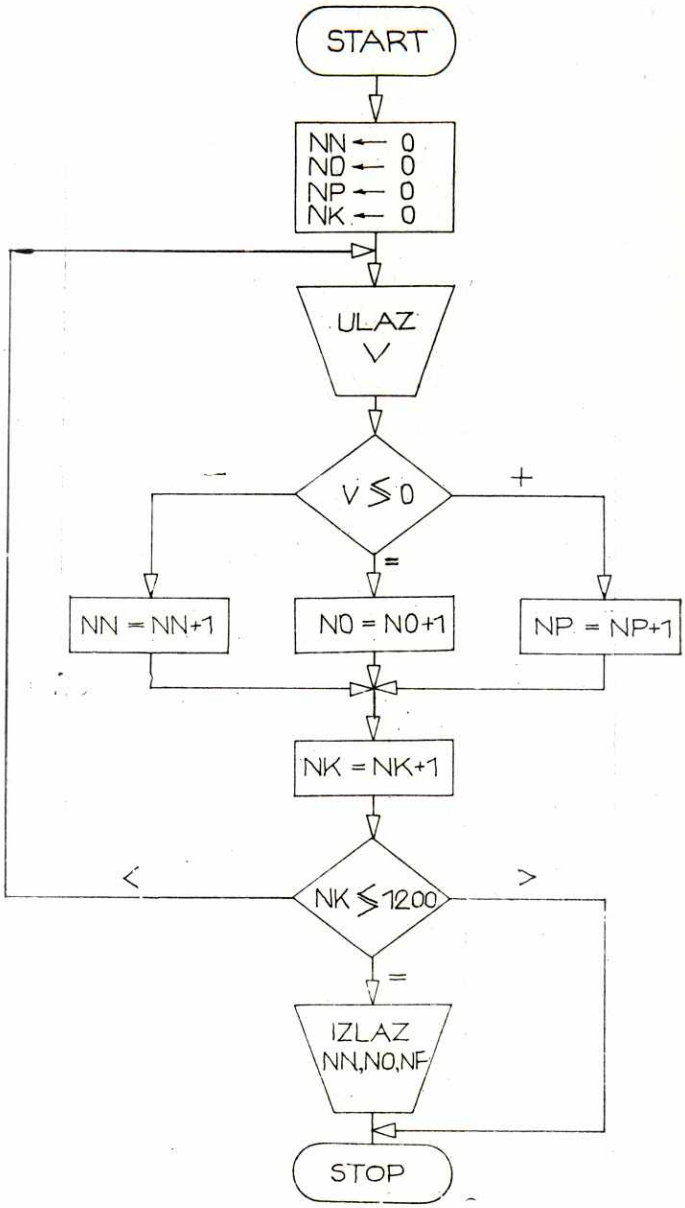
gram završava. Mogućnost da ukupan broj pročitanih kartica bude veći od 1200 teorijski ne postoji; ipak, naš program ćemo u tom slučaju završiti bez štampanja rezultata.



Slika 5.2 Dijagram toka za program iz odeljka 5.2



Slika 5.3 Dijagram toka za program iz odeljka 5.6



Slika 5.4 Dijagram toka za program iz odeljka 5.7

Po ovom dijagramu napisaćemo sada FORTRAN-program:

```
 50 FORMAT (E14.7)
 51 FORMAT (3I10)
    NN = 0
    NO = 0
    NP = 0
    NK = O
  3 READ (5, 50) V
    IF (V) 5, 4, 8
  5 NN = NN + 1
    GO TO 20
  4 NO = NO + 1
    GO TO 20
  8 NP = NP + 1
 20 NK = NK + 1
    IF (NK — 1200) 3, 25, 26
 25 WRITE (7, 51) NN, NO, NP
 26 STOP
    END
```

Na svim ovim primerima, koji su još uvek jednostavni u odnosu na slučajeve u stvarnoj praksi, ne može se u punoj meri sagledati korist od crtanja dijagrama toka. Ipak, u poslednja dva zadatka već se vidi koliko je niz operacija jasniji kada se predstavi grafički nego kada se samo opiše rečima. Kod još složenijih problema postaje potpuno nemoguće da se tok operacija prati bez ovakvog dijagrama. Zato se čitaocu posebno preporučuje da dobro ovlada ovom tehnikom i da je u radu uvek primenjuje.

VEŽBE

Za svaki od sledećih zadataka treba nacrtati dijagram toka i na osnovu dijagrama napisati odgovarajuće FORTRAN-naredbe. Pretpostaviti da sve promenljive već imaju konkretne vrednosti, dobijene u prethodnom delu programa. Ne treba, znači, pisati potpun program, sa naredbama za ulaz i izlaz ukoliko se izričito ne zahteva.

* 1. Uporediti vrednosti dveju promenljivih, x i y, i promenljivoj AMAX dati vrednost one koja je algebarski veća. Ako su ove vrednosti jednake, promenljivoj AMAX dati tu zajedničku vrednost.

2. Algebarski najveću od tri promenljive, x, y i z, izjednačiti sa promenljivom EMAX. (To ispitivanje se može rešiti sa samo dve aritmetičke naredbe IF: ispitati da li je veće x ili y, smestiti veću vrednost na neko privremeno mesto i zatim je uporediti sa z).

3. Promenljive, čiji su simboli U i V, mogu imati pozitivne ili negativne vrednosti. Veću po *apsolutnoj vrednosti* izjednačiti sa promenljivom UVMAX.

* 4. Za ugao GAMA zna se da je pozitivan i da nije veći od 24 radijana. Oduzeti 2π od GAMA potreban broj puta, da ostatak bude ugao manji od 2π. Za novu vrednost ugla zadržati isti simbol.

* 5. XREAL i XIMAG su realni i imaginarni deo jednog kompleksnog broja. Ako su *istovremeno* oba po apsolutnoj vrednosti manji od 1, preći na naredbu 21; u suprotnom slučaju preći na naredbu 22.

6. Y1, Y2 i Y3 su ordinate triju tačaka jedne krive u ravni. Ako je Y2 lokalni maksimum, tj. ako je istovremeno $Y2 > Y1$ i $Y2 > Y3$, preći na naredbu 35, inače na naredbu 46.

7. Ako je *apsolutna vrednost* promenljive DELTA manja od 10^{-5}, preći na naredbu 62, inače na naredbu 63. Napisati deo programa koji je za to potreban i to:
   a. ne upotrebljavajući standardnu funkciju ABS,
   b. pomoću standardne funkcije ABS.

* 8. Ako je $2.499 \leqslant x \leqslant 2.501$, preći na naredbu 51, inače preći na naredbu 52. Napisati potreban deo programa i to:
   a. sa dve aritmetičke naredbe IF, bez funkcije ABS,
   b. sa jednom aritmetičkom naredbom IF, u okviru koje je upotrebljena funkcija ABS.

* 9. Ako je $m = 3$ i $R < S$, preći na naredbu 827; ako je $m = 3$ i $R \geqslant S$, preći na naredbu 583; ako je $m \neq 3$, preći na naredbu 654.

* 10. Ako je $J = 1$, 3 ili 5, preći na naredbu 161; ako je $J = 2$ ili 4, preći na naredbu 162; ako je $J = 6$, 7, 8 ili 9, preći na naredbu 163; za svaku drugu vrednost J preći na naredbu 333.

11. U nekom programu treba na izvesnom mestu izvršiti prelaz na drugu naredbu i to:

    na naredbu 312, ako je $L = 1$
    na naredbu 325, ako je $1 < L < N$
    na naredbu 338, ako je $L = N$

* 12. Za sve vrednosti x od 1.0 do 49.9, u intervalima po 0.1, izračunati vrednosti funkcije y:

$$y = 24.6\,x + 16.7\,x^2 - 8.32\,x^3$$

Odštampati sve parove vrednosti x i y.

13. Izračunati y kao funkciju od x prema sledećem obrascu:

$$y = \sqrt{1 + x} + \frac{\cos 2x}{1 + \sqrt{x}}$$

za niz vrednosti promenljive x u jednakim intervalima. Sa jedne kartice se čitaju tri vrednosti koje određuju promenljivu x: XPOCET (početna vrednost), DELTAX (interval ili priraštaj) i XKONAC (konačna vrednost). Pretpostavlja se da je vrednost priraštaja pozitivna kao i da je XPOCET < XKONAC. Sve parove vrednosti x i y treba takođe i odštampati.

Glava 6

LOGIČKI IZRAZI I NAREDBE

6.1 *LOGIČKE VELIČINE*

Govoreći o brojevima sa kojima se može operisati u FORTRANu, pomenuli smo i jednu vrstu konstanti koje nemaju karakter brojeva (vidi 2.3 »Konstante«). To su *logičke konstante* .TRUE. i .FALSE. (*true* znači tačno, *false* pogrešno), koje za razliku od svih ostalih nemaju brojnu (numeričku ili aritmetičku) nego logičku vrednost. Te dve konstante se u izvornom FORTRAN-programu uvek pišu sa tačkama, ispred i iza odgovarajućih reči.

Logičke promenljive, analogno definiciji aritmetičke promenljive, su veličine predstavljene jednim simbolom umesto konkretnom logičkom vrednošću. Te promenljive mogu u toku izvršavanja programa dobijati posebne logičke vrednosti ili konstante — »tačno« (.TRUE.) ili »pogrešno« (.FALSE.). Nikakvu drugu vrednost, osim dve navedene konstante, logičke promenljive ne mogu imati. *Simbol* logičke promenljive u FORTRANu formira se po pravilima koja važe za aritmetičke promenljive, s tom razlikom što ovde nema smisla unutrašnja konvencija i druge naredbe za deklarisanje vrste broja, pa se simbol može birati tako da počinje bilo kojim slovom. Koji simboli u programu predstavljaju logičke promenljive veličine, određuje se naredbom za deklarisanje vrste LOGICAL (vidi 2.2 »Promenljive i njihovi simboli«).

6.2 *LOGIČKI IZRAZI*

Isto kao što se na FORTRANu, i uopšte na jezicima digitalnih računara, ne mogu direktno izraziti složenije matematičke operacije kao što su diferencijal, integral, operacije matrične algebre i ana-

lize i dr., nego se moraju po pravilima numeričke analize svesti na poznatih pet aritmetičkih operacija, tako se ne mogu direktno izraziti ni komplikovaniji odnosi među logičkim pojmovima. Dozvoljene logičke operacije u FORTRANu* jesu operacije upoređivanja (relacije) i operacije udruživanja (asocijacije).

Operatori za *operacije upoređivanja*, tzv. relacioni operatori, znače ispitivanje kvantitativnog odnosa dvaju *aritmetičkih* izraza. Takvih operatora ima u FORTRANu šest i to su:

.GT.	(*g*reater *t*han)	— veće od ($>$)
.GE.	(*g*reater or *e*qual)	— veće ili jednako ($\geqslant$)
.LT.	(*l*ess *t*han)	— manje od ($<$)
.LE.	(*l*ess or *e*qual)	— manje ili jednako ($\leqslant$)
.EQ.	(*e*qual)	— jednako ($=$)
.NE.	(*n*ot *e*qual)	— nije jednako ($\neq$)

Svi operatori za logičke operacije u FORTRANu pišu se između tačaka, koje prevodilac smatra sastavnim delom operatora.

Odnos (relacija), koji se izražava pomoću operatora upoređivanja, može biti tačan (true) ili pogrešan (false). Aritmetički izrazi koji učestvuju u FORTRAN-izrazu logičkog odnosa mogu biti samo oni čija je vrednost ceo broj (INTEGER) ili realan broj (REAL i DOUBLE PRECISION). Na primer, 3.NE.5 je jedan tačan odnos, tj. njegova logička vrednost je konstanta .TRUE., dok su 2.EQ.7 i 8.27.LE.3.14 pogrešni odnosi, čija je logička vrednost konstanta .FALSE.. Nedozvoljen logički odnos je, na primer, (12,5.3.7).GT. (8.5,2.6), u kome su aritmetički izrazi kompleksni brojevi.

Da bismo formirali složenije primere ove vrste logičkih izraza, pretpostavimo sledeće: neka su sve promenljive u izrazima deklarisane prema unutrašnjoj konvenciji, izuzev što je L logička, a C kompleksna promenljiva. Tada su, uz pretpostavku da FORTRAN-prevodilac dozvoljava mešovite aritmetičke izraze, sledeći logički odnosi pravilno napisani:

a. (R*A).GT.E
b. A.LT.I
c. E**2.7.EQ. (5*R + 4)
d. 57.9.LE.(4.7 + F)
e. 0.5.GE.9*R
f. E.EQ.27.3D+05

* logičke operacije udruživanja su moguće samo kod FORTRAN-prevodilaca za veće računare

Sledeći logički odnosi su nepravilni:

g.	C.LT.R	(kompleksni izraz se ne može stavljati u logički odnos)
h.	L.EQ.(A + F)	(logički izraz se ne može stavljati u logički odnos)
i.	E**2.EQ97.1E9	(izostavljena tačka iza operatora odnosa)
j.	.GT.9	(nedostaje aritmetički izraz ispred operatora odnosa)

Neka su nam vrednosti promenljivih u gornjim, ispravno napisanim izrazima, određene u programu ovako:

```
R = 0.8E2
A = 2.0
I = 1
E = 0.4E+01
F = 532.26
```

Tada će logička vrednost izraza a. i d. biti konstanta .TRUE., a svih ostalih konstanta .FALSE.. (Razume se, pre upoređivanja moraju već biti poznate vrednosti aritmetičkih izraza).

Kada se logički izrazi ovakve vrste međusobno povežu pomoću logičkih operatora za udruživanje, dobijaju se složeni logički izrazi. Odavde odmah vidimo da ova druga vrsta logičkih operatora udružuje samo čisto *logičke* veličine, tj. takve čija se vrednost svodi na jednu od logičkih konstanti .TRUE. ili .FALSE., dok operatori odnosa povezuju samo čisto *aritmetičke veličine*. Logičkih operatora udruživanja ili asocijacije ima u FORTRANu tri: .AND. (logičko »i«, operator konjunkcije), .OR. (logičko »ili«, operator disjunkcije) i .NOT. (logičko »ne«, operator negacije). Njihove definicije su:

.NOT. — ako logička promenljiva A ima vrednost .TRUE., tada izraz .NOT.A ima vrednost .FALSE.; ako pak A ima vrednost .FALSE., tada .NOT.A ima vrednost .TRUE.

.AND. — ako obe logičke promenljive, A i B, imaju vrednost .TRUE., tada izraz A.AND.B ima takođe vrednost .TRUE.; ako bilo A bilo B ili obadve promenljive imaju vrednost .FALSE., tada izraz A.AND.B ima vrednost .FALSE.

.OR. — ako bilo koja od logičkih promenljivih A i B ili obadve istovremeno imaju vrednost .TRUE., tada izraz A.OR.B ima vrednost .TRUE.; jedino kada A i B imaju vrednost .FALSE., onda je vrednost izraza A.OR.B .FALSE.

Ova pravila se mogu sažeto izraziti pomoću sledeće tablice:

A	B	.NOT.A	A.AND.B	A.OR.B
.FALSE.	.FALSE.	.TRUE.	.FALSE.	.FALSE.
.FALSE.	.TRUE.	.TRUE.	.FALSE.	.TRUE.
.TRUE.	.FALSE.	.FALSE.	.FALSE.	.TRUE.
.TRUE.	.TRUE.	.FALSE.	.TRUE.	.TRUE.

Dva logička operatora asocijacije mogu se pisati jedan do drugog jedino ako je drugi od njih .NOT..

Primera radi, od napred navedenih izraza a. i b. formirajmo složeni logički izraz sa operatorom .AND.:

(R*A).GT.E .AND. A.LT.I

Za napred date vrednosti aritmetičkih promenljivih R, A, I i E, ceo izraz biće logički »pogrešan« jer je već vrednost izraza A.LT.I konstanta .FALSE.. Isti parcijalni izrazi, povezani operatorom .OR. čine složeni logički izraz:

(R*A).GT.E .OR. A.LT.I

koji je u celini logički »tačan« zato što je vrednost levog izraza, (R*A).GT.E, konstanta .TRUE., za ranije date vrednosti za R, A, I i E.

Kao i u aritmetičkim, i u logičkim izrazima se za preciziranje redosleda izvođenja operacija upotrebljavaju zagrade; ako ima više parova zagrada koji se obuhvataju, prednost imaju operacije u najužem paru, zatim u prvom širem i tako dalje. Upotreba zagrada se preporučuje, da bi se obezbedio tačan smisao izraza. Pošto se u složenim logičkim izrazima pojavljuju svi operatori, aritmetički i logički, tamo gde redosled operacija nije određen zagradama važe sledeći rangovi:

izračunavanje vrednosti funkcije (najviši)
**
algebarski predznak

* i /
+ i —
.LT., .LE., .EQ., .NE., .GT., .GE.
.NOT.
.AND.
.OR. (najniži)

Na primer, konačna vrednost sledećeg složenog logičkog izraza određuje se postepeno, niže naznačenim redom:

A.GT.D**B.AND. .NOT.L.OR.N

D**B	vrednost je W	(A.GT.W.AND. .NOT.L.OR.N)
A.GT.W	vrednost je X	(X.AND. .NOT.L.OR.N)
.NOT.L	vrednost je Y	(X.AND.Y.OR.N)
X.AND.Y	vrednost je Z	(Z.OR.N)
Z.OR.N	konačna vrednost	

Ako treba da se negira ceo jedan logički izraz, tada se taj izraz mora pisati u zagradama. Uzmimo na primer poslednji deo gornjeg izraza (.NOT.L.OR.N) i neka je vrednost L konstanta .FALSE., a vrednost N konstanta .TRUE.. U tom slučaju, dva sledeća izraza nisu ekvivalentni:

.NOT.(L.OR.N)
.NOT.L.OR.N

U prvom izrazu najpre će biti određena logička vrednost u zagradama, L.OR.N, i to je konstanta .TRUE.. Pošto .NOT.(.TRUE.) znači .FALSE., to je konačna vrednost konstanta .FALSE..

U drugom izrazu najpre se određuje logička vrednost negacije, .NOT.L, i rezultat je konstanta .TRUE.. Dalje, logička vrednost izraza .TRUE. .OR.N postaje konstanta .TRUE., a to je i konačna vrednost celog tog izraza.

6.3 *LOGIČKA NAREDBA ZA DODELJIVANJE VREDNOSTI*

Analogno definiciji aritmetičke naredbe a = b, pod logičkom naredbom za dodeljivanje vrednosti podrazumevamo određivanje vrednosti jednog logičkog izraza na desnoj strani znaka jednakosti i dodeljivanje ili pripisivanje te vrednosti logičkoj promenljivoj na levoj strani. Opšti oblik naredbe je:

a = b

Važe analogno sva formalna pravila koja smo videli kod aritmetičke naredbe. Izvršavanjem naredbe, prethodna vrednost promenljive *a* je uništena, a njeno mesto zauzima nova vrednost, dobijena određivanjem vrednosti izraza *b*; u izrazu na desnoj strani, vrednosti svih konstanti i promenljivih ostaju sačuvane.

Neka se, u nekom programu, redom izvršavaju sledeće naredbe:

```
LOGICAL G, H
I = 4
G = .TRUE.
H = .NOT.G
G = 3..GT.I
H = .NOT.G.OR.H
```

Najpre će logička promenljiva G dobiti vrednost konstante .TRUE.. Zatim će logička promenljiva H dobiti vrednost konstante .FALSE.. U sledećoj naredbi, posle upoređivanja na desnoj strani, promenljiva G će dobiti vrednost konstante .FALSE. umesto ranije vrednosti .TRUE.. U poslednjoj naredbi, logička vrednost .NOT.G je konstanta .TRUE., dok je vrednost H još uvek konstanta .FALSE., pa će zato nova vrednost promenljive H postati konstanta .TRUE..

6.4 *LOGIČKE VELIČINE NA ULAZU I IZLAZU*

U polja kartice, koja sadrži ulazne podatke, logičke konstante .TRUE. i .FALSE. upisuju se svojim tekstom, bez tačaka. Umesto potpunih reči TRUE i FALSE, mogu se pisati samo njihova početna slova ili neki drugi tekst, sastavljen od znakova koje FORTRAN dozvoljava, s tim da prvi znak s leva, koji nije blanko, mora biti slovo T ili F. Na primer, niže navedene konstante, bušene u polja ulazne kartice, imaće za program sledeće značenje:

TRUE	.TRUE.
FALSE	.FALSE.
bbbbbbbbF	.FALSE.
Tbbbbb	.TRUE.
TANGENS	.TRUE.
bbFIbbb	.FALSE.
bbbbb	.FALSE.

gde je *b* blanko.

U poslednjem primeru vidimo da će vrednost biti konstanta .FALSE. čak i kada odgovarajuće polje ne sadrži nijedan drugi znak osim blanko.

Za čitanje logičkih konstanti pomoću naredbe READ i štampanje (ili uopšte izlaz) pomoću naredbe WRITE, postoji posebna FORMAT-specifikacija (kao što su za brojne veličine I, F, D i E), čiji je oblik:

Lw

gde je *w* broj znakova ulaznog ili izlaznog polja (vidi 9.3.5). Na primer, za gore navedene ulazne konstante trebalo bi sa naredbom za ulaz pisati u izvornom programu jednu naredbu FORMAT sa sledećim specifikacijama (gornjim redom):

L4, L5, L9, L6, L7, L7, L5

U izlaznim poljima logičke konstante će zauzimati *w* pozicija, od kojih će poslednja sadržati slovo T ili F, a sve ostale znake blanko.

6.5 *LOGIČKA NAREDBA IF*

Logička naredba IF je još jedna od naredbi za upravljanje, kao što su naredbe GO TO i aritmetička naredba IF, o kojima smo govorili u Glavi 5. Njen opšti oblik je:

IF (t) s

gde je *t* logički izraz, a *s* neka od aktivnih naredbi FORTRANa, izuzev naredbe IF i DO.

Dejstvo logičke naredbe IF je sledeće:

1. najpre se određuje logička vrednost izraza u zagradama, čija vrednost kao što znamo može biti samo jedna od logičkih konstanti, .TRUE. ili .FALSE.;
2. ako je vrednost logičkog izraza u zagradama konstanta .TRUE., odmah zatim će biti izvršena naredba *s*; ako se po naredbi *s* ne pređe na neku drugu naredbu u programu, posle nje se izvršava sledeća naredba po redu; ako je pak vrednost logičkog izraza konstanta .FALSE., program se nastavlja sledećom naredbom posle te logičke naredbe IF.

Pogledajmo dva primera. U prvom imamo jednu aritmetičku naredbu u istom redu sa logičkom naredbom IF, a u drugom logičkoj naredbi IF sledi jedna naredba za bezuslovni prelaz.

Prvi primer

```
IF (A.GT.B) C = A**2
K = K + 1
```

Ako je A veće od B, izraz u zagradama je logički tačan (.TRUE.). U tom slučaju se odmah zatim izvršava naredba C = A**2, pa tek onda naredba K = K + 1.

Ako je A manje ili jednako B, izraz u zagradama je logički pogrešan (.FALSE.). Program se odmah nastavlja sa naredbom K = K + 1.

Drugi primer

```
IF (A.GT.B) GO TO 18
K = K + 1
```

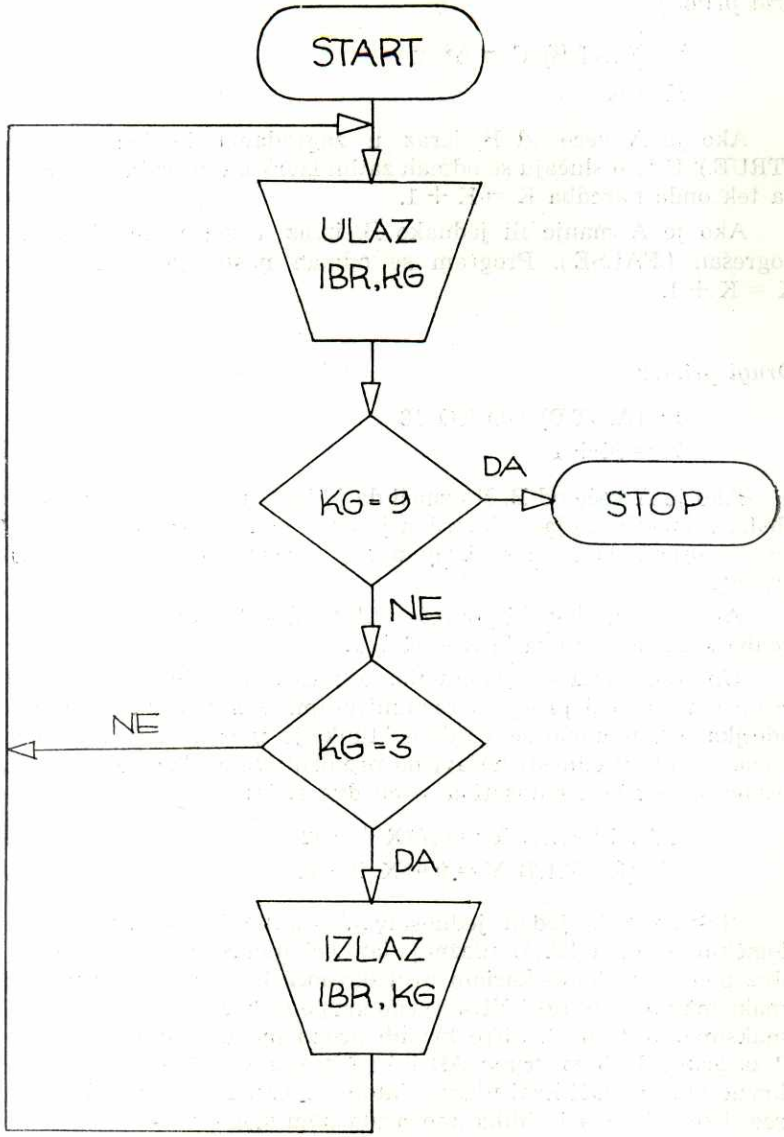
Ako je A veće od B, što znači da je izraz logički tačan (.TRUE.), sledeća naredba koja se izvršava je GO TO 18 i akcija se nastavlja izvršavanjem naredbe sa brojem 18; naredba K = K + 1 neće se izvršiti.

Ako je izraz logički pogrešan (.FALSE.), tj. $A \leqslant B$, sledeća naredba koja se izvršava je K = K + 1.

Upotreba logičkih operacija, a naročito logičke naredbe IF, često čini izvorni program razumljivijim. Uzmimo onaj zadatak iz odeljka 5.3, u kome se vrednost funkcije y izračunavala alternativno za dve vrednosti nezavisno promenljive x. Taj deo programa bismo sada mogli napisati u samo dva reda:

```
IF (X.LE.1.7) Y = 0.7*X + 0.62
IF (X.GT.1.7) Y = 0.9*X + 0.41
```

Rešimo sada jedan jednostavan praktični zadatak primenom logičkih naredbi IF. U ulaznoj datoteci imamo više kartica sa po dva podatka: identifikacioni broj davaoca krvi (pozitivan ceo broj, maksimalno 5 cifara) i šifra krvne grupe (takođe pozitivan ceo broj, maksimalno 2 cifre). Šifre krvnih grupa mogu biti: 1 za grupu A, 2 za grupu B, 3 za grupu AB i 4 za grupu 0. Cifra 9 u polju šifre krvne grupe znači kraj ulazne datoteke. Treba odštampati po jedan red, istog sastava i oblika kao u ulaznom slogu, za svakog davaoca krvi grupe AB. Radi upoređenja, isti program ćemo napisati pomoću logičkih i pomoću aritmetičkih naredbi IF.



Slika 6.1 Dijagram toka za prvi primer iz odeljka 6.5

Dijagram toka je dat na slici 6.1, a niže su dv. FORTRAN-programa, sa kojima je zadatak rešen.

```
1 FORMAT (I5, I2)
2 READ (5,1) IBR, KG
  IF (KG.EQ.9) STOP 111
  IF (KG.EQ.3) WRITE (7,1) IBR, KG
  GO TO 2
  END
```

```
1 FORMA
2 READ (5
  IF (KG -
3 IF (KG -
4 WRITE (7,1) IBR, KG
  GO TO 2
5 STOP 111
  END
```

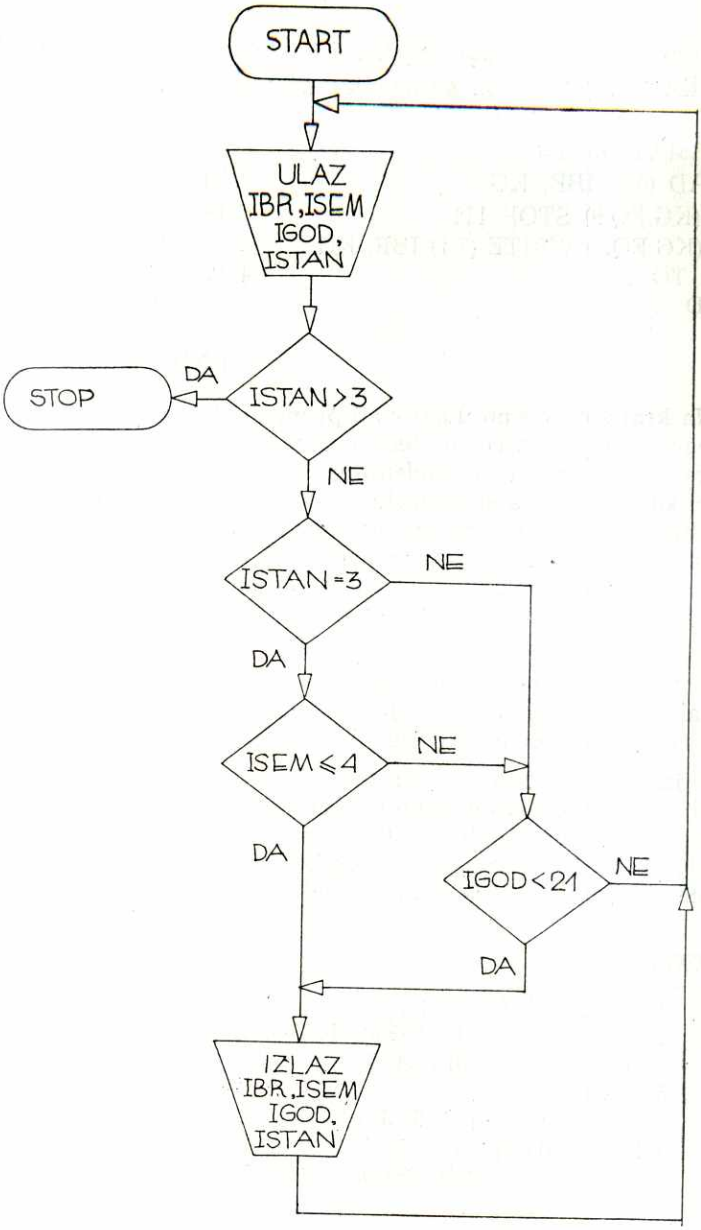
Na kraju, da bismo ilustrovali primenu složenih logičkih izraza u programiranju, uzećemo sledeći primer. Ulazna datoteka sadrži kartice sa podacima o studentima nekog fakulteta. Na svakoj kartici su četiri polja sa podacima za jednog studenta: indentifikacioni broj (5 kolona), semestar na koji je upisan (2 kolone), godine starosti (3 kolone) i šifra stanovanja (2 kolone). Podaci u svim poljima su numerički i brojevi su uvek bušeni do desne granice polja. Šifra stanovanja može biti: 1 — studentski dom, 2 — privatan stan, 3 — kod roditelja, 4 — oznaka kraja datoteke.

Potrebno je da se odštampaju podaci, sadržani u ulaznom slogu, za svakog studenta koji stanuje kod roditelja, a upisan je na neki semestar do uključivo četvrtog ili je mlađi od 21 godine.

Dijagram za rešenje ovog zadatka vidimo na slici 6.2. Prema tom dijagramu, program ćemo napisati na tri načina: (a) primenom samo aritmetičkih naredbi IF, (b) pomoću obe vrste naredbi GO TO za uslovni prelaz i sa prostim logičkim naredbama IF, i (c) primenom složenih logičkih izraza u jednoj logičkoj naredbi IF.

REŠENJE (A)

```
1 FORMAT (I5, I2, I3, I2)
2 READ (5,1) IBR, ISEM, IGOD, ISTAN
  IF (ISTAN — 3) 2, 4, 5
5 STOP 111
4 IF (ISEM — 4) 6, 6, 3
3 IF (IGOD — 21) 6, 7, 7
6 WRITE (7,1) IBR, ISEM, IGOD, ISTAN
7 GO TO 2
  END
```



Slika 6.2 Dijagram toka za drugi primer iz odeljka 6.5

REŠENJE (B)

```
    1 FORMAT (I5, I2, I3, I2)
    2 ASSIGN 8 TO L
    8 READ (5,1) IBR, ISEM, IGOD, ISTAN
      GO TO (8, 8, 4, 5), ISTAN
    4 IF (ISEM.LE.4) ASSIGN 11 TO L
      IF (IGOD.LT.21) ASSIGN 11 TO L
      GO TO L, (8, 11)
   11 WRITE (7,1) IBR, ISEM, IGOD, ISTAN
      GO TO 2
    5 STOP 111
      END
```

REŠENJE (C)

```
    1 FORMAT (I5, I2, I3, I2)
    2 READ (5,1) IBR, ISEM, IGOD, ISTAN
      IF (ISTAN.GT.3) STOP 111
     0IF ( (ISTAN.EQ.3).AND.(ISEM.LE.4.OR.IGOD.LT.21) )
     1WRITE (7,1) IBR, ISEM, IGOD, ISTAN
      GO TO 2
      END
```

Sva tri programska rešenja su ekvivalentna. Izbor će zavisiti od toga da li FORTRAN-prevodilac našeg računara sadrži ili ne sadrži logičke operacije. U rešenju (c) morali smo napisati jedan produžni red zbog dužine logičke naredbe IF i pripadajuće naredbe WRITE, koje inače čine jednu celinu.

VEŽBE

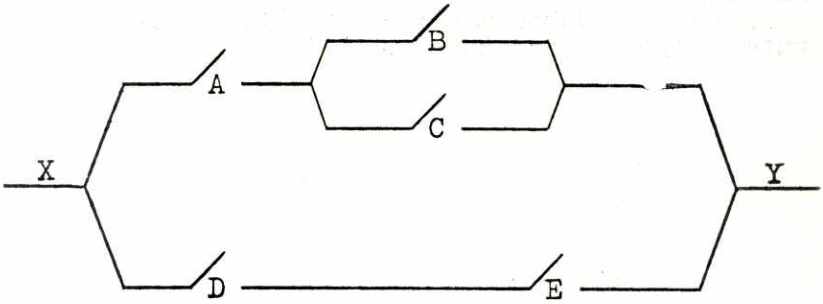
Za svaki od sledećih zadataka treba nacrtati dijagram toka i na osnovu dijagrama napisati potrebne FORTRAN-naredbe. Pretpostaviti da sve promenljive već imaju konkretne vrednosti, dobijene u prethodnom delu programa. Ne treba, znači, pisati potpun program, sa naredbama za ulaz i izlaz, ukoliko se to izričito ne zahteva.

1. Upotrebljavajući logičke naredbe IF, rešiti sledeće vežbe iz prethodne glave: *4, *5,*8, 9, 10 i 12
2. Rešiti zadatak iz odeljka 5.7 (dijagram na slici 5.4) pomoću logičkih naredbi IF.

*3. Ako je $0.999 \leqq x \leqq 1.001$, preći na naredbu STOP, inače preći na naredbu 69. Rešiti na dva načina:
   a. pomoću jedne logičke naredbe IF sa dva operatora odnosa (upoređivanja) i jednim operatorom .AND.;
   b. pomoću jedne logičke naredbe IF sa samo jednim operatorom odnosa (upoređivanja), uz primenu standardne funkcije za apsolutnu vrednost.

*4. XREAL i XIMAG su realni i imaginarni deo jednog kompleksnog broja. Promenljivoj X2 dati vrednost 1 ako su oba dela po apsolutnoj vrednosti manja od jedinice, inače preći na sledeću naredbu po redu.

*5. Pomoću logičkih operatora asocijacije u jednoj logičkoj naredbi za dodeljivanje vrednosti, izraziti uslov da postoji veza između tačaka X i Y na datoj šemi:



Postojanje veze iskazuje se vrednošću .TRUE. logičke promenljive Y, nepostojanje vrednošću .FALSE.. Prekidači A, B, C, D i E predstavljaju se logičkim promenljivima: vrednost .FALSE. odgovara otvorenom, vrednost .TRUE. zatvorenom prekidaču.

* 6. Na karticama se nalaze sledeći podaci: identifikacioni broj poreskog obveznika (I6), lični dohodak (F8.2) i broj izdržavanih članova porodice (I2). Osnovica za oporezivanje je lični dohodak, umanjen za 20000 i po 3000 za svakog izdržavanog člana. Na osnovicu do 5000 porez je 3%, u intervalu 5001—10000 4,5%, a u intervalu 10001—15000 7,5%. Za svakog obveznika treba obračunati porez i štampati ulazne podatke i dobijeni iznos poreza. Za one obveznike kod kojih osnovica premašuje 15000, štampati samo identifikacioni broj. Na kraju štampati jedan red sa brojem obveznika u ulaznoj datoteci i ukupno obračunatim porezom (na početku programa staviti nule kao početne vrednosti za broj obveznika i ukupno obračunati porez!). Na kraju ulazne datoteke je jedna kartica sa brojem 99 u trećem polju.

Glava 7

INDEKSNE PROMENLJIVE

7.1 *MATRICE*

Matricom* se u FORTRANu naziva skup promenljivih koje sve zajedno imaju jedan isti simbol. Jedna određena promenljiva, *član* ili *element* matrice, definisana je tim zajedničkim simbolom i *indeksima* (jedan ili više celih brojeva), koji se pišu u zagradama iza simbola. Broj indeksa predstavlja nam broj *dimenzija* matrice, odakle nazivi jednodimenzionalna, dvodimenzionalna i višedimenzionalna matrica. Jedan član matrice predstavljen je, dakle, jednim simbolom i indeksima u zagradama, neposredno iza simbola; zato se takva promenljiva zove *indeksna promenljiva* ili promenljiva sa indeksima.

Uzmimo jednu jednodimenzionalnu matricu, čiji je simbol M, a koja se sastoji od pet članova sa vrednostima 162, 85, 3296, 31, 4. To znači da je vrednost indeksne promenljive M(1) jednaka 162, da indeksna promenljiva M(2) ima vrednost 85, M(3) vrednost 3296 itd. Ako je, recimo, simbol M usvojen za jedan skup od pet raznih veličina mase, matematički oblik pisanja ovih indeksnih promenljivih bi bio: m_1, m_2, m_3, m_4, m_5. Sve te veličine mogu se predstaviti jednom zajedničkom oznakom, m_i, što se u FORTRANu piše analogno kao M(I), a znači I-ti član jednodimenzionalne matrice M, gde indeks I može imati vrednost od 1 do 5.

Dvodimenzionalnu matricu možemo zamisliti tako da su joj članovi raspoređeni u više horizontalnih redova i vertikalnih kolona. Pored simbola takve matrice pišu se u zagradama dva indeksa, odvojena zarezom, od kojih prvi znači *broj reda*, a drugi *broj kolone*. Najmanja vrednost i jednog i drugog indeksa je 1:

* Pojam matrice koji se ovde upotrebljava nije identičan sa pojmom matrice u matematici

vrednost prvog može ići najviše do ukupnog broja redova, a drugog najviše do ukupnog broja kolona matrice.

Prema dogovorenom načinu predstavljanja, uzmimo dvodimenzionalnu matricu A, čiji prvi indeks ima vrednosti 1 do 5, a drugi 1 do 3:

	kolona 1	kolona 2	kolona 3
red 1	82	4	7
red 2	12	13	15
red 3	91	1	31
red 4	24	16	10
red 5	3	8	6

Ako nam je u programu potrebna vrednost člana u drugom redu i trećoj koloni (15), pisaćemo A(2,3). Član sa vrednošću 16 poziva se pisanjem A(4,2) itd. U matematici se za opšti slučaj bilo kog člana matrice piše $a_{i,j}$, dok se u FORTRANu to piše A(I,J), gdje je za gornji primer I = 1 do 5, a J = 1 do 3.

Kao primer za trodimenzionalnu matricu uzmimo matricu PREM, kod koje je prva dimenzija starost osiguranika (IGOD), druga pol osiguranika (IPOL), a treća vrednost ili iznos polise osiguranja (IZNOS). Članovi matrice su vrednosti premija koje uplaćuju osiguranici. Pogledajmo tabelu 7.1 gde su data značenja za sve vrednosti indeksa.

Neka treba da se odredi visina premije za osiguranika koji je star 29 godina, muškog pola i želi osiguranje na iznos od 50000 dinara. Iz tabele 7.1 vidimo da za taj slučaj indeksi imaju vrednosti prikazane u tabeli str. 113:

IGOD = 6
IPOL = 1
IZNOS = 7

Vrednost premije za taj slučaj dobićemo ako u programu napišemo PREM (6,1,7).

Broj članova matrice je proizvod maksimalnih vrednosti svih njenih indeksa. Tako je za primer jednodimenzionalne matrice M taj broj bio 5, za dvodimenzionalnu matricu A $5 \times 3 = 15$, a za trodimenzionalnu matricu PREM $20 \times 2 \times 8 = 320$.

Za razliku od indeksnih promenljivih, promenljive bez indeksa, koje smo do sada isključivo upotrebljavali, zovu se proste, obične ili *skalarne promenljive*. Za simbole indeksnih promenljivih važe ista pravila koja smo već naveli za skalarne promenljive, tj. unu-

TABELA 7.1

Godine starosti		Pol	
1- 5	IGOD = 1	muški	IPOL = 1
6-10	= 2	ženski	= 2
11-15	= 3		
16-20	= 4	Vrednost polise	
21-25	= 5		
26-30	= 6	1000	IZNOS = 1
⋮	⋮	2000	= 2
		3000	= 3
		5000	= 4
		10000	= 5
		25000	= 6
		50000	= 7
96-100	IGOD = 20	100000	= 8

trašnja konvencija, eksplicitne naredbe INTEGER, REAL, DOUBLE PRECISION, COMPLEX i LOGICAL i naredba IMPLICIT. Jedino što je ovde novo jeste to da deklarisanjem vrste simbola matrice, time ujedno deklarišemo vrstu svih njenih članova. Iz toga sledi pravilo da svi članovi matrice moraju biti homogeni u pogledu vrste brojeva.

7.2 *INDEKSI*

Kao što smo videli, indeksi su celi brojevi koji određuju mesto indeksne promenljive u okviru matrice. Broj indeksa, dakle broj dimenzija matrice, kod većine FORTRAN-prevodilaca ne može biti veći od 3, dok je kod nekih 7 (vidi tabelu 12.2). Kod višedimenzionalnih matrica indeksi se odvajaju zarezima, a svi moraju biti zatvoreni u zajednički par zagrada.

Indeks ne mora uvek biti samo celobrojna konstanta nego može da bude i skalarna promenljiva, pa čak i do izvesne mere složen aritmetički izraz, u jednom od sledećih oblika:

v
c'
v + c'
v — c'
c*v
c*v + c'
c*v — c'

gde je *v* jedna skalarna celobrojna promenljiva, pisana bez algebarskog znaka, dok su *c* i *c'* proizvoljne celobrojne konstante, takođe pisane bez algebarskog znaka.

FORTRAN-prevodioci za veće računare* dozvoljavaju za indekse i složenije aritmetičke izraze, u kojima se javljaju svi FORTRAN-aritmetički operatori, simboli standardnih i programiranih funkcija, indeksne promenljive, veličine raznih vrsta. Od vrsta brojeva dozvoljeni su samo celi i realni brojevi (REAL i DOUBLE PRECISION). Posle izračunavanja vrednosti izraza po pravilu 3 i tabelama 2.2 i 2.3 (Glava 2), ona se pretvara u ceo broj.

U svakom slučaju, ako je indeks neki aritmetički izraz, njegova vrednost mora da bude veća od nule, a manja ili najviše jednaka veličini odgovarajuće dimenzije matrice. Sasvim malobrojni su takvi FORTRAN-prevodioci koji dozvoljavaju da vrednost indeksa bude manja ili jednaka nuli (vidi tabelu 12.2).

Pogledajmo nekoliko primera pravilno napisanih indeksnih promenljivih; poslednja tri dozvoljena su samo kod većih računara:

```
AMATR (IBR)
NUM (23)
A (5*L)
MATR (I—5, K*J+8)
BAK (I, J(K+2*L, 0.6*A(M,N)))
AMATR (I, J/4*K**2)
```

Sledeće indeksne promenljive su nepravilno napisane; oblici u poslednja dva primera nisu dozvoljeni ni kod većih računara jer sadrže logičke i kompleksne veličine:

* na primer IBM 360, prevodilac H

AMATR (—8)	(indeks ne sme da bude negativan)
LAMBDA (0)	(indeks ne sme da ima vrednost 0)
EBL (—7*J)	(konstanta u indeksu mora biti bez algebarskog znaka)
TOT (K*2)	(dozvoljen je samo oblik: 2*K)
TOT (2+K)	(dozvoljen je samo oblik: K+2)
ASR (1.GE.I)	(indeks ne može da bude logička konstanta .TRUE. ili .FALSE.)
NOVI (1+(1.3,2.0))	(indeks ne može imati kompleksnu vrednost)

7.3 *NAREDBA DIMENSION*

Veličina matrice izražava se ukupnim brojem njenih članova, a on je određen brojem indeksa (dimenzija) i maksimalnom vrednošću svakog indeksa. Ta informacija se u programu mora dati za svaku matricu, kako bi u memoriji računara bio rezervisan odgovarajući prostor. Rezervisanje se može izvršiti na tri načina; jedan od tih načina je naredba DIMENSION. Drugi način je pomoću naredbe COMMON, a treći pomoću neke od naredbi za deklarisanje vrste (INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL). Drugi i treći način biće opisani u Glavi 10.

Opšti oblik naredbe DIMENSION je:

DIMENSION $v_1(c_1)$, $v_2(c_2)$, . . . , $v_n(c_n)$

gde su v_i simboli matrica, a c_i grupe celobrojnih konstanti, pisanih bez algebarskog znaka, međusobno odvojenih zarezima; u jednoj grupi c_i može biti maksimalno tri odnosno sedam brojeva, što zavisi od FORTRAN-prevodioca.

Pomoću jedne naredbe DIMENSION može se, dakle, odrediti veličina većeg broja matrica. To ne znači da u programu može da se piše samo jedna naredba DIMENSION, naprotiv, njih može biti proizvoljan broj. Za većinu FORTRAN-prevodilaca, naredba DIMENSION mora u programu biti napisana ispred one naredbe u kojoj se prvi put koristi indeksna promenljiva, koja je član te matrice.

Na primer, ako naredba u programu glasi

DIMENSION X(20), A(3,15), M(3,3,8)

prevodilac će rezervisati mesto u memoriji za dvadeset članova jednodimenzionalne matrice X, četrdeset pet članova dvodimenzionalne

matrice A i sedamedeset dva člana trodimenzionalne matrice M. Ako u istom programu nisu korišćene eksplicitne naredbe za deklarisanje vrste promenljivih X, A i M, tada će po unutrašnjoj konvenciji FORTRANa svi članovi matrica X i A biti realni, a svi članovi matrice M celi brojevi. Pišući naredbe u kojima upotrebljava te indeksne promenljive, programer mora da vodi računa da nijedan indeks ne bude veći od njegove dimenzije, definisane naredbom DIMENSION.

I naredba DIMENSION spada u tzv. neaktivne naredbe (kao FORMAT, INTEGER, REAL itd.), tj. ona samo daje određene informacije FORTRAN-prevodiocu, a ne izaziva nikakvu akciju u mašinskom programu. Zato, izuzev uslova koji smo napred pomenuli, naredba DIMENSION se može pisati bilo gde u programu, na primer između dveju aritmetičkih naredbi. Kasnije ćemo još videti i da naredba DIMENSION ne može biti prva u ciklusu jedne naredbe DO.

Sa nekoliko izuzetaka, o kojima će kasnije biti reči, indeksne promenljive se mogu upotrebljavati na svim onim mestima gde se upotrebljavaju i skalarne promenljive. Kao jedan jednostavan primer za to, napišimo potrebne naredbe za ulaz koeficijenata a_{ij} i slobodnih članova b_i u programu za rešavanje sistema linearnih jednačina sa dve nepoznate:

```
      DIMENSION A(2,2), B(2), X(2)
      READ (5,20) A(1,1), A(2,1), A(1,2), A(2,2), B(1), B(2)
   20 FORMAT (6F10.0)
```

Redosled ulaza pojedinih elemenata određen je pomoću liste u naredbi READ. Mogli smo izabrati i bilo kakav drugi poredak u toj listi. Važno je samo da se i same vrednosti koeficijenata i slobodnih članova moraju u ulaznom slogu nalaziti u istom poretku.

Ovakav slučaj, da se na ulazu ili na izlazu javljaju svi članovi jedne matrice, dosta je čest u programiranju. Zato je dozvoljen i takav oblik naredbe READ odnosno WRITE, gde se navodi samo simbol matrice, bez informacije o indeksima. Tada ta naredba znači ulaz odnosno izlaz cele matrice. U gornjem primeru smo mogli, na primer, pisati:

```
      READ (5,20) A, B
```

i u memoriju bi, u rezervisane zone za matrice A i B, bile prenete vrednosti za sve članove obeju matrica.

Pošto ovde FORTRAN-prevodiocu nije eksplicitno saopšteno u kom poretku se nalaze članovi matrica na ulaznom medijumu, ra-

zume se da mora postojati neka konvencija u pogledu tog redosleda. Takva konvencija zaista i postoji i sastoji se u sledećem. Ako je matrica jednodimenzionalna, računar smešta pročitane vrednosti po rastućem redosledu njenih indeksa. Na primer, računar očekuje da je prva vrednost ona koja odgovara članu B(1), zatim ona koja odgovara članu B(2). Kod dvodimenzionalne matrice računar će smatrati da su vrednosti za njene članove poređane na ulaznom medijumu *po kolonama*. Na primer, tim redom su upravo napisani članovi matrice A u listi prethodne naredbe READ: A(1,1), A(2,1), A(1,2), A(2,2). Isto važi i za višedimenzionalne matrice: opšte pravilo je da se *najbrže* menja *prvi*, a *najsporije poslednji* indeks. To možemo ilustrovati primerom jedne trodimenzionalne matrice, C (3,2,4); ako je u ulaznoj naredbi naveden samo simbol C, vrednosti će se smeštati sledećim redom (s leva na desno i odozgo na dole):

C(1,1,1), C(2,1,1), C(3,1,1), C(1,2,1), C(2,2,1), C(3,2,1)
C(1,1,2), C(2,1,2), C(3,1,2), C(1,2,2), C(2,2,2), C(3,2,2)
C(1,1,3), C(2,1,3), C(3,1,3), C(1,2,3), C(2,2,3), C(3,2,3)
C(1,1,4), C(2,1,4), C(3,1,4), C(1,2,4), C(2,2,4), C(3,2,4)

Gledano u smislu adresa u memoriji računara, na najnižu adresu će doći član C(1,1,1), do njega član C(2,1,1), zatim C(3,1,1), C(1,2,1), ..., C(2,1,4), C(3,1,4), C(1,2,4), C(2,2,4), C(3,2,4).

Analogno važi i za izlaz, na primer štampanje vrednosti članova matrice kada se u listi naredbe WRITE izostave indeksi.

7.4 *PROSTIJI PRIMERI UPOTREBE INDEKSNIH PROMENLJIVIH*

Date su dve jednodimenzionalne matrice, X i Y, obe sa po tri člana. Članovi prve matrice su koordinate tačke X u prostoru, a članovi druge matrice koordinate tačke Y. Rastojanje tih dveju tačaka dato je obrascem:

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + (x_3 - y_3)^2}$$

što se piše aritmetičkom naredbom u FORTRANu:

```
D = SQRT ((X(1)—Y(1))**2+(X(2)—Y(2))**2+(X(3)—Y(3))**2)
```

Na ovom mestu možemo se podsetiti koje sve značenje mogu imati zagrade u FORTRANu. U gornjem primeru ilustrovana su tri glavna slučaja:

1. zagrade grupišu faktore za aritmetičke i logičke operacije u složenijim izrazima,

2. sa njima se obuhvataju argumenti funkcija ili argumenti FORTRAN-naredbi,
3. upotrebljavaju se za pisanje indeksa kod indeksnih promenljivih.

Kao drugi primer, na jednoj kartici nalaze se četiri koeficijenta $a_{i,j}$ i dva slobodna člana b_i sistema linearnih jednačina sa dve nepoznate; takvih kartica ima više u ulaznoj datoteci. Poslednja kartica sadrži samo jednu vrednost, različitu od nule, u polju od 61. do 70. kolone; ta kartica će služiti za označavanje kraja ulazne datoteke.

Ako sistem jednačina napišemo koristeći indeksnu notaciju:

$$a_{1,1}x + a_{1,2}y = b_1$$
$$a_{2,1}x + a_{2,2}y = b_2$$

rešenja se dobijaju po sledećim obrascima:

$$x = \frac{a_{2,2}\,b_1 - a_{1,2}\,b_2}{a_{1,1}\,a_{2,2} - a_{2,1}\,a_{1,2}}$$

$$y = \frac{a_{1,1}\,b_2 - a_{2,1}\,b_1}{a_{1,1}\,a_{2,2} - a_{2,1}\,a_{1,2}}$$

Program za rešavanje proizvoljnog broja sistemâ ovakvih jednačina mogao bi se napisati ovako:

```
10 FORMAT (8E14.7)
20 FORMAT (7F10.0)
   DIMENSION A (2,2), B (2)
 5 READ (5,20) A, B, D
   IF (D) 2, 4, 2
 2 STOP 333
 4 C = A(1,1)*A(2,2) — A(2,1)*A(1,2)
   IF (ABS(C) — 1.0E-5) 3, 6, 6
 3 PAUSE 222
   GO TO 5
 6 X = (A(2,2)*B(1) — A(1,2)*B(2))/C
   Y = (A(1,1)*B(2) — A(2,1)*B(1))/C
   WRITE (7,10) A, B, X, Y
   GO TO 5
   END
```

Prema naredbi READ, vrednosti koeficijenata i slobodnih članova moraju u karticu biti bušene sledećim redom: $a_{1,1}$, $a_{2,1}$, $a_{1,2}$, $a_{2,2}$,

b_1, b_2, a u obliku koji propisuje naredba FORMAT, tj. F10.0. Naredbom WRITE te vrednosti će biti odštampane u jednom redu i to u istom poretku; na kraju reda štampaće se još vrednosti x i y.

Pošto se u obrascima za obe nepoznate javlja isti imenilac, to je prvom aritmetičkom naredbom programirano njegovo izračunavanje, da bi se izbeglo ponavljanje aritmetičkih operacija. Sledećom aritmetičkom naredbom IF obezbeđuje se da imenilac ne bude nula, što bi izazvalo grešku i nemogućnost daljeg nastavka programa. Prethodnom aritmetičkom naredbom IF ispituje se polje D u ulaznoj kartici, koje je u svim karticama prazno (što kod ulaznih podataka u FORTRANu znači nula) sem u poslednjoj.

Treba napomenuti da je u ispitivanju imenioca upotrebljena jedna konačna, vrlo mala vrednost (1.0E-5 = 10^{-5}), koja će dovesti do zaustavljanja programa i u slučaju ako vrednost imenioca nije tačno nula. Ovakvo ispitivanje vrši se često u programima, sa ciljem da se spreči da vrednost nekog količnika ne premaši maksimalno dozvoljenu veličinu za realne brojeve. Ukoliko se takav slučaj ne reši programski (kao u ovom primeru ili slično), računar će prilikom izvršavanja stati sa porukom o *prekoračenju kapaciteta,* posle čega se rad programa više ne može nastaviti. Analogna situacija se javlja kada neka vrednost postane suviše mala u odnosu na minimalno dozvoljenu veličinu za realne brojeve, za što postoji posebna poruka. (Videti o ovome i u odeljku 12.5).

7.5 *KORIST OD UPOTREBE INDEKSNIH PROMENLJIVIH*

U prethodnim odeljcima objašnjeni su osnovni pojmovi i data pravila za indeksne promenljive. Međutim, iz jednostavnih primera nismo još mogli videti stvarnu korist od takvog načina predstavljanja promenljivih veličina. Izuzev primera sa matricom premija životnog osiguranja (odeljak 7.1), koji je dat kao ilustracija za trodimenzionalnu matricu, svi drugi primeri su se očigledno mogli rešiti isto tako dobro sa skalarnim promenljivima. Zašto bi onda indeksne promenljive bile neka naročito značajna karakteristika FORTRANa?

Razlog leži upravo u svojstvu indeksâ da mogu biti promenljive veličine ili čak izvesni tipovi aritmetičkih izraza. Koristeći to svojstvo, možemo napisati program za jedan osnovni ciklus izračunavanja, a zatim menjanjem indeksa učiniti da se taj isti ciklus ponavlja za nove vrednosti promenljivih.

Uzmimo da treba izračunati zbir kvadrata 20 brojeva, x_1 do x_{20}, koji se već nalaze u računaru. Mogli bismo, razume se, svakom od tih brojeva dati poseban simbol i napisati jednu dugu aritmetičku naredbu za sabiranje kvadrata. Umesto toga, pretpostavićemo da su ti brojevi u memoriji organizovani u vidu jedne jednodimenzionalne matrice, sa zajedničkim simbolom X. Tako možemo pozivati svaki član pišući simbol X(I), a u programu ćemo udesiti da indeks I redom menja sve vrednosti od 1 do 20. Prema tome, matematički obrazac

$$s = \sum_{i=1}^{20} x_i^2$$

možemo u FORTRAN-programu jednostavno realizovati na sledeći način:

```
      SUMA2 = 0.0
      I = 1
    3 SUMA2 = SUMA2 + X(I)**2
      I = I + 1
      IF (I — 20) 3, 3, 4
    4 . . . . . .
```

Na početku, promenljivoj SUMA2 dajemo vrednost nula, a indeksu I vrednost 1. Naredba 3 je ustvari *radni obrazac*, koji će na kraju programa dati traženu vrednost zbira kvadrata. U prvom ciklusu I ima vrednost 1, pa će prvim izvršavanjem naredbe 3 biti izračunat kvadrat prvog člana matrice. Zatim se toj vrednosti promenljive I dodaje jedinica i ispituje da li je $I > 20$. Ako nije, naredba 3 se ponavlja još jedanput, a ako jeste, završeno je formiranje zbira kvadrata i program prelazi na naredbu 4.

Da bi se dobio zbir kvadrata svih 20 brojeva, poslednje tri naredbe se moraju izvršiti tačno 20 puta. U Glavi 8 ćemo videti da se upotrebom naredbe DO ovaj deo programa može rešiti još jednostavnije, tako da ćemo kasnije samo izuzetno koristiti način prikazan u ovom primeru.

Kao drugi primer, neka se u programu traži vrednost y jednog polinoma drugog stepena za tri grupe koeficijenata:

$$y = ax^2 + bx + c \quad \text{ako je } K = 1$$
$$y = dx^2 + ex + f \quad \text{ako je } K = 2$$
$$y = gx^2 + hx + i \quad \text{ako je } K = 3$$

Vrednosti K i x određene su ranije u programu, na bilo koji način. Mi smo već videli da mogu da se napišu tri aritmetičke naredbe sa različitim koeficijentima, pa da se sa jednom uslovnom naredbom GO TO izabere ona koja odgovara. Postupak je, međutim, prostiji ako se koriste indeksi.

Neka su nam koeficijenti polinoma članovi jedne jednodimenzionalne matrice, čiji je simbol C. To znači, C(1) = a, C(2) = b, . . ., C(9) = i. Ako sada za sve kvadratne članove polinoma usvojimo kao indeks aritmetički izraz 3*K — 2, za sve članove prvog stepena izraz 3*K — 1 i za sve slobodne članove izraz 3*K, tada možemo u programu pisati samo jednu aritmetičku naredbu:

```
Y = C(3*K—2)*X**2 + C(3*K—1)*X + C(3*K)
```

i vrednost polinoma biće uvek računata sa odgovorajućom grupom koeficijenata.

Da su koeficijenti bili uneti u matricu ovako: C(1) = a, C(2) = d, C(3) = g, C(4) = b, C(5) = e, C(6) = h, C(7) = c, C(8) = f i C(9) = i, tada bi gornja naredba trebalo da glasi:

```
Y = C(K)*X**2 + C(K+3)*X + C(K+6)
```

Drugi način, na koji se dobijaju isti rezultati, jeste da se koeficijenti organizuju u vidu dvodimenzionalne matrice C, na primer:

```
a b c
d e f
g h i
```

Za prvi indeks u ovom slučaju treba uzeti promenljivu K, a drugi indeks će biti 1, 2 ili 3, pa će aritmetička naredba biti:

```
Y = C(K,1)*X**2 + C(K,2)*X + C(K,3)
```

Iz varijanata ovog primera videli smo kako se promenljive sa indeksima mogu iskoristiti za alternativni izbor vrednosti iz jednog skupa, organizovanog u vidu matrice, i na taj način uprostiti pisanje naredbi u programu.

Najčešća primena indeksnih promenljivih u programima jeste u izvođenju iste grupe operacija sa različitim vrednostima promenljivih. Takvu primenu moći ćemo videti u sledećoj Glavi, gde se svojstva indeksne promenljive koriste u vezi sa naredbom DO.

VEŽBE

U sledećim zadacima treba napisati i naredbu DIMENSION i, gde je to potrebno, odgovarajuću eksplicitnu naredbu za deklarisanje vrste (kasnije ćemo videti da se za simbole matrica, dakle indeksnih promenljivih, funkcija naredbe DIMENSION može uključiti u eksplicitnu naredbu za deklarisanje vrste). Ove naredbe pisati na početku programa.

* 1. Koordinate jedne tačke u prostoru date su kao članovi jedne jednodimenzionalne matrice, čiji je simbol X (uočiti značenje pojma dimenzije: članovi *jednodimenzionalne* matrice su ovde koordinate tačke u *trodimenzionalnom* prostoru!). Napisati naredbu za izračunavanje rastojanja te tačke od koordinatnog početka; rastojanje je kvadratni koren zbira kvadrata pojedinih koordinata.

2. Duž koja spaja tačku iz prethodnog zadatka sa koordinatnim početkom zove se *poteg te* tačke. Napisati naredbe za izračunavanje kosinusa ugla koji zaklapa poteg sa koordinatnim osama, uzimajući da su koordinate tačke date u vidu jednodimenzionalne matrice X. Kosinusi uglova dati su obrascima:

$$c_1 = \frac{x_1}{\sqrt{x_1^2 + x_2^2 + x_3^2}}$$

$$c_2 = \frac{x_2}{\sqrt{x_1^2 + x_2^2 + x_3^2}}$$

$$c_3 = \frac{x_3}{\sqrt{x_1^2 + x_2^2 + x_3^2}}$$

* 3. Date su dve dvodimenzionalne matrice, A i B. Napisati naredbe za izračunavanje članova treće dvodimenzionalne matrice, C, prema sledećim jednačinama. Maksimalna veličina svih indeksa je 2.

$$c_{11} = a_{11} \cdot b_{11} + a_{12} \cdot b_{21}$$
$$c_{12} = a_{11} \cdot b_{12} + a_{12} \cdot b_{22}$$
$$c_{21} = a_{21} \cdot b_{11} + a_{22} \cdot b_{21}$$
$$c_{22} = a_{21} \cdot b_{12} + a_{22} \cdot b_{22}$$

(Ovo su obrasci za množenje dveju kvadratnih matrica reda 2×2).

4. Data je dvodimenzionalna matrica R, čiji su članovi elementi jedne determinante 3×3. Napisati naredbu za izračunavanje vrednosti determinante nekom od poznatih metoda. Vrednost determinante označiti simbolom D.

* 5. Dve jednodimenzionalne matrice, A i B, imaju po 30 članova. Svi članovi su realni brojevi dvostruke tačnosti. Izračunati:

$$D = \left(\sum_{i=1}^{30} (A_i - B_i)^2 \right)^{1/2}$$

(Ovo je obrazac za rastojanje dveju tačaka u 30-dimenzionalnom prostoru).

* 6. Jedna jednodimenzionalna matrica, X, sadrži 50 članova. Članove druge matrice, DX, izračunati po obrascu:

$$DX(I) = X(I+1) - X(I) \qquad I = 1, 2, \ldots, 49$$

(Ovo je obrazac za izračunavanje *prvih razlika*).

7. Trideset dve ordinate jedne eksperimentalno dobijene krive čine jednodimenzionalnu matricu Y. Uz pretpostavku da se apscise tačaka međusobno razlikuju za konstantnu vrednost H, koja je data u programu, napisati program za približno izračunavanje integrala krive po trapeznom pravilu:

$$s = \frac{h}{2}(y_1 + 2y_2 + 2y_3 + \ldots + 2y_{30} + 2y_{31} + y_{32})$$

8. Dvodimenzionalna matrica AMATR ima 10 redova i 10 kolona. Formirati jednodimenzionalnu matricu DIAG od dijagonalnih članova matrice AMATR ($d_i = a_{i,i}$).

* 9. Data je jednodimenzionalna matrica Y od 50 članova, date su vrednosti promenljivih U i I. Napisati aritmetičku naredbu za izračunavanje vrednosti S po sledećem obrascu:

$$s = y_i + u\,\frac{y_{i+1} - y_{i-1}}{2} + \frac{u^2}{2}(y_{i+1} - 2y_i + y_{i-1})$$

(Ovo je Sterlingova interpolaciona formula drugog reda)

10. Uz pretpostavke iz zadatka 9, napisati naredbu za izračunavanje vrednosti *t* po sledećem obrascu:

$$t = y_i + u(y_{i+1} - y_i) + \frac{u(u-1)(y_{i+2} - y_{i+1} - y_i + y_{i-1})}{4} + \frac{\left(u - \frac{1}{2}\right)u(u-1)(y_{i+2} - 3y_{i+1} + 3y_i - y_{i-1})}{6}$$

Članovi matrice Y su realni brojevi dvostruke tačnosti. (Gornji obrazac je Beselova interpolaciona formula trećeg reda).

* 11. Date su dve jednodimenzionalne matrice, A i B, svaka od po sedam članova. Sedam članova matrice A bušeno je u jednu karticu tako da vrednosti zauzimaju polja po 10 kolona. Na isti način bušene su vrednosti članova matrice B u drugu karticu. Za ulaz treba upotrebiti specifikaciju F10.0. Treba izračunati vrednost c po obrascu:

$$c = \sqrt{\sum_{i=1}^{7} a_i b_i}$$

Izračunatu vrednost c odštampati po specifikaciji 1PE20.7.

12. Uz pretpostavke iz zadatka 11, napisati program koji će u istom redu sa vrednošću c odštampati 1 ako je $a_i > b_i$ za svako *i*, odnosno 0 ako taj uslov nije zadovoljen.

* 13. C je jedna jednodimenzionalna matrica od 30 članova, koji su kompleksni brojevi. Napisati naredbe za formiranje zbira apsolutnih vrednosti svih članova matrice, koristeći standardnu funkciju CABS (vidi tabelu 12.3).

14. Uz pretpostavke iz zadatka 13, napisati naredbe za formiranje zbira imaginarnih delova onih članova matrice čiji su imaginarni delovi pozitivni.

* 15. Jednodimenzionalna matrica BUL ima 40 članova, koji su logičke veličine (konstante .TRUE. odnosno .FALSE.). Napisati program koji će pod simbolom TRUE prebrojati sve vrednosti .TRUE., a pod simbolom FALSE sve vrednosti .FALSE.. Simboli TRUE i FALSE mogu biti simboli za promenljive bilo koje vrste; samo kada se pišu između tačaka, tada su to logičke konstante.

Glava 8

PROGRAMSKI CIKLUSI — NAREDBA DO

8.1 *UVOD*

U ranijim primerima često smo se sretali s potrebom da se jedna grupa naredbi u programu ponavlja više puta. Videli smo da broj ponavljanja može biti nepoznat unapred i da može zavisiti od ulaznih podataka (na primer, kraj datoteke) ili od nekog drugog uslova u programu. Neki put je, međutim, potrebno da se grupa naredbi ponavlja određen broj puta: u tom slučaju se u program uvodi *brojač,* tj. jedna celobrojna promenljiva, čija vrednost se povećava sa svakim izvršavanjem određenih naredbi. Ispitivanjem brojača pomoću logičke ili aritmetičke naredbe IF utvrđuje se da li je ponavljanje izvršeno određeni broj puta, pa ako jeste, program se nastavlja sa sledećom naredbom.

Opisano ponavljanje često se javlja u programiranju. Grupa naredbi koje se ponavljaju zove se zatvoreni krug ili ciklus (zatvorena petlja) odnosno prostije samo *ciklus (petlja).* Broj izvršenih ponavljanja ogleda se u vrednosti brojača, koji se naziva *indeks.*

Neka u sledećem programu treba da se pročitaju ulazni podaci sa 20 kartica, što znači da se dvadeset puta ponove naredbe 3, 4 i 5:

```
1 FORMAT (3F5.1)
2 FORMAT (F7.1, 3F5.1)
8                          <-----------------+
3 READ (5,1) A, B, C                  potrebno
4 S = A + B + C                          20
                                     ponavljanja
5 WRITE (7,2) S, A, B, C  -------------------+
  STOP 555
  END
```

Naredba 8 treba da definiše granice ciklusa, da odredi brojaču neku početnu i krajnju vrednost i da utvrdi interval ili priraštaj za koji će se vrednost brojača menjati pri svakom ponavljanju. Takva naredba, koja programu daje sve te elemente, za koje smo do sada morali upotrebljavati više naredbi, jeste naredba DO. Njen opšti oblik je:

DO n i = m_1, m_2, m_3

gde je n broj poslednje naredbe u ciklusu ili petlji, *i indeks* ili brojač naredbe DO, m_1 *početna vrednost* indeksa, m_2 *krajnja vrednost* indeksa ($i \leqslant m_2$) i m_3 *interval* ili *priraštaj* indeksa. Sve tri veličine m_1, m_2 i m_3 zovu se zajedničkim imenom *parametri* indeksa naredbe DO.

Broj n u naredbi DO mora da bude broj neke naredbe u istom programu, koja nije ispred same naredbe DO, i može da bude samo celobrojna skalarna promenljiva (bez indeksa), a m_1, m_2 i m_3 moraju biti ili celobrojne konstante, pisane bez algebarskog znaka, ili celobrojne skalarne promenljive. U pisanju naredbe DO može se izostaviti jedino indeksni parametar m_3, u kom slučaju se podrazumeva da je on 1.

Primenimo sada naredbu DO u prethodnom primeru. Uporedo ćemo napisati isti program na način koji smo do sada primenjivali.

```
 1 FORMAT (3F5.1)
 2 FORMAT (F7.1, 3F5.1)
   DO 3 I = 1, 20
   READ (5,1) A, B, C
   S = A + B + C
 3 WRITE (7,2) S, A, B, C
   STOP 555
   END
```

```
 1 FORMAT (3F5.1)
 2 FORMAT (F7.1, 3F5.1)
   I = 0
 5 READ (5,1) A, B, C
   S = A + B + C
   WRITE (7,2) S, A, B, C
   I = I + 1
   IF (I — 20) 3, 4, 4
 3 GO TO 5
 4 STOP 555
   END
```

Dok je u programu, napisanom na stari način, ponavljanje pod kontrolom aritmetičke naredbe IF i naredbe GO TO, a pored toga su potrebne i aritmetičke naredbe za dodeljivanje početne vrednosti i za porast brojača, sve te funkcije sada obavlja samo jedna naredba DO.

Još efikasniju primenu naredba DO ima kada je njen indeks ujedno i indeks neke indeksne promenljive u njenom ciklusu. Uzmi-

mo isti primer sumiranja kvadrata dvadeset brojeva, koji smo imali u odeljku 7.5; sada će taj deo programa biti:

```
      SUMA2 = 0.0
      DO 37 I = 1, 20
   37 SUMA2 = SUMA2 + X(I)**2
      . . . . . .
```

U ovom slučaju, indeks I naredbe DO ima ujedno i ulogu indeksa promenljive X; u nekom drugom slučaju on bi mogao biti član u aritmetičkom izrazu, čija je vrednost indeks promenljive, na primer Y(2*I—1). U prvoj ulozi, indeks I ima dimenziju *vremena* jer je svako sledeće izvršavanje ciklusa *kasnije* od prethodnog, dok u drugoj ulozi I ima dimenziju *prostora* jer pokazuje *mesto* indeksne promenljive u okviru matrice. Spajanje te dve uloge kod jedne iste promenljive vrlo je korisno i često u programiranju, a postiže se pomoću naredbe DO.

Za razliku od prethodnog primera, gde je ciklus naredbe DO obuhvatao tri sledeće naredbe u programu, u ovom poslednjem primeru ciklus obuhvata samo jednu naredbu (37). U opštem slučaju, proizvoljan broj naredbi jednog programa može se nalaziti u ciklusu naredbe DO.

Ako bismo hteli da nam program formira zbir kvadrata samo neparnih članova jednodimenzionalne matrice X, isti deo programa bi trebalo da glasi:

```
      SUMA2 = 0.0
      DO 37 I = 1, 20, 2
   37 SUMA2 = SUMA2 + X(I)**2
      . . . . . .
```

8.2 *OSTALE DEFINICIJE*

Posle osnovnih informacija o naredbi DO, iz kojih smo videli njenu ulogu u programu, razmotrimo sada detaljnije kako se ona u izvornom programu koristi.

Kada u programu prvi put dođe red na naredbu DO, njenim prvim izvršavanjem dodeljuje se početna vrednost indeksu i ($i = m_1$) i računar registruje broj n naredbe koja će biti poslednja u ciklusu. Početak ciklusa određen je položajem same naredbe DO: to je prva naredba iza naredbe DO. Ciklus čine sve naredbe koje se nalaze između ove dve.

Sve naredbe u ciklusu izvršavaju se redom, prvi put sa vrednošću indeksa $i = m_1$. Posle izvršavanja naredbe n, akcija se ponovo vraća na naredbu DO. Novo izvršavanje naredbe DO sastoji se u povećanju indeksa i za vrednost priraštaja m_3 i ispitivanju da li je i još uvek jednako ili manje od m_2 ($i \leqslant m_2$). Ako jeste, ceo ciklus se ponavlja još jedanput, sa novom vrednošću indeksa i. Posle naredbe n ponovo se izvršava naredba DO i ciklus se ponavlja, sve dok povećanjem indeksa ne nastupi slučaj $i > m_2$. Taj slučaj mora nastupiti *prilikom jednog izvršavanja naredbe DO*. Kada se ovo dogodi, akcija u programu prelazi na prvu naredbu posle naredbe n.

Važno je uočiti da je naredba DO *poslednja* naredba koja će se izvršiti pre nego što se nastavi dalji rad programa. Ovo važi u slučaju koji je gore opisan, a to je slučaj napuštanja ciklusa posle ispunjenja svih uslova koje postavlja naredba DO na početku ciklusa. Taj način se zove *normalni izlazak* iz ciklusa.

Drugu mogućnost za napuštanje ciklusa imamo kada se među naredbama u okviru ciklusa nalazi neka naredba GO TO, aritmetička ili logička naredba IF. Tada može doći do izlaska iz ciklusa pre nego što budu ispunjeni uslovi naredbe DO, zadati na početku ciklusa. U tom slučaju, naredba DO neće biti poslednja izvršena naredba pre izlaska iz ciklusa.

Posledice ova dva moguća načina izlaska iz ciklusa iteracije su različite: (a) ako je izlazak bio normalan, vrednost indeksa i je neodređena i ona se ne može upotrebiti u daljem radu programa. Na primer, u programu

```
      . . . . . .
      DO 30 K = 1, 8
      . . . . . .
30    . . . . . .
40    J = K
```

posle normalnog izlaska iz ciklusa, naredba 40 se neće moći izvršiti jer je vrednost K neodređena. S druge strane, (b) ako je ciklus napušten ranije, vrednost indeksa i je ona koja je bila u momentu izlaska i ona se može koristiti dalje u programu. Na primer, ako je u sledećem programu ciklus napušten pre normalnog izlaska:

```
7     . . . . . .
      DO 19 K = 1, 20, 3
      IF (X — Y) 8, 6, 8
8     . . . . . .
```

```
19 . . . . . .
   GO TO 7
 6 J = K
```

promenljiva J dobiće po naredbi 6 poslednju (tekuću) vrednost indeksa K, koju je ovaj već imao u momentu izvršavanja naredbe IF.

Ukupan broj ponavljanja do normalnog izlaska iz ciklusa može se izračunati po obrascu:

$$\left[\frac{m_2 - m_1}{m_3}\right] + 1$$

gde se u zagradama uzima prva celobrojna vrednost koja je manja od tačne vrednosti tog izraza.

Pogledajmo sada još dva karakteristična primera korišćenja indeksa naredbe DO. U jednom zadatku potrebno je da se izračuna proizvod svih celih brojeva od 1 do M, pri čemu je vrednost M određena ranije u programu. Uzimajući M za simbol, mi smo samim tim označili da je to celobrojna promenljiva (nijednom eksplicitnom naredbom za deklarisanje vrste nismo za simbol M ukinuli važnost unutrašnje konvencije). Međutim, za dalje računanje je potrebno da proizvod bude realan broj. Ako će naš program biti preveden na jezik mašine pomoću FORTRAN-prevodioca koji dozvoljava mešovite aritmetičke izraze, tekuća vrednost proizvoda može se slobodno množiti tekućom celobrojnom vrednošću I jer će se pri svakoj operaciji obaviti potrebno pretvaranje po pravilu 3 za aritmetičke operacije i tabeli 2.2. Ako će se, međutim, program koristiti na računaru čiji FORTRAN-prevodilac ne dozvoljava mešovite izraze, tada tekuću vrednost faktora I moramo prethodno pretvoriti u realan broj. Deo programa koji formira traženi proizvod napisaćemo zato u dve verzije:

Mešovit aritmetički izraz

```
   PROD = 1.0
   DO 8 I = 2, M
 8 PROD = PROD*I
```

Homogen aritmetički izraz

```
   PROD = 1.0
   DO 8 I = 2, M
   EI = I
 8 PROD = PROD*EI
```

Vrednost proizvoda formiraće se repetitivnim postupkom pod simbolom PROD, koji pre početka iteracija dobija vrednost 1. Ciklus će se ponavljati M-1 puta, sa rastućom vrednošću faktora I. U primeru na desnoj strani, ciklus počinje aritmetičkom naredbom EI = I, čija je jedina svrha da celobrojnu vrednost I pretvori u realan broj. Poslednje izvršavanje ciklusa biće sa I = M, posle čega je vrednost promenljive PROD jednaka proizvodu svih celih brojeva od 1 do M.

Drugi primer se odnosi na slučaj kada su u radnoj naredbi u okviru ciklusa potrebni intervali manji od 1. To ne možemo rešiti direktno pomoću indeksa naredbe DO jer, kao što znamo, njegov interval ili priraštaj ne može biti manji od 1. Pogledajmo u sledećem primeru kako se to može rešiti kada jedna promenljiva treba redom da prođe kroz niz vrednosti u intervalima po 0.1:

```
      DO 33 I = 200, 2000
      S = I
      S = 0.1*S
   33 WRITE (5,40) S
```

Indeks I naredbe DO proći će redom kroz vrednosti 200 do 2000, u intervalima po 1. Promenljiva S će u isto vreme redom dobijati vrednosti od 20 do 200, u intervalima po 0.1. Ako FORTRAN-prevodilac dozvoljava mešovite artimetičke izraze, tada nisu potrebne obadve naredbe za S nego je dovoljna samo jedna, S = 0.1*I.

8.3 *PRAVILA ZA KORIŠĆENJE NAREDBE DO*

Uz pridržavanje izvesnih pravila, dozvoljena je veoma fleksibilna upotreba naredbe DO. Najpre ćemo navesti sva ta pravila, a kasnije ćemo najvažnija od njih ilustrovati primerima.

1. Nijedna naredba u ciklusu naredbe DO ne sme da menja indeksne parametre, tj. veličine i, m_1, m_2 i m_3 dotične naredbe DO. Napred smo videli kako se ti brojevi praktično koriste u radnim naredbama u okviru ciklusa, s tim što vrednosti indeksnih parametara nisu menjane.
2. U okviru ciklusa jedne naredbe DO može se nalaziti ciklus druge naredbe DO; prvi u tom slučaju zovemo *spoljni*, a drugi *unutrašnji* ciklus. Celina od dva ili više obuhvaćenih

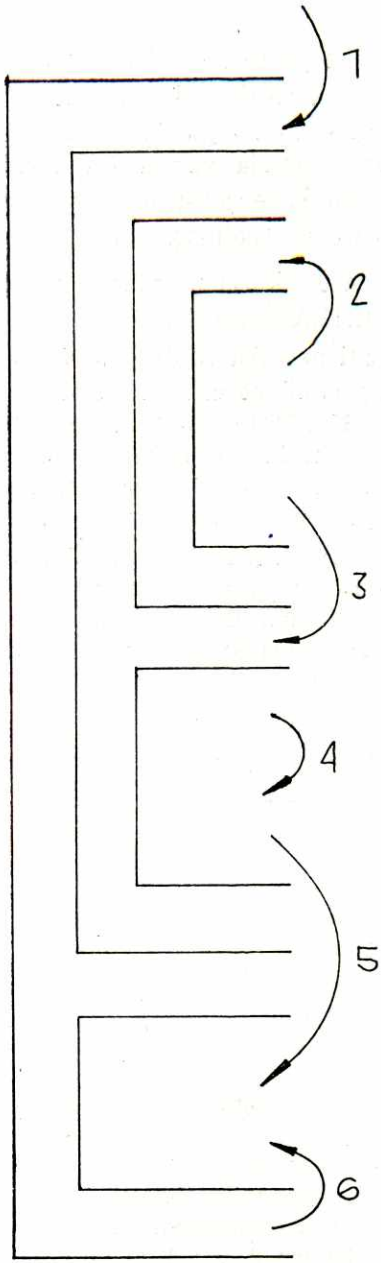
ciklusa zove se *gnezdo*. U takvim slučajevima, uslov je da sve naredbe unutrašnjeg ciklusa budu u granicama spoljnog ciklusa. Time se ne isključuje mogućnost da se dva ili više ciklusa jednog gnezda završavaju istom naredbom, samo nije dozvoljeno da se ijedan unutrašnji ciklus proteže preko poslednje naredbe spoljnog, koji ga obuhvata.

3. Prva naredba u ciklusu naredbe DO mora uvek biti jedna aktivna FORTRAN-naredba. Znači, na tom mestu se ne smeju nalaziti naredba DIMENSION, naredba FORMAT ili neka od eksplicitnih naredbi za deklarisanje vrste (INTEGER, REAL, DOUBLE PRECISION, COMPLEX, LOGICAL). Kasnije ćemo videti da ima još takvih naredbi, koje samo prenose prevodiocu izvesne informacije, a ne prouzrokuju nikakvu akciju u mašinskom programu.

4. Poslednja u ciklusu naredbe DO, dakle ona koja nosi broj n iz naredbe DO, ne sme biti nijedna od naredbi GO TO, aritmetička naredba IF, RETURN (o njoj će biti reči u Glavi 10), STOP, PAUSE ili druga naredba DO. Da se izbegnu takve situacije, postoji naredba CONTINUE, čiju ćemo upotrebu videti u primerima.

5. Pošto po definiciji nikada ne sadrži drugu naredbu IF ili naredbu DO, logička naredba IF može biti poslednja u ciklusu naredbe DO (opšti oblik te naredbe je: IF (t) s). Naredba s će se u ciklusu DO izvršavati samo kada je vrednost izraza t logička konstanta .TRUE..

6. Sa jednim izuzetkom, u ciklus naredbe DO ne sme se ući ni po kojoj naredbi koja nije i sama u okviru tog ciklusa. Tako se izričito zabranjuje upotreba naredbi GO TO ili IF za prelaz na neku naredbu u ciklusu DO, a da se pre toga ne izvrši sama naredba DO. Ovo pravilo takođe *zabranjuje* prelaz iz spoljnog u unutrašnji ciklus ali *dozvoljava* prelaz iz unutrašnjeg u spoljni ciklus. Posljednje je dozvoljeno jer se, sa gledišta spoljnog ciklusa, prelaz u celini vrši u njegovom sopstvenom okviru. Na slici 8.1 grafički su prikazani neki stavovi ovog pravila. Uglaste linije predstavljaju cikluse naredbi DO, a strelicama su označeni prelazi. Prelazi numerisani brojevima 2, 3 i 4 su dozvoljeni jer 2 i 3 idu iz



Slika 8.1 Ilustracija pravila 6 za pisanje naredbi DO

unutrašnjeg ciklusa u spoljne cikluse istog gnezda, a 4 je ceo u okviru istog ciklusa. Ostala tri prelaza, numerisana brojevima 1, 5 i 6, nepravilni su i nisu dozvoljeni jer dolaze u ciklus naredbe DO po nekoj naredbi izvan tog ciklusa.

7. Iz svakog ciklusa nekog gnezda dozvoljen je prelaz na neki potprogram našeg programa, pod uslovom da se povratnim prelazom vratimo opet u isti ciklus iz koga smo pošli. Ovo pravilo definiše onaj izuzetak, koji je pomenut na početku pravila 6. Ako prelaz na potprogram nazovemo »izlazom«, a ponovni prelaz iz potprograma u ciklus »ulazom«, tada ovo pravilo možemo iskazati i ovako: između »izlaza« i »ulaza« ne sme se nalaziti nijedna naredba DO niti takva koja je poslednja u nekom ciklusu DO našeg gnezda. Razume se, naredbama potprograma ne sme biti promenjena vrednost nijednog indeksnog parametra bilo koje naredbe DO u sklopu gnezda.

Naredba CONTINUE je jedna prividna ili pomoćna naredba, koju možemo pisati bilo gde u izvornom programu, a da se time ne poremeti normalni redosled izvršavanja. Ona postoji samo da bi se moglo uvek zadovoljiti pravilo 4, po kojem poslednja naredba u ciklusu DO ne sme biti jedna od naredbi za prelaz. Takođe se koristi i kao naredba na koju se prelazi naredbom IF kada je završen rad u ciklusu neke naredbe DO. To je potrebno zato što bi prelaz naredbom IF, iz okvira ciklusa, na naredbu DO na čelu ciklusa izazvao ponavljanje ciklusa *ispočetka* ($i = m_1$). Za obe upotrebe naredbe CONTINUE videćemo primere u sledećem odeljku.

8.4 *PRIMERI UPOTREBE NAREDBE DO*

Zbog neobično velike primene naredbe DO u FORTRAN-programiranju najraznovrsnijih zadataka, njenu upotrebu ćemo ilustrovati sa još nekoliko primera.

Neka su nam ulazni podaci za jedan program veličine koje dobijamo merenjem u eksperimentima. Svakim merenjem dobijamo po jednu vrednost za x i y, dakle dve koordinate jedne tačke u ravni. Koordinate na ulazu u računar nisu sređene po veličini, tj. poznato je samo koje veličine sačinjavaju parove koordinata, ali to ne znači da je prva vrednost za x ili y manja od svih ostalih. Radi izračunavanja, koje će se vršiti kasnije u programu, potrebno je da se ove veličine u memoriji računara srede tako, da prva

vrednost x-a bude najmanja, sledeća po veličini itd., drugim rečima, moramo parove koordinata srediti po rastućoj vrednosti x-a.

Pretpostavićemo da su vrednosti x-a, onim redom kako ulaze (tj. još nesređene) članovi jedne jednodimenzionalne matrice, čiji je simbol X, i da ih ima 25. Isto tako, vrednosti y-a su članovi jednodimenzionalne matrice Y, i njih takođe ima 25.

U FORTRAN-programu, koji će preurediti te podatke da budu u rastućem redosledu po vrednostima x-a, potrebno je jedno gnezdo od dve naredbe DO. Mi ćemo, međutim, za početak ceo zadatak uprostiti na taj način što ćemo najpre napisati samo verziju unutrašnjeg ciklusa DO. Taj unutrašnji ciklus smešta najmanju vrednost x-a na mesto prvog člana matrice i to na sledeći način: najpre se upoređuju prve dve vrednosti x-a originalne matrice; ako je prva manja ili jednaka drugoj, one ostaju na svojim mestima; ako je pak prva veća od druge, tada njih dve zamenjuju mesta u okviru matrice. Pošto se uporede prva dva člana i po potrebi promene mesta, upoređuje se prvi član sa trećim, pa se opet ili ostavljaju na svojim mestima ili, ako je prvi veći, međusobno zamenjuju. Može se desiti da je taj »prvi« član malopre bio na drugom mestu, no to ništa ne smeta. Na isti način, nastavlja se upoređivanje prvog sa četvrtim, prvog sa petim itd., sve dok se član na prvom mestu ne uporedi sa svima ostalim, zamenjujući ih pri tome ako je potrebno. Ovim postupkom obezbeđuje se da na kraju najmanja vrednost x-a bude na mestu prvog člana matrice X. Ne zaboravljajući da svakom x-u odgovara jedan y, istovremeno sa zamenjivanjem članova matrice X zamenjuju mesta i članovi matrice Y, samo bez upoređivanja njihovih vrednosti.

Da bi dva člana jedne matrice zamenila mesta u memoriji, obavlja se sledeći postupak od tri koraka: (1) prva veličina se premešta na jedno privremeno mesto u memoriji, kojem dajemo simbol PRIVR, (2) druga veličina se premešta na mesto koje je prvobitno zauzimala prva, (3) prva veličina, koja je sada vrednost promenljive PRIVR, premešta se na ono mesto gde je pre toga bila druga. Evo kako bi izgledao taj deo programa, bez naredbe za ulaz:

```
DO 15 K = 2, 25
IF (X(1).LE.X(K)) GO TO 15
PRIVR = X(1)
X(1) = X(K)
X(K) = PRIVR
PRIVR = Y(1)
```

```
      Y(1) = Y(K)
      Y(K) = PRIVR
   15 CONTINUE
```

Ovdje je pretpostavljeno da su podaci već u memoriji računara, uneti u matrice X i Y prethodnim delom programa. Takođe, prikazano je samo nalaženje najmanje vrednosti za član x_1 (i odgovarajuće premeštanje njegovog para, y_1), a ne i za ostale članove matrice X.

U napisanom delu programa treba zapaziti dve stvari. Najpre, ovde vidimo još jedan primer naredbe DO, čiji ciklus ne počinje sa vrednošću 1 indeksa (vidi primer na kraju odeljka 8.2). Međutim, dozvoljeno je da početna vrednost indeksa bude i jednaka, pa čak i veća od krajnje vrednosti ($m_1 \geqslant m_2$); u oba slučaja *ceo ciklus naredbi izvršiće se jedanput,* razume se ako u ciklusu nema naredbe koja bi ga prekinula.

Zatim, ovde imamo jedan primer upotrebe naredbe CONTINUE. Ako je izraz u zagradama naredbe IF logički tačan (.TRUE.), tada treba da se zaobiđu sve naredbe koje zamenjuju mesta vrednostima x-a i y-a, a da se ipak akcija vrati na naredbu DO radi ponavljanja ciklusa sa povećanim indeksom. Rekli smo da se to ne sme učiniti direktnim prelazom iz sredine ciklusa na naredbu DO, jer bi u tom slučaju indeks opet dobio vrednost m_1, a ne sledeću, veću za m_3 od prethodne. Jedini način da se obezbedi normalno ponavljanje ciklusa u ovom primeru jeste da se napiše naredba CONTINUE, kao što je i učinjeno. Ovakav slučaj je izričito pomenut u pravilu 4 za korišćenje naredbe DO.

U zadatku koji rešavamo ovim programom može se desiti da postoje i dve ili više jednakih vrednosti x-a. To je programom predviđeno upotrebom operatora »manje ili *jednako*«. Tako, ako su vrednosti jednake, one se neće zamenjivati nego će prelazom na naredbu CONTINUE početi novo izvršavanje ciklusa.

Kada se završe ponavljanja ciklusa, tj. kada budu ispunjeni uslovi koje postavlja naredba DO, sigurni smo da su podaci poređani tako da se najmanja vrednost x-a nalazi na mestu prvog člana matrice X, a istovremeno njoj odgovarajuća vrednost y-a na mestu prvog člana matrice Y. Posle toga treba da nađemo način da na mesto drugog člana matrice X dođe sledeća po veličini vrednost x-a. To se postiže upoređivanjem drugog člana sa trećim i svima ostalima redom, zamenjujući ih kad god je potrebno. Isto to treba zatim učiniti za treći član matrice, pa za četvrti, peti itd., sve dok

se sve vrednosti x-a ne svrstaju u okviru matrice u rastućem redosledu.

Vidimo da za to treba da definišemo kao promenljive sve indekse koji se menjaju sa intervalom $m_3 = 1$. Tada će sam indeks birati član koji treba da se upoređuje sa svima narednim članovima, a vršiće se i zamenjivanje gde bude potrebno. Jedan indeks, za koji ćemo usvojiti simbol J, počinjaće sa vrednošću 1 i ići do 24. Drugi indeks, za koji smo već napred usvojili simbol K, definisaćemo sada tako da počinje sa vrednošću za 1 većom od bilo koje vrednosti, koju ima indeks J i da ide do 25. Ako se setimo definicije naredbe DO, tamo stoji da svi indeksni parametri (m_1, m_2, m_3) moraju biti ili celobrojne konstante ili celobrojne skalarne promenljive. Ovakva naredba DO, na primer, nije dozvoljena:

```
DO 15 K = J + 1
```

Da ne bismo narušili to pravilo, jednostavno dodajemo još jednu naredbu, kojom se izračunava vrednost nove promenljive I = J + 1, koja je dakle uvek za 1 veća od svake vrednosti indeksa J. Potpun deo programa koji uređuje empirijske podatke na traženi način, izgledao bi tada ovako:

```
      DO 15 J = 1, 24
      I = J + 1
      DO 15 K = I, 25
      IF (X(J).LE.X(K)) GO TO 15
      PRIVR = X(J)
      X(J) = X(K)
      X(K) = PRIVR
      PRIVR = Y(J)
      Y(J) = Y(K)
      Y(K) = PRIVR
   15 CONTINUE
```

Da ilustrujemo nešto drukčiji tip programskih ciklusa, učinićemo neke nove pretpostavke u vezi sa prethodnim zadatkom. Uzmimo da tačke, čije smo kordinate uredili u rastućem redosledu apscisa, leže sve na jednoj krivoj i da se traži da izračunamo još i površinu, koju ograničava ta kriva sa dvema krajnjim ordinatama i apscisnom osom; to znači, treba naći određeni integral te krive u datim granicama. Postoji više načina numeričke integracije; mi ćemo ovde napisati programe za integriranje po poznatom trapeznom i Simpsonovom pravilu. Ako pretpostavimo da je interval

između apscisa susednih tačaka konstantan i jednak h, tada je približna vrednost integrala po trapeznom pravilu:

$$s = \frac{h}{2} (y_1 + 2y_2 + 2y_3 + \ldots + 2y_{23} + 2y_{24} + y_{25})$$

Za formiranje zbira ordinata, čiji su indeksi od 2 do 24, može se upotrebiti ciklus jedne naredbe DO. Kada dobijemo taj zbir, možemo ga pomnožiti sa 2, dodati vrednost prve i poslednje ordinate i sve zajedno pomnožiti sa $h/2$. Taj deo programa, koji vidimo niže, logički dolazi u nastavku ranije napisanog dela, posle naredbe CONTINUE.

```
      ZBIR = 0.0
      DO 20 I = 2, 24
   20 ZBIR = ZBIR + Y(I)
      S=(X(2)—X(1))/2.0*(Y(1)+2.0*ZBIR+Y(25))
```

U program je uključeno i izračunavanje intervala h, na bazi pretpostavke da je taj interval jednak razlici bilo kojih dveju susednih apscisa. Ako interval nije konstantan, program neće dati tačan rezultat. Kada bi se zahtevalo da program radi i sa nejednakim intervalima između apscisa tačaka, tada bi trebalo primeniti neki drugi metod numeričkog integriranja.

Po Simpsonovom pravilu rezultat integriranja je bliži tačnoj vrednosti integrala funkcije, koja je data koordinatama svojih tačaka na sledeći način:

$$s = \frac{h}{3} (y_1 + 4y_2 + 2y_3 + 4y_4 + 2y_5 + \ldots + 2y_{23} + 4y_{24} + y_{25})$$

Program za računanje po ovom obrascu nešto je složeniji, zbog naizmeničnog javljanja koeficijenata 2 i 4. Jedan očigledan način za rešenje tog problema jeste pisanje dvaju posebnih ciklusa DO, od kojih se u svakom posebno kumuliraju zbirovi ordinata, prema koeficijentima. Evo jednog takvog rešenja:

```
      CNPR = 0.0
      CPAR = 0.0
      DO 51 I = 2, 24, 2
   51 CPAR = CPAR + Y(I)
      DO 52 I = 3, 23, 2
   52 CNPR = CNPR + Y(I)
      S = (X(2)—X(1))/3.0*(Y(1)+4.0*CPAR+2.0*CNPR+Y(25))
```

CPAR je simbol za zbir parnih ordinata, CNPR simbol za zbir svih neparnih ordinata osim prve i poslednje.

Ako se postupi na sledeći način, računanje se može programirati i sa samo jednom naredbom DO, čime se ušteđuje nešto vremena u izvršavanju mašinskog programa. Definišimo jedan indeks, čija vrednost ide od 1 do 11. Dvostruka vrednost tog indeksa uvek će biti paran broj, pa otuda i indeks onih ordinata koje treba množiti sa 4. Ta dvostruka vrednost, uvećana za jedinicu, biće uvek neparan broj, dakle indeks onih ordinata koje se množe sa 2. Program napisan na taj način, izgledao bi:

```
      ZBIR = 0.0
      DO 66 I = 1, 11
   66 ZBIR = ZBIR + 4.0*Y(2*I) + 2.0*Y(2*I+1)
      S=(X(2)—X(1))/3.0*(Y(1)+ZBIR+4.0*Y(24)+Y(25))
```

U ovom primeru veoma je lepo došla do izražaja fleksibilnost upotrebe indeksnih promenljivih. Bez te osobine ne bi bilo moguće programirati potrebni zbir ordinata sa samo jednom naredbom DO.

Pada u oči da smo oba obrasca za numeričko integriranje pisali sa indeksima koji počinju od 1, dok je uobičajeno da indeksi počinju sa vrednošću 0. To je učinjeno namerno, da bi se jednostavnije mogao napisati program jer, kao što znamo, indeks naredbe DO mora biti pozitivna celobrojna veličina. To ne znači da se ne može programirati izračunavanje i po obrascima gde se javljaju i negativni indeksi i takvi čija je vrednost nula, no u tom slučaju treba uložiti nešto više truda nego što je to na ovom mestu opravdano. (Treba napomenuti da u FORTRAN-prevodiocima za neke tipove računara nema ovog ograničenja, tj. indeksi mogu biti i nula i negativni).

VEŽBE

U svim zadacima gde se radi o matricama, upotrebiti naredbu DIMENSION i naredbu DO.

* 1. Napisati program kojim se formira zbir 50 članova jednodimenzionalne matrice AX4. Zbiru dati simbol SUMAX4.

2. Jednodimenzionalne matrice A i B imaju po 30 članova. Izračunati:

$$D = \left(\sum_{i=1}^{30} (A_i - B_i)^2 \right)^{1/2}$$

3. Članovi dvodimenzionalne matrice AMTR raspoređeni su u 10 redova i 10 kolona. Jednodimenzionalna matrica DIAG ima 10 članova. Napisati naredbe za izračunavanje vrednosti članova matrice DIAG po obrascu:

$$DIAG(I) = AMTR(I,I) \qquad I = 1,2,\ldots,10$$

* 4. Jednodimenzionalna matrica M ima 20 članova, čije su vrednosti celi brojevi. Napisati naredbe sa kojima se vrednost svakog člana izračunava kao proizvod njegove prvobitne vrednosti i njegovog indeksa, tj. zameniti m_i sa $i \cdot m_i$, $i = 1,2,\ldots,20$.

* 5. Jednodimenzionalne matrice R i S imaju *maksimalno* po 40 članova. *Stvarni* broj članova dat je kao vrednost celobrojne promenljive M. Izračunati prvih M članova matrice T, koja takođe ima najviše 40 članova, po obrascu:

$$T(I) = R(I) + S(I) \qquad I = 1,2,\ldots,M$$

6. Jednodimenzionalne matrice A i B imaju maksimalno po 18 članova. N je celobrojna promenljiva, čija vrednost ne prelazi 18. Izračunati:

$$C = \sum_{k=1}^{N} A_k B_k$$

* 7. Jednodimenzionalna matrica F ima maksimalno 50 članova. Svaki od prvih M članova, izuzev prvog i M-tog, treba da bude zamenjen sa:

$$F_i = \frac{F_{i-1} + F_i + F_{i+1}}{3}$$

Ovo je primer postupka koji se primenjuje kod eksperimentalnih podataka, da se smanji uticaj slučajnih grešaka.

* 8. Jednodimenzionalna matrica B ima 50 članova, sa međusobno različitim vrednostima. Vrednost člana koji je algebarski najveći dodeliti promenljivoj BMAX, a indeks tog člana promenljivoj NBMAX.

9. Dve jednodimenzionalne matrice, X i Y, imaju po 50 članova. Vrednost promenljive XS jednaka je vrednosti jednog člana ma-

trice X (XS = X_i). Odgovarajuću vrednost Y_i treba dodeliti promenljivoj YS.

* 10. Članovi dvodimenzionalne matrice A raspoređeni su u 15 redova i 15 kolona. Jednodimenzionalna matrica X ima 15 članova. Izračunati vrednosti 15 članova jednodimenzionalne matrice B po obrascu:

$$B_i = \sum_{j=1}^{15} A_{ij}X_j$$

Ovo je množenje dvodimenzionalne i jednodimenzionalne matrice.

11. Tri dvodimenzionalne matrice, A, B i C, imaju dimenzije 15 × 15. Ako su date matrice A i B, vrednosti članova matrice C izračunati po obrascu:

$$C_{ij} = \sum_{k=1}^{15} A_{ik}B_{kj}$$

Ovo je množenje matrica.

* 12. Dvodimenzionalna matrica AMTR ima 20 redova i 20 kolona. Izračunati proizvod vrednosti svih članova na glavnoj dijagonali (član na glavnoj dijagonali ima isti indeks reda i kolone). Obrazac je:

$$DPR = \prod_{i=1}^{20} AMTR(I,I)$$

* 13. Izračunati vrednost promenljive Y po obrascu:

$$Y = 35.28\sqrt{1 + x^2} + x^{1/3}e^x$$

za sledeće vrednosti x-a: 1.00, 1.01, 1.02, ..., 3.00.

14. Po obrascu:

$$z = \frac{e^{ax} - e^{-ax}}{2} \sin(x+b) + a \cdot 1n \frac{b+x}{2}$$

izračunati vrednosti promenljive z za sve kombinacije vrednosti x, a i b u sledećim granicama:

x = 1.0, 1.01, 1.02, ..., 2.0
a = 0.10, 0.15, 0,20, ..., 0.80
b = 1.0, 2.0, 3.0, ..., 10.0

Za svaku kombinaciju x, a i b, kojih ukupno ima 1650, odštampati u jednom redu vrednosti x, a, b i z.

15. Naći ekonomično programsko rešenje za rešavanje sledećeg sistema linearnih jednačina:

$$
\begin{array}{ll}
a_{11}x_1 & = b_1 \\
a_{21}x_1 + a_{22}x_2 & = b_2 \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & = b_3 \\
\cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot\ \cdot & \\
a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + a_{n4}x_4 + \ldots + a_{nn}x_n & = b_n
\end{array}
$$

Radi uštede u prostoru u memoriji, koeficijente definisati kao članove jedne *jednodimenzionalne* matrice i napisati program za nalaženje rešenja. Kapacitet programa treba da bude 100 jednačina sa 100 nepoznatih, a stvarni broj jednačina dat je vrednošću promenljive N.

Glava 9

NAREDBE ZA ULAZ I IZLAZ

9.1 *UVOD*

Govoreći o FORTRAN-naredbama za ulaz i izlaz u Glavi 4, rekli smo da ćemo se za početak zadržati samo na operaciji ulaza pomoću bušenih kartica i na operaciji izlaza u obliku štampanih redova. To smo učinili zato što je čitaocu, radi savlađivanja ostalog gradiva, bio potreban jedan minimum znanja o tim funkcijama računara. Na taj način mogli smo posvetiti maksimalnu pažnju pravilima pisanja i upotrebe ostalih naredbi, izradi dijagrama toka i što boljem upoznavanju mogućnosti koje pruža naredba DO sa indeksnim promenljivima. Sada, međutim, možemo preći na izučavanje celokupne materije ulazno-izlaznih operacija u FORTRANu, što će nam dozvoliti da u programiranju tih operacija postignemo punu efikasnost koju pruža FORTRAN.

Naredbe za ulaz i izlaz u FORTRAN-programima omogućuju prenos podataka, u oba smera, između memorije računara, s jedne strane, i kartičnih jedinica, jedinica za papirnu traku, štampača, jedinica magnetne trake i magnetnog diska, s druge strane. Pri operacijama ove vrste, koje smo upotrebljavali u dosadašnjim primerima, videli smo da su podaci na spoljnim nosiocima grupisani u *datoteke,* koje se dele na *slogove.* Tipični slogovi, sa kojima smo već radili, jesu kartice i štampani redovi. Takve iste slogove, pri čemu je samo sadržaj ubeležen drugim kôdom, možemo zamisliti i na medijumima kao što je magnetna traka i magnetni disk.

Kao što smo videli u Glavi 4, za ulazno-izlazne operacije moraju se u izvornom programu dati sledeće informacije:

1. pisanjem reči READ odnosno WRITE određuje se najpre da li je to operacija ulaza ili izlaza; prvim brojem u zagra-

dama, koji smo nazvali pozivnom šifrom datoteke, posredno se definiše jedinica-nosilac ulazno-izlazne datoteke;

2. pomoću liste simbola određuje se koje će promenljive u programu dobiti nove vrednosti iz ulazne datoteke odnosno vrednosti kojih promenljivih će se upisati u izlaznu datoteku;
3. redosledom simbola u listi ujedno je određen i poredak prenošenja podataka;
4. naredbom FORMAT i njenim specifikacijama definiše se oblik u kome se podaci prenose sa ulaznog odnosno na izlazni medijum; broj naredbe FORMAT je drugi broj u zagradama naredbe READ odnosno WRITE.

U mnogim primerima do sada koristili smo te pojmove i pravila u najnužnijem obimu. U ovoj glavi ćemo uvesti proširenja, koja često uprošćavaju pisanje naredbi za ulazno-izlazne operacije, ali ćemo uvesti i nove naredbe i pravila, sa kojima se maksimalno koriste mogućnosti savremenog računara.

9.2 *LISTA SIMBOLA U ULAZNO-IZLAZNIM NAREDBAMA*

Najprostija lista simbola je takva u kojoj se eksplicitno navode sve promenljive i to onim redom kojim se prenose pripadajuće vrednosti sa odnosno na spoljni medijum. U odeljku o naredbi DIMENSION videli smo kakva olakšica postoji za pisanje liste ako se prenose cele matrice. Postoje i drugi, složeniji oblici liste, koje ćemo razmotriti u ovom odeljku. Svima njima je, međutim, zajednički princip istog redosleda *simbola* u listi, *polja* u slogu na spoljnom nosiocu i *specifikacije* oblika u naredbi FORMAT.

U primeru za najprostiji oblik liste

```
READ (3,18) A, X, F(1), F(2), M3
```

redosledom A, X, F(1), F(2), M3 definisano je da će vrednost iz prvog polja ulaznog sloga pripasti promenljivoj A, vrednost iz drugog polja promenljivoj X, vrednost iz trećeg polja indeksnoj promenljivoj F(1) itd. U naredbi FORMAT, koja nosi broj 18, prva specifikacija se odnosi na prvo polje ulaznog sloga i prvi simbol u listi, druga specifikacija na drugo polje i drugi simbol itd.

Prva novina kod liste simbola u ulazno-izlaznim naredbama veoma je korisna u programiranju, a ne komplikuje opisani proces: u listi je dozvoljena upotreba vrednosti celobrojnih promenljivih

u svojstvu indeksa za *naredne* indeksne promenljive iste liste. Na primer:

```
READ (3,25) I, J, A(I,J), K, L, B(K,L)
```

gde će vrednosti za indekse I i J biti pročitane *pre* vrednosti za indeksnu promenljivu A(I,J); isto važi i za indekse K i L i indeksnu promenljivu B(K,L). Praktičnu primenu ovog pravila imamo na primer kada treba čitati iz kartica vrednosti članova jedne dvodimenzionalne matrice, pri čemu kartice sa vrednostima nisu sređene po indeksima redova i kolona. Neka su te vrednosti bušene po jedna u jednu karticu i to tako što su najpre izbušene celobrojne vrednosti indeksâ, a zatim sama vrednost odgovarajućeg člana. Naredbe za ulaz mogle bi se napisati ovako:

```
      READ (3,33) I, J, A(I,J)
   33 FORMAT (2I5, E14.7)
```

Pročitane vrednosti za indekse I i J, odrediće mesto člana A(I,J) u matrici A. Ako bi kartice sa vrednostima članova bile sređene po kolonama matrice, dakle prema konvenciji FORTRANa, tada ne bi trebalo bušiti indekse, a naredbe bi se mogle napisati jednostavnije:

```
      READ (3,13) A
   13 FORMAT (E14.7)
```

Uvek kada se na ovaj način prenose vrednosti svih članova jedne matrice, oni se, kao što smo videli u odeljku 7.3, prenose tako da najbrže varira prvi indeks, a najsporije poslednji. Drugim rečima, računar u takvom slučaju smatra — ako se radi o dvodimenzionalnoj matrici — da su vrednosti sređene po kolonama; isto pravilo važi i za višedimenzionalne matrice.

Kada treba da se prenesu samo vrednosti određenih članova matrice, ovaj prostiji način se prema tome ne može koristiti. Međutim, ako u redosledu vrednosti na ulazu odnosno izlazu postoji bar delimična pravilnost, tada se u listi može upotrebiti takozvani *implicitni ciklus DO*. Ako na primer napišemo:

```
READ (3,13) (A(1,J), J = 1,10)
```

pročitaće se vrednosti iz prvih deset kartica i one će u računaru zauzeti mesta prvih deset članova *prvog reda* dvodimenzionalne matrice A. Sa parom naredbi:

```
      READ (3,11) (A(1,J), J = 1,10)
11    FORMAT (5E14.7)
```

čitale bi se dve kartice, svaka sa po pet vrednosti, i one bi u matrici zauzele mesta kao u prethodnom slučaju. Ako iste dve kartice čitamo po naredbi:

```
      READ (3,11) (A(I,1), I = 1,10)
```

tada će pročitane vrednosti zauzeti mesta prvih deset članova *prve kolone* matrice A.

Vidimo iz ovih primera da implicitni ciklus DO u listi ulazno-izlazne naredbe funkcioniše slično kao prava naredba DO u izvornom programu. Izborom indeksa u dva poslednja primera mogli smo, znači, da odstupimo od konvencije FORTRANa u pogledu redosleda članova matrice. Proširimo sada pravilo iz ta dva slučaja jednim ovakvim primerom: neka treba da se štampa 60 vrednosti članova jedne veće dvodimenzionalne matrice R i to tako da se u prvom redu štampa prvih 12 članova *prvog reda* matrice, u drugom redu prvih 12 članova *trećeg reda* matrice, u trećem redu prvih 12 članova *petog reda* matrice itd. Naredbe bi mogle biti sledeće:

```
      WRITE (5,66) ((R(I,J), J = 1,12), I = 1,13,2)
66    FORMAT (12E10.5)
```

Ovde vidimo gnezdo od dva implicitna ciklusa DO, čije nam je dejstvo poznato iz Glave 8. Indeks I će najpre dobiti vrednost 1, a J će izmenjati sve vrednosti od 1 do 12. Zatim će vrednost I postati 3, pa će opet J izmenjati sve vrednosti od 1 do 12, itd. Sa svakom novom vrednošću indeksa I izlaziće na spoljni medijum jedan novi red od 12 vrednosti (to je dejstvo naredbe FORMAT, o kojoj ćemo govoriti u narednom odeljku).

Implicitni ciklus DO iz liste u ovoj naredbi mogli bismo na sledeći način uporediti sa programskom ili eksplicitnom naredbom DO:

Implicitni ciklus DO	*Naredba DO*
↑ I = 1,13,2)	DO 28 I = 1, 13, 2 ↓
J = 1,12),	DO 28 J = 1, 12
((R(I,J),	28 *izlaz* R(I,J)

Treba dobro uočiti način pisanja liste sa implicitnim ciklusima DO, pojedinačnim i u gnezdu. Indeksna promenljiva se mora za-

tvoriti u zajedničke zagrade sa indeksnim parametrima, a unutrašnji ciklus u poseban par zagrada, odvojen zarezom od indeksnih parametara spoljnog ciklusa. Ove zagrade i zarezi su obavezan sastavni deo naredbe i ne mogu se izostavljati niti proizvoljno pisati.

Iz svih gornjih primera smo videli da poredak prenošenja vrednosti može da bude znatno složeniji nego po prostoj listi, gde je svaka promenljiva navedena posebno svojim simbolom. Zato se početnicima preporučuje da i za prenošenje celih matrica radije pišu listu sa implicitnim ciklusima DO nego da se oslanjaju na konvenciju o redosledu članova matrice, jer se tu najčešće greši. Ono što čitaocu mora biti jasno, jeste princip sinhronog »kretanja« kroz listu, polja u slogu i niz specifikacija u naredbi FORMAT. Taj princip možemo još jedanput kratko iskazati na sledeći način: prvi simbol — prvo polje — prva specifikacija, drugi simbol — drugo polje — druga specifikacija, itd. On ostaje u važnosti uvek, kao što ćemo videti u složenijim oblicima specifikacija u naredbi FORMAT.

9.3 *NAREDBA FORMAT*

Videli smo da se naredba FORMAT piše u vezi sa svakom naredbom za ulaz ili izlaz (osim izuzetaka, o kojima ćemo uskoro govoriti) i da utiče na redosled i oblik podataka, bilo da se oni prenose u memoriju sa spoljnog nosioca ili iz memorije na spoljni nosilac. Dalje, svakom podatku ili polju u okviru sloga mora uvek odgovarati jedna specifikacija u pripadajućoj naredbi FORMAT.

Opšti oblik naredbe FORMAT je:

n FORMAT (c_1, c_2, ..., c_n)

gde je n jedan broj, celobrojna konstanta dužine 1 do 5 cifara, čija je maksimalna vrednost 99999 (sem za neke FORTRAN-prevodioce, gde je manja — vidi tabelu 12.2), a c_1, c_2, ..., c_n su *specifikacije polja* ili prosto specifikacije.

Specifikacije se uvek sastoje od jednog karakterističnog slova i izvesnog broja celobrojnih konstanti, koje se pišu ispred ili iza tog karakterističnog slova. Neke specifikacije smo upoznali još u Glavi 4 i tamo smo videli još i izvesna dodatna pravila, koja ćemo sada sistematizovati, govoreći o raznim mogućim specifikacijama. Najpre ćemo izneti zajedničke osobine svih specifikacija, a zatim ono što posebno važi za svaku od njih.

Specifikacije sa karakterističnim slovima F, E i D imaju, pored jedne celobrojne konstante (neposredno iza karakterističnog slova), koja znači ukupan broj znakova u polju, još i drugu celobrojnu konstantu, koja pokazuje koliko ima decimalnih mesta.

Da ih ne bismo ponavljali kod svake specifikacije ponaosob, sledeće karakteristike su zajedničke za specifikacije I, F, E i D.

Kod *ulaznih* podataka, prvi znak u polju koji nije blanko mora biti algebarski znak, i to samo ako je vrednost podatka negativna. Pisanje i bušenje pozitivnog algebarskog znaka nije obavezno; ako ga nema, vrednost se smatra pozitivnom. Znaci blanko ispred, iza i između značajnih cifara u polju, smatraju se nulama; takođe, polje koje je u celini nebušeno (svi znaci blanko) predstavlja vrednost nula.

Kod *izlaznih* podataka, ako je specifikacijom predviđen veći broj znakova u polju nego što broj u memoriji ima cifara (i uopšte znakova), tada će znaci zauzeti mesta do desne granice polja, dok će ostatak pozicija do leve granice polja biti ispunjen znacima blanko. Ako broj pozicija, predviđen specifikacijom, nije dovoljan da primi sve cifre broja (i uopšte sve znake od kojih se sastoji veličina u memoriji), tada će sadržaj polja na spoljnom nosiocu biti različit za razne specifikacije. Prilikom izlaza na karticu ili štampani red, znak plus se obično ne štampa ispred pozitivnog broja.

Ispred svih specifikacija, izuzev H i X, može se pisati *faktor ponavljanja*, tj. broj koji pokazuje na koliko se uzastopnih polja u jednom ili više slogova odnosi data specifikacija.

Faktor razmere sa karakterističnim slovom P, čiju ulogu smo takođe videli na primerima u Glavi 4, može se pisati ispred faktora ponavljanja i karakterističnih slova specifikacija F, E i D; kada je napisan u sastavu jedne specifikacije, on se odnosi na sve *sledeće* specifikacije F, E i D do one u kojoj se javlja novi faktor razmere. Ako se želi da on deluje samo na jednu specifikaciju, u sledećoj se mora napisati faktor razmere nula.

U sledećem izlaganju, pored karakterističnog slova svake specifikacije data je u zagradama sažeto njena uloga. To je potrebno zato što je broj raznih specifikacija prilično veliki, pa se smisao svake od njih u početku teško pamti.

9.3.1 *SPECIFIKACIJA I (CEO BROJ)*

Oblik specifikacije je *Iw*. Slovo *I* znači pretvaranje internog oblika celog broja u spoljni oblik dekadnog celog broja ili obratno. Parametar *w* znači ukupan broj znakova u polju, uključivo eventualni algebarski znak i znake blanko. Pisanje decimalne tačke nije dozvoljeno.

Ako je broj znakova *w* u specifikaciji za izlaz manji od broja cifara koje se prenose, tada će algebarski znak i značajnije cifre biti izgubljeni, bez upozorenja. Kod FORTRAN-prevodilaca za neke računare*, u ovom slučaju će u svih *w* pozicija polja na spoljnom nosiocu biti upisane zvezdice, kao upozorenje da je polje nedovoljno da primi potpunu brojnu vrednost.

9.3.2 *SPECIFIKACIJA F (SPOLJNI OBLIK NEPOKRETNOG ZAREZA)*

Oblik specifikacije je *Fw.d*. Slovo F znači pretvaranje internog oblika realnog broja u spoljni oblik decimalnog broja sa nepokretnim zarezom, bez izložioca, i obratno. Parametar *w* znači ukupan broj znakova u polju, uključujući mesto za algebarski znak, decimalnu tačku i eventualne znake blanko. Parametar *d* znači broj decimalnih mesta.

Kod *ulaznih* podataka pisanje decimalne tačke nije obavezno, međutim ako se napiše, tada nema dejstva slovo *d* u specifikaciji, već je broj decimalnih mesta određen položajem tačke.

Kod *izlaznih* podataka uvek se pojavljuje decimalna tačka i desno od nje *d* decimalnih cifara. Ako je broj znakova *w* u specifikaciji za izlaz manji od broja cifara koje se prenose, tada će algebarski znak i značajnije cifre biti izgubljeni, bez upozorenja. Međutim, kod nekih FORTRAN-prevodilaca* će u tom slučaju u svih *w* pozicija polja na spoljnom nosiocu biti upisane zvezdice, kao upozorenje da je polje nedovoljno da primi potpunu vrednost.

Specifikacija F se može pisati sa faktorom razmere (*s*) i faktorom ponavljanja (*r*) u opštem obliku *sPrFw.d*. Dejstvo faktora razmere je:

$$\textit{broj u spoljnom obliku} = \textit{broj u unutrašnjem obliku} \cdot 10^{s}$$

---

* ovo važi kod svih FORTRAN-prevodilaca za računare IBM 360, izuzev za konfiguracije koje mogu da koriste samo BPS (Basic Programming Support)

Faktor razmere može biti pozitivan, nula ili negativan. Primenjuje se i za ulaz i za izlaz, mada za ulaz retko. Jedan primer upotrebe faktora razmere za izlaz jeste ako treba da se menjaju jedinice mere (na primer, dužine u metrima treba da se štampaju u kilometrima: faktor razmere je 3).

9.3.3 *SPECIFIKACIJA E (POKRETNI ZAREZ)*

Oblik specifikacije je *Ew.d*. Slovo E znači pretvaranje internog oblika realnog broja u pokretnom zarezu (normalizovani oblik pokretnog zareza) u spoljni oblik realnog broja sa izložiocem E, i obratno. Ukupan broj znakova je *w*, uključivo mesta za algebarski znak, decimalnu tačku i eventualne znake blanko. Slovo *d* znači broj decimalnih mesta mantise.

Kod *ulaznih* podataka pisanje decimalne tačke nije obavezno, ali ako se napiše, tada nema dejstva parametar *d* u specifikaciji. Opšti oblik izložioca u polju na spoljnom medijumu je E±ee, isto kao kod konstante sa izložiocem E u naredbama izvornog programa. Radi uštede pri bušenju u kartice, dozvoljeni su i sledeći oblici (primer izložioca za stepen 10^2): E+02, E 02, E02, E+2, E2, +2. Izložilac se mora bušiti do desne granice polja u kartici, jer znaci blanko znače nule. Na primer, sva sledeća polja će po specifikaciji E14.7 biti interno pretvorena u istu vrednost (imati u vidu da je bušena decimalna tačka »starija« od konstante *d* u specifikaciji):

```
+13896542E03
  13896542.E—4
  1389.6542E0
+0.13896542+4
```

Kod *izlaznih* podataka ovaj oblik je normalno ±0.XXXXXXXXE ±ee (s tim što će znaci plus obično biti zamenjeni sa znacima blanko), gde je broj decimala jednak konstanti *d* u specifikaciji, a *x* predstavlja decimalnu cifru.

U vezi sa ulaznim podacima, koji se prenose po specifikaciji E, faktor razmere (*s*) nema nikakvog uticaja. U vezi sa izlaznim podacima njegovo dejstvo je da se mantisa množi sa 10^s, a izložilac istovremeno smanjuje za *s*. Faktor razmere može biti pozitivan, nula ili negativan.

9.3.4 *SPECIFIKACIJA D (DVOSTRUKA TAČNOST)*

Oblik specifikacije je *Dw.d*. Interna vrednost broja mora da bude dvostruko tačna (sa približno dvostrukim brojem dekadnih cifara u mantisi). Izložilac se piše sa slovom D umesto sa E ali je u svakom drugom pogledu ova specifikacija analogna specifikaciji E.

Pogledajmo sada dva primera prenošenja numeričkih vrednosti prema gore izloženim specifikacijama. Vrste brojeva, koje predstavljaju pojedini simboli promenljivih, deklarisane su prema unutrašnjoj konvenciji FORTRANa; samo je D realna promenljiva dvostruke tačnosti. Neka se sledeći slog

—21.234bbb5bbb.2378b0.12345E+02b0.123456789123D—02

čita pomoću sledeće dve naredbe:

```
8 FORMAT(—2PF7.3,I4,F8.4,E12.5,D19.12)
  READ (5,8) A, N, B, C, D
```

Interne vrednosti brojeva će biti:

$A = -0.21234 \cdot 10^4$
$N = 5$
$B = 0.2378 \cdot 10^2$
$C = 0.12345 \cdot 10^2$
$D = 0.123456789123 \cdot 10^{-2}$

Ako sada te interne vrednosti odštampamo pomoću naredbi:

```
6 FORMAT (2PF9.1, I4, F7.1, E12.3, D19.10)
  WRITE (7,6) A, N, B, C, D
```

na štampaču ćemo dobiti sledeći red:

—212340.0bbb5b2378.0bb12.345Eb00bb12.3456789123D—04

(znak *b* znači blanko).

9.3.5 *SPECIFIKACIJA L (LOGIČKA VELIČINA)*

Oblik specifikacije je *Lw*. Slovo L znači pretvaranje unutrašnje logičke konstante .TRUE. odnosno .FALSE. u slovo T odnosno F na spoljnom medijumu, i obratno. Ukupan broj znakova u polju je jednak konstanti *w*.

Kod *ulaznih* podataka, prvi znak u polju, koji nije blanko, mora biti jedno od slova T ili F, što će interno biti pretvoreno u logičku konstantu .TRUE. odnosno .FALSE.. Svi ostali znaci u polju nemaju nikakvog uticaja na logičku vrednost podatka.

Kod *izlaznih* podataka, u krajnju desnu poziciju polja biće upisano slovo T ili F, a sve prethodne pozicije počev od prve zauzeće znaci blanko.

9.3.6 *SPECIFIKACIJA A (ALFANUMERIČKI PODATAK)*

Oblik specifikacije je *Aw*. Sadržaj polja, koje se prenosi po ovoj specifikaciji, je proizvoljan: mogu se koristiti svi znaci koji imaju svoje kodove za dati računar (slova, cifre, specijalni znaci i znak blanko). Slovo A znači prenošenje u memoriju, na mesto rezervisano za promenljivu iz liste, sadržaja polja dužine w znakova na spoljnom medijumu, i obratno.

Kod ove specifikacije igra ulogu i broj znakova koji staje u jednu mašinsku reč u memoriji računara. Taj broj je obično 4 ili 8; obeležićemo ga sa v.

Kod *ulaznih* podataka, ako je w veće od v, tada će na mesto rezervisano za promenljivu iz liste (čija je dužina v) biti iz polja u ulaznom slogu preneto $w—v$ desnih znakova tog polja; ako je w manje od v, tada će iz polja u ulaznom slogu biti preneto svih w znakova, a mašinska reč (v) će s desne strane biti dopunjena znacima blanko.

Kod *izlaznih* podataka, ako je w veće od v, u polje na spoljnom medijumu biće upisano svih v znakova i to do desne granice tog polja, a pozicije do leve granice biće ispunjene znacima blanko; ako je w manje od v, u polje na spoljnom medijumu biće preneto w levih znakova mašinske reči, a ostatak znakova neće biti prenet.

Izbor simbola za promenljivu, čija će se vrednost čitati kao ulazni podatak pomoću specifikacije A, podleže svim pravilima za deklarisanje vrste koja smo videli kod numeričkih veličina. Sa alfanumeričkim podacima mogu se obavljati sve operacije predviđene u FORTRANu, isto kao sa numeričkim podacima. Drugo je pitanje da li ima smisla, na primer, sabiranje alfanumeričke konstante ARAL sa realnom konstantom $0.286 \cdot 10^1$. Međutim, aritmetičke operacije u izrazima sastavljenim od samih alfanumeričkih podataka, mogu naći primenu u primerima kao što je sledeći.

U jednu jednodimenzionalnu matricu F unosi se po specifikaciji A neki alfanumerički tekst, dužine najviše 100 znakova, pri

čemu je svaki znak teksta bušen u posebnu karticu; poslednja kartica sadrži u polju od prve četiri kolone četiri zvezdice. Treba ispitati da li u tekstu postoji znak blanko, u kom slučaju se obavlja specijalni postupak, koji nije dat u prikazanom delu programa. Deo programa je napisan uporedo sa aritmetičkim i sa logičkim naredbama IF:

```
      DIMENSION F(100)
    1 FORMAT (A4)
      READ (5,1) BLANKO
      READ (5,1) ZVEZD
      DO 9 I = 1, 101
      READ (5,1) F(I)
      IF (F(I)—ZVEZD) 9, 8, 9
    9 CONTINUE
      STOP 666
    8 DO 7 J = 1, I
      IF (F(J)—BLANKO) 7, 6, 7
    7 CONTINUE
      GO TO 16
    6 . . . . . .
```

```
      DIMENSION F(100)
    1 FORMAT (A4)
      READ (5,1) BLANKO
      READ (5,1) ZVEZD
      DO 9 I = 1, 101
      READ (5,1) F(I)
    9 IF(F(I).EQ.ZVEZD) GO TO 8
      STOP 666
    8 DO 7 J = 1, I
    7 IF (F(J).EQ.BLANKO) GO TO 6
      GO TO 16
    6 . . . . . .
```

Za upoređivanje su prethodnim ulaznim naredbama pročitane dve kartice, jedna blanko i jedna sa četiri zvezdice, i to su alfanumeričke vrednosti dveju promenljivih, BLANKO i ZVEZD. Vidimo da se sa alfanumeričkim podacima, čiji simboli su ovde realne veličine, obavljaju aritmetičke i logičke operacije.

9.3.7 *SPECIFIKACIJA H (LITERAL)*

Oblik specifikacije je *wH* ili niz znakova između dva apostrofa, '*XX*.....*XX*'. Što se tiče sadržaja polja na spoljnim medijumima, on može biti sastavljen od svih znakova bez izuzetka, isto kao za polja čiji se sadržaj prenosi po specifikaciji A. Razlika između ove specifikacije i svih ostalih je u tome što specifikacija H nije vezana ni za jedan simbol u listi ulazno-izlazne naredbe. Takav podatak, koji se zove *literal*, izlazi u polje na spoljnom nosiocu direktno iz naredbe FORMAT odnosno ulazi iz polja na spoljnom nosiocu na mesto postojećeg literala u naredbi FORMAT. Ispred specifikacije sa apostrofima može se upotrebiti faktor ponavljanja.

Kod *ulaznih* podataka, ako je oblik specifikacije *wH*, sadržaj *w* znakova polja ulazi na mesto postojećeg literala u naredbi FOR-

MAT. Ako je oblik specifikacije '*XX.....XX*', gde je X proizvoljan znak, u pozicije između apostrofa u naredbi FORMAT ulazi iz polja na spoljnom nosiocu onoliko znakova koliko ima pozicija.

Kod *izlaznih* podataka, ako je oblik specifikacije *wH*, na spoljni medijum će biti preneto *w* znakova literala; ako je oblik specifikacije '*XX.....XX*', na spoljni medijum biće preneti svi znaci koji se nalaze između apostrofa.

Specifikacija literala, bilo sa slovom H ili sa apostrofima, najčešće se upotrebljava za izlaz naslova i drugih tekstova potrebnih za štampanje. Druga, vrlo česta primena je za upravljanje sa proredima na hartiji prilikom štampanja. U svim slogovima koji će se štampati, prvi znak se ne štampa nego služi za prorede i skokove hartije u štampaču, prema sledećoj tabeli:

prvi znak u slogu	*dejstvo*
blanko	običan prored
0	dvostruki prored
1	skok na prvi red nove strane
+	bez proreda*

9.3.8 *SPECIFIKACIJA X (BLANKO)*

Oblik specifikacije je *wX*. Na *ulazu*, dejstvo specifikacije je da se preskače *w* znakova sloga na spoljnom nosiocu. Razlog za upotrebu ove specifikacije može biti taj da se polje u slogu sastoji od dužeg niza znakova blanko, pa ih treba preskočiti, da se u sledećoj specifikaciji iste naredbe FORMAT ne bi morali uzimati u obzir i ti znaci blanko. Ili, sadržaj izvesnog polja u slogu nije potreban u programu, na primer identifikacioni brojevi kartica i tsl., pa to polje treba preskočiti.

Na *izlazu*, ovom specifikacijom se ubacuju znaci blanko u slog na spoljnom nosiocu. Ovo se obično praktikuje kada treba da se rasporedi tekst u zaglavlju neke tabele odnosno da se dobiju veći razmaci između kolona sa brojevima.

---

* prored je pomeranje hartije u položaj za štampanje novog reda, pa će znak 0 dati jedan blanko red između dva štampana reda, dok će znak + značiti štampanje preko već odštampanog reda.

9.3.9 *SPECIFIKACIJA T (POČETNA POZICIJA)*

Oblik specifikacije je *Tw*. Ova specifikacija određuje poziciju u slogu (konstanta *w*), od koje počinje prenos podataka sa spoljnog nosioca ili na spoljni nosilac. Ona se odnosi na prvu susednu specifikaciju u istoj naredbi FORMAT, koja može biti bilo koja od ostalih specifikacija. Ako u jednoj naredbi FORMAT ima više specifikacija T, koje se odnose na razna polja istog sloga, tada sve konstante *w* ne moraju biti u rastućem redosledu s leva na desno (vidi sledeći primer).

Kod *izlaza* sloga koji će se štampati, bilo direktno iz memorije ili indirektno sa nekog drugog spoljnog nosioca, u konstantu *w* uključen je i prvi znak, koji se ne štampa nego upravlja proredom (vidi 9.3.7). Na taj način, stvarna pozicija od koje počinje štampanje jeste *w—1*. Na primer, sa sledećim naredbama štampa se zaglavlje jedne tabele:

```
5. FORMAT (T40,'GODIŠNJI IZVEŠTAJ 1969', T80,
   1'DECEMBAR', T1, '0ART. BR. 10095')
   WRITE (6,5)
```

Raspored teksta u štampanom redu biće:

```
1. pozicija      39. pozicija                  79. pozicija
↓ u redu         ↓ u redu                      ↓ u redu
ART. BR. 10095   GODIŠNJI IZVEŠTAJ 1969        DECEMBAR
```

U tekst (literal) prvog polja u štampanom redu uključen je kao prvi znak cifra 0, koja prema tabeli u 9.3.7 znači dvostruki prored u *odnosu na prethodno štampani red*. Sam taj znak (cifra 0) nije štampan, kao što se vidi u gornjem primeru.

Naredba WRITE nema listu, pošto se ne prenose vrednosti nijedne promenljive nego samo literal iz naredbe FORMAT.

Kod *ulaza*, neka je dat sledeći slog:

```
                    14. kolona 25. kolona
                          ↓          ↓
b128637—18.26b1233b14.58b50.29
```

koji treba da se pročita sa sledećim naredbama:

```
1 FORMAT (T14, I5, T25, F6.2)
  READ (5,1) J, C
```

Sadržaj polja koje počinje sa 14. kolonom daje internu vrednost promenljivoj J = 1233; sadržaj polja koje počinje sa 25. kolonom daje internu vrednost promenljivoj C = $0.5029 \cdot 10^2$; ostala polja se ignorišu.

9.3.10 *SPECIFIKACIJA G (UNIVERZALNA)*

Oblik specifikacije *Gw.s* za sve realne veličine odnosno *Gw* za celobrojne i logičke veličine. Ova specifikacija je univerzalna, za prenos svih vrsta podataka. Konstanta *w* je ceo broj, bez algebarskog znaka, i znači ukupan broj znakova u polju. Konstanta *s* je takođe ceo broj, bez algebarskog znaka, i znači broj *značajnih* cifara u polju.

Za prenos realnog broja, u konstantu *w* se moraju uključiti pozicije za algebarski znak, cifre mantise i četiri pozicije za izložilac u obliku E±ee ili D±ee, gde *ee* znači dvocifreni ceo broj. Ako izložilac nije potreban, njegove četiri pozicije su znaci blanko. Značenje izložioca smo videli u 9.3.3 i u primeru na kraju 9.3.4.

U sledeća dva primera, L je deklarisano kao logička promenljiva, a ostali simboli prema unutrašnjoj konvenciji FORTRANa.

Na *ulazu* dat je sledeći slog:

b123.567bbbb—98.765bbbbb2.131E—10b8642b753bbT

koji treba da se prenese (pročita) po sledećim naredbama:

```
9 FORMAT (G12.6, G11.5, G10.4, G5, G4, G3)
  READ (5,9) A, B, C, I, J, L
```

Posle izvršavanja tih naredbi, interne vrednosti promenljivih su: A = $0.123567 \cdot 10^3$, B = $-0.98765 \cdot 10^2$, C = $0.2131 \cdot 10^{-11}$, I = 8642, J = 753, L = .TRUE..

Na *izlaz* sada treba preneti gornje interne vrednosti promenljivih A, B, C, I, J i L, prema sledećim naredbama:

```
8 FORMAT (G12.4, G10.3, G10.4, G5, G2, G5)
  WRITE (7,8) A, B, C, I, J, L
```

Posle izvršavanja ovih naredbi, na izlaznom nosiocu dobija se sledeći slog:

bbb123.5bbbbb—98.7bbbbb.2131E—11b864253bbbbT

Vidimo da vrednost promenljive J nije preneta u celini: zato što je specifikacija G2 kraća za jedan znak, prenete su samo desne dve cifre potpune vrednosti promenljive J = 753.

9.4 *UPOTREBA NAREDBE FORMAT*

Naredba FORMAT se upotrebljava za određivanje oblika i rasporeda sadržaja koji se prenosi pomoću ulazno-izlaznih naredbi. Do sada smo videli prostije primere njene upotrebe, kao i sve specifikacije oblika podataka koje se mogu javiti. Sada ćemo, međutim, razmotriti složenije oblike pisanja specifikacija uz upotrebu zagrada i kosih crta, čime se omogućuje prenošenje *više slogova* pomoću samo jednog para naredbi (ulazno-izlazna naredba i naredba FORMAT).

Parom naredbi

```
      READ (3,6) A, B, (C(I), I = 1,5)
    6 FORMAT (8F10.0)
```

prenose se vrednosti za sedam promenljivih iz sedam polja jednog ulaznog sloga, svaka po specifikaciji F10.0; osmi put specifikacija F10.0 neće se koristiti jer je lista u naredbi READ već iscrpljena. Odavde vidimo da broj prenetih vrednosti diktira lista ulazno-izlazne naredbe, a ne broj specifikacija u naredbi FORMAT.

S druge strane, ako imamo naredbe:

```
      READ (3,8) A, B, (C(I), I = 1,5)
    8 FORMAT (E14.7)
```

tada, posle prenošenja vrednosti za promenljivu A, u naredbi FORMAT više nema specifikacija za ostale promenljive. Zato se, od desne zagrade FORMATa, akcija vraća na početak i koristi se ponovo ista specifikacija. To će se ponoviti onoliko puta koliko je potrebno, sve dok se ne iscrpi lista u naredbi READ. Međutim, svako vraćanje od desne zagrade na početak istovremeno znači i ulaz novog sloga. Drugim rečima, gornji par naredbi pretpostavlja na ulazu sedam slogova, svaki sa po jednim poljem od 14 znakova.

Isto tako kao što se može izazvati ponavljanje jedne specifikacije pisanjem faktora ponavljanja ispred nje, može se programirati i ponavljanje cele jedne *grupe* specifikacija. Ta grupa se zatvara u zagrade i željeni faktor ponavljanja se piše ispred leve zagrade. Na primer, ako se na osam polja u jednom slogu naizmenično odnose

specifikacije I2 i F10.0, to ćemo napisati 4(I2,F10.0). Ovo nije isto što i 4I2, 4F10.0, jer to bi značilo da su u slogu najpre četiri polja oblika I2, pa za njima četiri polja oblika F10.0.

Kod većine FORTRAN-prevodilaca dozvoljen je samo jedan *nivo* unutrašnjih zagrada (spoljne su one koje obuhvataju sve specifikacije naredbe FORMAT), dok su kod FORTRAN-prevodilaca za veće računare* dozvoljena dva nivoa, dakle zagrade u zagradama.

Kada lista jedne ulazno-izlazne naredbe služi za prenošenje više od jednog sloga, pri čemu razni slogovi imaju različite oblike, tada se za odvajanje grupa specifikacija za pojedine slogove koristi kosa crta (/). Na primer, uzmimo da pomoću jedne naredbe READ treba da se čitaju dve kartice; u prvoj kartici se nalazi samo jedan četvorocifreni ceo broj, a u drugoj šest realnih brojeva u obliku sa izložiocem E. Naredbu FORMAT bi trebalo napisati:

```
FORMAT (I4/6E14.7)
```

Može se programirati i poseban oblik za jedan ili više slogova na početku, a drugi oblik za sve ostale slogove. To se postiže ako se grupa specifikacija za ostale slogove zatvori u poseban par zagrada. Na primer, ako prva kartica jednog paketa sadrži jedan ceo i jedan realan broj, a sve ostale kartice po dva cela i po jedan realan broj, tada naredbu FORMAT možemo pisati:

```
FORMAT (I4, E14.7/(2I4, E14.7))
```

Posle ulaza sadržaja prve kartice po specifikacijama I4, E14.7, sadržaji svih ostalih kartica se prenose po specifikacijama 2I4, E14.7. Prenošenjem sadržaja druge kartice došli smo do desne zagrade naredbe FORMAT; prema ranije rečenom, specifikacije se ponavljaju od početka, dakle od leve zagrade ali to ovde treba shvatiti *od prve leve zagrade idući od kraja unatrag.* Ako su za računar dozvoljena dva nivoa unutrašnjih zagrada, tada ovaj deo pravila o ponavljanju specifikacija glasi *od poslednje leve zagrade prvog nivoa.*

Pisanjem uzastopnih kosih crta u naredbi FORMAT, na izlazu se upisuju blanko slogovi, a slogovi na ulazu se preskaču. Ako na početku ili na kraju specifikacionog dela naredbe FORMAT ima n uzastopnih kosih crta, biće preskočeno n ulaznih slogova odnosno

---

* npr. IBM 360, prevodilac H, gde je drugi nivo zagrada rezervisan za specifikacije kompleksnih veličina, npr. (E8.1,E8.1)

upisano n blanko izlaznih slogova. Ako se pak n uzastopnih kosih crta nalazi na bilo kom drugom mestu u okviru naredbe FORMAT, broj preskočenih odnosno blanko slogova biće n-1. Na primer, po naredbama

```
10 FORMAT (///I6)
   READ (3,10) MULT
```

biće preskočena tri sloga iz ulazne datoteke sa pozivnom šifrom 3 pre nego što se iz četvrtog sloga prenese vrednost za promenljivu MULT. Sledeće naredbe, gde je ' ' znak blanko među apostrofima, koji daje običan prored pri štampanju ovih slogova (vidi 9.3.7):

```
15 FORMAT (' ', I5////' ', F5.2, I2//)
   WRITE (5,15) K, A, J
```

daju prilikom štampanja sledeće oblike izlaznih slogova:

```
ceo broj
red znakova blanko
red znakova blanko
red znakova blanko
realni broj, ceo broj
red znakova blanko
red znakova blanko
```

Prilikom pisanja specifikacija u naredbi FORMAT, mora se voditi računa o tome da se slažu vrsta specifikacije, oblik podatka na spoljnom nosiocu i vrsta promenljive u listi ulazno-izlazne naredbe. Takođe, važno je da se ima u vidu odakle se podaci uzimaju i gde se upućuju, tj. koji spoljni medijumi su u pitanju. Na primer, ako neki slog treba da se buši u karticu, zbir svih pozicija u odgovarajućim specifikacijama ne sme da bude veći od 80. Kada se radi o ulaznim slogovima, naredbom FORMAT se ne sme definisati veća dužina sloga od one koju stvarno imaju slogovi u ulaznoj datoteci.

Uzmimo sada kao primer da treba da se štampa jedna stranica rezultata proračuna ubrzanja, sa naslovom i identifikacionim oznakama za vrednosti u kolonama, prema ilustraciji na slici 9.1.

```
PRORACUN UBRZANJA
-----------------------------------------

X=    4.9143060E-02  Y=   -6.1243300E 05
X=    6.1462200E-02  Y=   -9.4016230E 05
X=    8.9001660E-02  Y=   -2.6033840E 06
X=    1.1297320E-01  Y=   -5.5610330E 07
X=    5.0163280E-01  Y=   -9.8632140E 07
X=    8.6489960E-01  Y=   -4.1126810E 08
```

Slika 9.1 Štampani izlaz (za poslednji primer iz odeljka 9.4)

Tu vidimo da između naslova i prvog reda sa podacima postoji jedan red crtica, sa kojima je naslov podvučen, i dva blanko reda. Sam naslov, red crtica i identifikacione oznake $X=$ i $Y=$ dobijaju se kao literali iz naredbi FORMAT. Vrednosti x—a i y—a su članovi dveju jednodimenzionalnih matrica, od kojih svaka ima šest članova. Ovde vidimo naredbe kojima je programirano štampanje tačno prema ilustraciji na slici 9.1:

```
      WRITE (5,100)
  100 FORMAT (' PRORAČUN UBRZANJA'/' ', 40'—'//)
      WRITE (5,101) (X(I), Y(I), I = 1,6)
  101 FORMAT (' X=', 1PE16.7, ' Y=', E16.7)
```

Prva naredba WRITE je bez liste jer se pomoću nje štampaju samo literali (naslov i crtice za podvlačenje). Ta naredba štampaće osim toga još i dva blanko reda. U drugoj naredbi WRITE upotrebljen je jedan implicitni ciklus DO, koji se ponavlja šest puta i svaki put daje jedan par vrednosti x i y. U obema naredbama FORMAT, prvi literal sadrži na početku jedan znak blanko koji se ne štampa nego samo izaziva jedan prored pre štampanja reda. Drugi literal u prvoj naredbi FORMAT (' ') kazuje da u sledećem redu ispod naslova nije potreban nikakav razmak nego da se tu odmah štampa četrdeset crtica za podvlačenje naslova (40'—'); sledeće dve kose crte izazivaju dva blanko reda. Prvi znak teksta naslova kao i identifikaciona oznaka X štampaju se u drugoj poziciji njihovih redova; red crtica, međutim, počinje od prve pozicije.

Kod odštampanih kolona brojeva treba zapaziti dve stvari. Prvo, obe vrednosti, x i y, štampane su po specifikaciji 1PE16.7,

iako je specifikacija za y bila E16.7 (vidi pretposlednji pasus ispred 9.3.1 i poslednji pasus 9.3.3). Treba još reći da se dejstvo faktora razmere prenosi na sledeće specifikacije iste naredbe FORMAT čak i kada su one obuhvaćene drugim parom unutrašnjih zagrada; to dejstvo se može zaustaviti jedino pisanjem faktora razmere nula.

Drugo, računar na kojem je prevođen ovaj program, interno predstavlja realne brojeve (obične tačnosti) sa sedam značajnih dekadnih cifara. Pošto je specifikacijom 1PE16.7 jedna od značajnih cifara preneta ispred decimalne tačke, mantisa je do traženog broja decimala dopunjena s desna sa nulom.

9.5 *PROMENLJIVE FORMAT-SPECIFIKACIJE*

Posle prevođenja izvornog FORTRAN-programa, svi podaci koje će naš mašinski program čitati sa ulaznih medijuma ili upisivati na izlazne medijume mogu imati isključivo samo one oblike koji su određeni specifikacijama u naredbama FORMAT. Kod nekih računara* mogu se, međutim, specifikacije menjati od jedne do druge upotrebe mašinskog programa, a da se pre toga ne moraju promeniti specifikacije u izvornom programu niti se program mora ponovo prevoditi.

Takve, promenljive FORMAT-specifikacije čitaju se sa spoljnih nosilaca kao i drugi ulazni podaci, u toku izvršavanja mašinskog programa. Specifikacije se prenose na jedan ulazni medijum, obično na kartice, i unose se kao alfanumerički podaci u jednu matricu. U ulazno-izlaznim naredbama izvornog programa, umesto broja naredbe FORMAT, navodi se simbol te matrice. Kao primer, uzmimo sledeću grupu specifikacija:

```
(2I4, F7.2, E14.7)
```

Tu grupu bušimo kao podatke u jednu karticu, tako da prvi znak (leva zagrada) bude u prvoj koloni. U izvornom programu treba da napišemo sledeće naredbe, sa kojima se rezerviše mesto za matricu i u nju unose znaci grupe specifikacija:

---

* vidi tabelu 12.2

```
      DIMENSION F(8)
    1 FORMAT (8A4)
      READ (5,1) (F(I), I = 1,8)
```

Posle izvršavanja ovih naredbi, sadržaj pojedinih članova matrice F biće:

```
F(1)   (2I4
F(2)   ,bF7
F(3)   .2,b
F(4)   E14.
F(5)   7)bb
F(6)   bbbb
F(7)   bbbb
F(8)   bbbb
```

Ulazno-izlazne naredbe u izvornom programu, koje prenose podatke po ovim specifikacijama, mogu se pisati iza ovih naredbi, na primer:

```
READ (5,F) I, J, A, B ili
WRITE (7,F) I, J, A, B
```

Za drugi primer neka je grupa promenljivih specifikacija sledeća:

```
(1H , 5I3//1H , 2(F6.1, I2))
```

a naredbe u izvornom programu:

```
      DIMENSION SPEC(18), NUM(10)
    3 FORMAT (18A4)
      READ (5,3) (SPEC(I), I = 1,18)
      . . . . . .
      WRITE (7, SPEC) (NUM(I) I = 1,5), A, K, B, J
```

Po naredbi READ čitaju se znaci gore navedenog sloga kao alfanumerički podaci, u grupama po četiri znaka, i smeštaju se u matricu SPEC. Po naredbi WRITE upisuju se podaci na spoljni medijum, prema speficikacijama iz matrice SPEC.

9.6 *NAREDBE READ, WRITE i NAMELIST*

Naredba NAMELIST, što znači spisak ili lista simbola (naziva), važi samo za FORTRAN-prevodioce većih računara.* Ona zamenjuje funkciju naredbe FORMAT kod ulaza i izlaza po naredbama

```
READ (i,x)
WRITE (i,x)
```

gde x znači naziv grupe simbola ili *grupni naziv* iz naredbe NAMELIST a i je pozivna šifra datoteke. Grupni naziv i svi simboli iz grupe moraju prethodno biti definisani u jednoj od naredbi NAMELIST.

Naredba NAMELIST je jedna neaktivna naredba (kao FORMAT, DIMENSION, REAL, COMPLEX i tsl.), kojom se daju grupni nazivi grupama simbola skalarnih i indeksnih promenljivih. Njen opšti oblik je:

```
NAMELIST /x/a,b, ... /y/c,d, ... /z/e,f, ...
```

gde su x, y, z, ... grupni nazivi za simbole navedene iza kose crte (do sledećeg grupnog naziva ili do kraja naredbe NAMELIST). Naziv grupe je i sam jedan simbol, sastavljen od 1—6 znakova, od kojih prvi mora biti slovo.

Pravila za naredbu NAMELIST su sledeća:

1. simbol promenljive ili matrice može se nalaziti u više grupa, tj. ne mora pripadati samo grupi jednog grupnog naziva;
2. jedan grupni naziv može biti u jednom programu naveden samo u jednoj naredbi NAMELIST i može se koristiti samo u naredbama READ (i,x) i WRITE (i,x) tog programa;
3. pre nego što se simbol jedne indeksne promenljive navede pod nekim grupnim nazivom jedne naredbe NAMELIST, odgovarajuća matrica mora biti dimenzionisana naredbom DIMENSION.

Pogledajmo jedan primer:

```
DIMENSION A(2,3), K(5), J(5), P(5)
NAMELIST /ULAZ/ A,C,K,J(2) /IZLAZ/ P,K,A(1,3)
READ (5, ULAZ)
. . . . . .
WRITE (7, IZLAZ)
. . . . . .
```

---

* IBM 360, prevodilac H i prevodilac u sistemu upravljačkih programa DOS III; CII 10070 (Sigma 7); ICL System 4, prevodilac za modele 50 i veće (65k); i dr.

Izvršavanjem naredbe READ čitaće se redom slogovi koji sadrže vrednosti šest članova matrice A, vrednost promenljive C, vrednosti pet članova matrice K i vrednost indeksne promenljive J(2). Izvršavanjem naredbe WRITE upisivaće se redom na izlazni medijum slogovi sa vrednostima svih članova matrica P i K i jedna vrednost za indeksnu promenljivu A(1,3).

9.6.1 *PODACI NA ULAZU I IZLAZU*

Podaci koji se čitaju sa spoljnog medijuma po naredbi READ (i,x), gde je x jedan grupni naziv, prethodno definisan naredbom NAMELIST (i eventualno naredbom DIMENSION, ako sadrži i indeksne promenljive), moraju biti na spoljnom medijumu upisani po sledećim pravilima:

1. prvi znak svakog sloga se ignoriše, što znači da podaci u slogu moraju počinjati od druge pozicije (kolone);
2. početak i kraj svakog niza vrednosti, koje pripadaju simbolima jednog grupnog naziva, moraju biti standardno obeleženi; u prvom slogu, sa početkom u drugoj koloni, mora se nalaziti znak & i *neposredno* iza njega grupni naziv; u poslednjem slogu, takođe od druge kolone, znak & i *neposredno* iza njega reč END; to su jedini znaci koji su dozvoljeni u prvom i poslednjem slogu*;
3. podaci počinju sa drugim slogom i mogu zauzimati proizvoljan broj slogova;
4. dozvoljeni oblici podataka su sledeći:
   a. *simbol promenljive = vrednost*
      promenljiva može biti skalarna ili indeksna; primeri: X = 1.5; J(1) = 0; indeks mora biti samo celobrojna konkonstanta;
   b. *simbol matrice = niz konstanti,* odvojenih zarezima
      u nizu konstanti, neke (ili sve) mogu biti date u obliku *k*konstanta;* broj *k* pokazuje koliko uzastopnih promenljivih treba da dobije istu vrednost *konstanta;* celokupan broj konstanti mora biti jednak broju članova matrice; primer: L = 2,3,4,5*8, gde je L jednodimenzionalna matrica od 8 članova;

* kod FORTRAN-prevodioca H za računar IBM 360 može se u prvom slogu odmah nastaviti sa podacima, a zadnji slog sadrži iza podataka tekst &END

c. brojevi na desnoj strani znaka jednakosti mogu biti celi, realni ili kompleksni, ili logičke konstante; po pravilu za aritmetičke naredbe, vrsta broja na desnoj strani pretvara se u vrstu kojoj pripada simbol na levoj strani znaka jednakosti, npr. A = 7 znači $0.7 \cdot 10^1$, a J = 7.5 znači 7; kompleksne i logičke konstante se ne pretvaraju u vrstu kojoj pripada simbol na levoj strani i moraju se pisati po pravilima za te konstante, tj. (*a*,*bi*) odnosno T ili .TRUE., F ili .FALSE.;

5. podaci se međusobno odvajaju zarezima;
6. znaci blanko su dozvoljeni svuda, izuzev u sastavu faktora ponavljanja *k* (vidi *b*, gore), numeričkih i logičkih veličina;
7. poslednji podatak u svakom slogu sa podacima mora biti neka konstanta i iza nje zarez (vidi fusnotu za tačku 2., na prethodnoj stranici).

Kao primer, neka su nam dati sledeći slogovi podataka, koji treba da se čitaju pomoću naredbi READ i NAMELIST (prvi znak je uvek na mestu druge pozicije sloga):

```
1. slog:  &AUZ
2. slog:  C = 5.2,      J(1) = 8,      D(1,1) = 0.5,
3. slog:  B = 5.0,      10.0,    15.0,
4. slog:  A = 10*0.0,   K = 1, 2, 3, 3*4,
5. slog:  L = 5,        E = 10,        F = 0.135E—11,
6. slog:  &END
```

Naredbe, potrebne za ulaz ovih slogova, jesu:

```
DIMENSION J(5), D(2,3), B(3), A(10), K(6)
NAMELIST /AUZ/C, J(1), D(1,1), B, A, K, L, E, F
READ (5,AUZ)
```

Interno, promenljive će posle izvršavanja naredbi imati vrednosti:

C	$= 0.52 \cdot 10^1$	A(1)	$= 0.0 \cdot 10^0$	K(5)	= 4
J(1)	= 8	A(2)	$= 0.0 \cdot 10^0$	K(6)	= 4
....		....		L	= 5
D(1,1)	$= 0.5 \cdot 10^0$	A(10)	$= 0.0 \cdot 10^0$	E	$= 0.1 \cdot 10^2$
....		K(1)	= 1	F	$= 0.135 \cdot 10^{-11}$
B(1)	$= 0.5 \cdot 10^1$	K(2)	= 2		
B(2)	$= 0.1 \cdot 10^2$	K(3)	= 3		
B(3)	$= 0.15 \cdot 10^2$	K(4)	= 4		
		K(1)	= 1,		

Podaci u ulaznim slogovima ne moraju ići istim redom kao simboli promenljivih i matrica u naredbi NAMELIST. Takođe, ne mora u ulaznim slogovima postojati vrednost za svaki simbol pod grupnim nazivom u naredbi NAMELIST.

Kao primer za izlaz napišimo naredbe, koje se odnose na iste simbole i njihove vrednosti kao u gornjem primeru:

```
DIMENSION J(5), D(2,3), B(3), A(10), K(6)
NAMELIST /AIZ/B, K(1), K(2), K(3), F, L
WRITE (7,AIZ)
```

Izvršavanjem ovih naredbi dobiće se na spoljnom medijumu slogovi sledećeg oblika (prvi znak je uvek na mestu druge pozicije sloga):

```
1. slog:  &AIZ
2. slog:  B  = 5.0,      10.0,          15.0, K(1) = 1,
3. slog:  K(2) = 2, K(3) = 3, F = 0.135E—11,    L = 5,
4. slog:  &END
```

U izlazne slogove se, dakle, upisuju ne samo vrednosti nego i simboli promenljivih i matrica, onako kako su navedeni pod grupnim nazivom u naredbi NAMELIST. To znači da se slogovi formirani naredbom WRITE (i,x) mogu čitati samo naredbom READ (i,x), uz odgovarajuće naredbe NAMELIST. Polje za svaki podatak u izlaznom slogu dovoljno je da primi sve značajne cifre, a vertikalno su podaci sređeni u kolone.

9.7 *NAREDBE READ I WRITE BEZ FORMATa*

Do sada smo imali prilike da upoznamo samo takve ulazno-izlazne naredbe, kod kojih smo mogli, u manjoj ili većoj mjeri, uticati na izbor pojedinih podataka (polja) u slogu i na njihov oblik na spoljnim nosiocima. Kod naredbi READ i WRITE sa naredbom FORMAT, za to su nam služile specifikacije, bilo čvrsto ugrađene u program ili promenljive, u vidu ulaznih podataka. Pomoću specifikacija i liste simbola moguć je istovremeno izbor podataka iz sloga i njihovih oblika, u vrlo širokim granicama.

Kod naredbi READ i WRITE sa naredbom NAMELIST takođe je moguća selekcija podataka u okviru sloga, ali je izbor njihovog oblika i rasporeda strožije ograničen pravilima, kao što smo videli u prethodnim primerima.

Isti par naredbi READ i WRITE može se koristiti još u jednom obliku:

READ (*i*) *lista*
WRITE (*i*) *lista*

gde *i* i *lista* imaju značenja koja su ista kao i u drugim oblicima ovih naredbi, dok je kôd kojim su svi podaci predstavljeni na spoljnim medijumima tzv. unutrašnji, interni kôd računara. Pošto je interni kôd računara nepodesan za direktnu upotrebu (binarni kôd), ove naredbe se koriste u izvornom programu isključivo za ulaze i izlaze međurezultata na magnetne medijume.

Po naredbi READ *(i) lista,* iz datoteke na spoljnom nosiocu, čija je pozivna šifra *i,* u memoriju se prenose podaci u vidu jednog sloga u binarnom kôdu i u programu se smeštaju na mesta rezervisana za promenljive iz *liste.* Pošto su ulazni podaci, za koje se koristi ovaj oblik naredbe READ, uvek u internom kôdu računara, to naredba FORMAT nije potrebna. Ovaj oblik naredbe READ upotrebljava se za čitanje slogova, koji su na spoljni nosilac prethodno upisani naredbama WRITE (*i*) *lista.*

Kao primer za upotrebu ovih naredbi, neka je potrebno da se formira jedan slog sa međurezultatima, koji će biti privremeno upisan u datoteku sa pozivnom šifrom J. Napišimo samo one naredbe sa kojima se slog upisuje i kasnije ponovo koristi u programu:

```
. . . . . .
2 FORMAT (I2)
  READ (5,2) J
. . . . . .
  WRITE (J) NBR, TSK, K, A
. . . . . .
  READ (J) L, C, M, R
. . . . . .
```

Pretpostavljeno je da promenljive, u trenutku izvršavanja naredbe WRITE, već imaju određene posebne vrednosti. Takođe, pre izvršavanja druge naredbe READ, pretpostavimo da je početak upisanog sloga u datoteku J već doveden u jedinici pod glavu za čitanje. Izvršavanjem naredbe READ (J) L, C, M, R biće pročitan slog sa ranije upisanim vrednostima i te vrednosti će sada biti dodeljene promenljivima iz liste. Da je ta naredba glasila READ (J) L, C, poslednja dva polja formiranog sloga bila bi preskočena, ali bi naredba bila izvršena bez zastoja. Međutim, ako bi naredba bila READ (J)

L, M, ona se ne bi mogla izvršiti jer se ne slažu vrste brojeva u listi i u slogu na spoljnom medijumu. Do zastoja bi takođe došlo i kada bi naredba glasila READ (J) L, C, M, R, A jer je broj promenljivih u listi veći od broja poljâ u ulaznom slogu.

U ovom primeru vidimo da pozivna šifra datoteke nije celobrojna konstanta, što smo do sada isključivo primenjivali. Kod svih oblika naredbi READ i WRITE pozivna šifra može biti i celobrojna promenljiva, pod uslovom da u trenutku izvršavanja ulazno-izlazne naredbe već ima određenu vrednost. U gornjem primeru vrednost je određena ulazom odgovarajućeg podatka sa spoljnog nosioca ali isto tako može biti i izračunata u prethodnom delu programa.

9.8 *NAREDBE END FILE, REWIND I BACKSPACE*

Ove tri naredbe služe za manipulisanje sa spoljnim magnetnim medijumima, koji se nalaze u perifernim jedinicama računara. Njihov opšti oblik je:

END FILE *i*
REWIND *i*
BACKSPACE *i*

gde je *i* pozivna šifra datoteke na magnetnoj traci ili disku. U izvornom programu, *i* je celobrojna konstanta ili celobrojna promenljiva. Ako je upotrebljen simbol promenljive, njemu mora biti dodeljena vrednost pre izvršavanja naredbe.

Naredba END FILE prilikom izvršavanja upisuje jedan specijalni* znak za kraj datoteke, posle upisivanja poslednjeg sloga u datoteku. Pri upotrebi magnetnih spoljnih medijuma, ovo je standardni način obeležavanja kraja datoteke. Prilikom ponovnog čitanja, specijalni znak će biti pročitan u toku poslednjeg izvršavanja naredbe READ, pa će upravljački program računara preduzeti dalji automatski postupak. Ovakav znak je potreban na magnetnim medijumima zato što prethodni sadržaj ostaje na medijumu iza onog mesta do kojeg je upisan novi. Taj specijalni znak predstavlja, dakle, fizičku granicu između novog sadržaja i prethodnog, koji je za naš program nepotreban.

Naredba REWIND odnosi se na magnetnu traku. Njenim izvršavanjem kotur trake se premotava unatrag do početka aktivne dužine trake, gde se nalazi početak prvog sloga datoteke. Na kraju

* ovaj znak se u originalu zove EOF (End of File)

programiranih operacija sa datotekom na traci, izvršavanjem ove naredbe omogućuje se upotreba iste datoteke ispočetka, bilo u istom ili u narednom programu. Ako se naredba daje u momentu kada je traka već premotana do kraja, ona nema nikakvog dejstva.

Naredba BACKSPACE deluje takođe samo u slučaju kada se datoteka nalazi na magnetnoj traci. Svakim izvršavanjem te naredbe magnetna traka se vraća unatrag za dužinu jednog sloga, posle čega je spremna za čitanje tog sloga ili upisivanje novog na njegovo mesto. Traka se po naredbi BACKSPACE uvek vraća na početak prethodnog sloga, bez obzira na njegovu dužinu, koja može biti vrlo različita (promenljiva dužina liste u naredbama WRITE!).

U primeru u odeljku 9.7, između naredbi WRITE (J) i READ (J) tog programa, trebalo bi da se nalazi jedna naredba REWIND (J) ili BACKSPACE (J), što zavisi od toga da li je naredba WRITE (J) izvršena u programu samo jedanput ili više puta. Ako to nije bilo prvo izvršavanje naredbe WRITE (J), tada je jedina praktična mogućnost da se dođe do početka upisanog sloga izvršavanje jedne naredbe BACKSPACE (J) pre naredbe READ (J).

Isto kao što naredba BACKSPACE znači preskakanje jednog sloga *unatrag,* tako se može preskočiti i jedan slog *unapred,* pomoću naredbe READ (*i*) *lista* u kojoj je lista izostavljena (blanko). Traka će se u jedinici pomeriti unapred za jedan slog i stati na mestu gde je početak narednog sloga. Razume se, ovo je moguće samo posle neke naredbe READ jer samo tada u datoteci već postoje slogovi, koji se mogu odbrojavati.

9.9 *ULAZ I IZLAZ U DIREKTNOM PRISTUPU*

Sve naredbe za ulaz i izlaz koje smo do sada upoznali (Glava 4 i Glava 9, odeljci 1 do 8) odnose se na podatke na medijumima u takvim perifernim uređajima računara, čija je karakteristika *sekvencijalni pristup* podacima.* Takvi uređaji su svi uređaji za kartice i papirnu traku i uređaji za magnetnu traku.

Za uređaje sa direktnim pristupom, kakvi su uređaji magnetnog diska, postoje posebne ulazno-izlazne naredbe zbog specifičnosti tih operacija, koje ćemo dalje videti. Četiri naredbe za operacije ulaza i izlaza na medijume sa *direktnim pristupom***, koje ćemo

---

* definiciju sekvencijalnog i direktnog pristupa vidi u Glavi 1, odeljak 1.2.2, »Magnetni disk«

** ovde opisane naredbe važe za FORTRAN-prevodilac H i onaj u upravljačkom sistemu DOS III kod računara IBM 360, za FORTRAN-prevodioce računara ICL System 4 i IBM 1130

izložiti u sledećim odeljcima i primerima, zovu se opštim imenom naredbe za direktni ulaz i izlaz ili kraće *direktne ulazno-izlazne naredbe*. Sve prethodno proučavane ulazno-izlazne naredbe, pošto se odnose na spoljne medijume u uređajima sa sekvencijalnim pristupom, zovu se *sekvencijalne ulazno-izlazne naredbe*.

Četiri direktne ulazno-izlazne naredbe su READ, WRITE, DEFINE FILE i FIND. Naredbe READ i WRITE služe za prenos podataka sa diska u memoriju računara i obratno. Pomoću određenih parametara u samim naredbama, može se odrediti *mesto* u datoteci odakle treba da se čita odnosno gde treba da se upiše slog podataka.

Naredbom DEFINE FILE definišu se karakteristike datoteke sa kojom će se izvoditi ulazno-izlazne operacije. Samo one datoteke, koje su prethodno u programu definisane pomoću ove naredbe, mogu se koristiti u direktnim ulazno-izlaznim naredbama FORTRAN-programa. Naredbom FIND skraćuje se ukupno vreme koje se troši na operacije ulaza: dok se po ovoj naredbi mehanizam sa glavama za čitanje-upisivanje premešta u novi položaj, koji odgovara sledećem podatku u datoteci, istovremeno se obavljaju interne operacije u memoriji računara. Kada dođe red na sledeću naredbu READ, u izvršavanju te naredbe neće biti gubitaka na traženje sledećeg podatka.

Pored ove četiri specijalne ulazno-izlazne naredbe, za operacije sa datotekama na diskovima koristi se i naredba FORMAT, u punom obimu koji je izložen u odeljku 9.3.

Svaki slog u jednoj datoteci sa direktnim pristupom, dakle takvoj čiji je spoljni nosilac magnetni disk, ima u svom sastavu jedan jedinstveni broj. Zato se u naredbama READ, WRITE i FIND, osim pozivne šifre datoteke koju smo davali u sekvencijalnim naredbama, mora uvek navesti i broj sloga koji treba da se čita, upiše ili nađe. Zahvaljujući tom broju, mi možemo da operišemo samo sa odabranim slogom, ne prolazeći pri tome kroz celu datoteku od početka pa do sloga koji nam je potreban.

9.10 *NAREDBA DEFINE FILE*

Slogovi svake datoteke, sa kojom će se raditi pomoću direktnih ulazno-izlaznih naredbi, moraju se prethodno definisati jednom naredbom DEFINE FILE i to kako u glavnom programu tako i u svakom potprogramu, koji se poziva naredbama glavnog programa. Ako se za definisanje jedne datoteke upotrebi u istom programu ili potprogramu više naredbi DEFINE FILE, sve ostale naredbe osim

prve neće imati nikakvog dejstva. Pòred toga, naredba DEFINE FILE za neku datoteku na disku mora hronološki biti *ispred* ostalih naredbi, koje se odnose na tu datoteku. Opšti oblik naredbe je:

DEFINE FILE a_1 (m_1,r_1,f_1,v_1), a_2 (m_2,r_2,f_2,v_2), . . ., a_n (m_n,r_n, f_n,v_n)

Parametar *a* je jedna celobrojna konstanta, pisana bez algebarskog znaka, koja predstavlja pozivnu šifru datoteke. Parametar *m* je takođe ceo broj i to broj slogova u datoteci čija je pozivna šifra *a*. Celim brojem *r*, bez algebarskog znaka, definiše se dužina najdužeg sloga u datoteci; ta dužina se može izraziti brojem znakova, brojem pozicija (u memoriji) ili brojem mašinskih reči. Koja od ovih jedinica će biti upotrebljena za dužinu sloga, zavisi od parametra *f*. Parametar *f* određuje da li je ulaz-izlaz po specifikacijama neke naredbe FORMAT ili bez toga; on može biti jedno od sledećih slova:

L — čitanje-upisivanje može se vršiti i sa naredbom FORMAT i bez nje; dužina sloga *r* izražava se brojem pozicija (adresâ) koje slog zauzima u memoriji;

E — čitanje-upisivanje vrši se isključivo po specifikacijama neke naredbe FORMAT; dužina sloga *r* izražava se brojem znakova;

U — čitanje-upisivanje vrši se isključivo bez naredbe FORMAT; dužina sloga *r* izražava se brojem mašinskih reči.

Parametar *v* je jedna celobrojna skalarna promenljiva, koja se zove *asocijativna promenljiva*. Na kraju svake ulazno-izlazne operacije, vrednost ove promenljive postaje broj onog sloga koji neposredno sledi poslednji čitani ili upisani slog. Na kraju operacije traženja (FIND), parametar *v* dobija vrednost broja nađenog sloga. Promenljiva *v* ne sme biti u listi onih naredbi READ i WRITE koje rade sa datotekom, koja je definisana pomoću naredbe DEFINE FILE.

Pogledajmo sledeći primer:

```
DEFINE FILE 8(50,100,L,I2), 9(100,50,L,J3)
```

Definisane su dve datoteke, sa pozivnim šiframa 8 i 9. U prvoj datoteci ima 50 slogova podataka, a maksimalna dužina sloga je 100 pozicija u memoriji računara. Slovo L znači da se podaci mogu prenositi sa i bez FORMATa. I2 je asocijativna promenljiva, čija vrednost znači broj sledećeg sloga.

Podaci u drugoj datoteci grupisani su u ukupno 100 slogova, maksimalne dužine 50 pozicija u memoriji računara. Slovo L znači da se podaci mogu prenositi sa i bez FORMATa. J3 je asocijativna promenljiva, čija vrednost znači broj sledećeg sloga.

Ako bismo u prethodnoj naredbi DEFINE FILE umesto slova L pisali E, tada bi uz svaku ulazno-izlaznu naredbu trebalo dati broj naredbe FORMAT, po čijim će se specifikacijama vršiti prenos podataka. Ako podaci treba da se prenose bez upotrebe naredbe FORMAT, tada gornje datoteke treba definisati ovako:

DEFINE FILE 8(50,25,U,I2), 9(100,13,U,J3)

za računar kod koga je dužina mašinske reči 4 pozicije.

Prilikom programiranja direktnih ulazno-izlaznih operacija, mora se voditi računa o saglasnosti dužine sloga po specifikacijama naredbe FORMAT i parametra za dužinu sloga u naredbi DEFINE FILE. Na primer, ako je datoteka definisana naredbom:

DEFINE FILE 8(10,48,L,K8)

tada naredba FORMAT, koja je upotrebljena sa naredbom za ulaz odnosno izlaz, ne sme da zahteva dužinu sloga veću od 48 znakova. Sledeće dve naredbe FORMAT odgovaraju tom uslovu:

FORMAT (4F12.1) ili
FORMAT (I12, 9F4.2)

jer je i po jednoj i po drugoj dužina sloga jednaka vrednosti parametra r u naredbi DEFINE FILE ($4 \cdot 12 = 48$ i $12+9 \cdot 4 = 48$). Uopšte, parametar za dužinu sloga u naredbi DEFINE FILE mora da definiše dužinu sloga jednaku ili veću od one koju će slogovi stvarno imati, bilo o kojoj jedinici dužine se radilo. Primer za to imamo u alternativno izraženoj dužini sloga u datoteci 9.

9.11 *DIREKTNE NAREDBE READ I WRITE*

Po direktnim naredbama READ i WRITE vrši se prenos podataka iz datoteka na magnetnim diskovima u memoriju računara i obratno. Datoteke na koje se odnose ove naredbe moraju biti definisane naredbama DEFINE FILE. Opšti oblik naredbi je:

READ (*a'r,b*) *lista* i
WRITE (*a'r,b*) *lista*

Parametar *a* je jedna celobrojna konstanta, pisana bez algebarskog znaka, ili jedna celobrojna promenljiva; ona predstavlja pozivnu šifru datoteke i mora se pisati sa apostrofom, kao gore. Parametar *r* je jedan aritmetički izraz, koji ima celobrojnu vrednost; taj broj je redni broj sloga u okviru datoteke *a*. Parametar *b* je broj naredbe FORMAT, po čijim se specifikacijama vrši prenos podataka; ovaj parametar se može i izostaviti*. Za *listu* važe sva pravila koja smo videli kod sekvencijalnih ulazno-izlaznih naredbi; u njoj se jedino ne sme nalaziti simbol asocijativne promenljive *v* iz odnosne naredbe DEFINE FILE.

Pogledajmo primer:

```
      DEFINE FILE 8(500,100,L,ID1), 9(100,28,L,ID2)
      DIMENSION M(10)
      . . . . . .
      . . . . . .
      ID2 = 21
      . . . . . .
      . . . . . .
   10 FORMAT (5I20)
    9 READ (8'16,10) (M(K), K = 1,10)
      . . . . . .
      . . . . . .
   13 READ (9'ID2+5) A, B, C, D, E, F, G
```

Po naredbi 9 READ prenose se podaci iz datoteke 8, počev sa slogom čiji je redni broj 16, i smeštaju se redom na mesta članova jednodimenzionalne matrice M, po dvadeset znakova za svaki član. Dva sloga datoteke su potrebna da se zadovolji lista, jer svaki slog sadrži vrednosti pet članova (100 znakova). Na kraju operacije, vrednost asocijativne promenljive ID1 je broj 18.

Po naredbi 13 READ prenose se podaci iz datoteke 9, počev sa slogom čiji je redni broj 26; podaci se čitaju bez naredbe FORMAT i to dok ne bude iscrpljena lista u naredbi 13. Pošto je naredbom DEFINE FILE utvrđena dužina sloga od 28 pozicija za datoteku 9, lista u naredbi 13 zahteva odgovarajuću količinu podataka (sedam promenljivih u listi su realne veličine obične tačnosti, a u ovom računaru** takve veličine zauzimaju po četiri pozicije). Na kraju

* Prevodilac H za računar IBM 360 dozvoljava još dva parametra, samo za ulazne naredbe: END = c i ERR = d (o tome videti priručnik IBM C28 — 6515, koji je preveden na srpskohrvatski)

** svi modeli računara IBM 360, koji rade pod kontrolom sistema upravljačkih programa DOS III

operacije, vrednost asocijativne promenljive ID2 je broj 27. Ako se ta vrednost u međuvremenu ne promeni po programu, pri sledećem izvršavanju naredbe 13 čitanje će početi sa slogom broj 32.

Naredba DEFINE FILE u prethodnom primeru mogla je biti napisana i na sledeći način:

```
DEFINE FILE 8(500,100,E,ID1), 9(100,7,U,ID2)
```

Na broj sloga sa kojim počinje čitanje može se uticati i pomoću naredbe FORMAT. Na primer, da je naredba 10 u prethodnom primeru bila:

```
10 FORMAT (//5I20)
```

tada bi po naredbi 9 READ bili preskočeni slogovi 16 i 17, zatim pročitan slog 18, preskočeni slogovi 19 i 20, pa pročitan slog 21 i, pošto je lista iscrpljena, asocijativna promenljiva ID1 bi dobila vrednost broja 22.

9.12 *NAREDBA FIND*

Po naredbi FIND biće nađen u datoteci sledeći ulazni slog, za vreme dok se podaci iz tekućeg ulaznog sloga obrađuju po programu. Time se povećava efektivna brzina rada mašinskog programa. Podaci iz nađenog ulaznog sloga ne ulaze u memoriju dok se ne izvrši naredba READ za taj slog.

Opšti oblik ove naredbe je:

```
FIND (a'r)
```

gde parametri *a* i *r* imaju ista značenja kao u direktnim naredbama READ i WRITE (vidi odeljak 9.11). Datoteka u kojoj se pomoću naredbe FIND traži sledeći slog, mora biti definisana u jednoj naredbi DEFINE FILE.

Primer:

```
10 FIND (8'50)
   . . . . . .
   . . . . . .
   . . . . . .
15 READ (8'50) A, B
```

Dok se izvršavaju naredbe između onih sa brojevima 10 i 15, biće nađen slog 50 u datoteci 8. Da se među tim naredbama (između 10 i 15) nalazila naredba WRITE (8'50) C, D, efekat koji se inače postiže naredbom FIND bio bi uništen.

Pogledajmo još jedan primer, u kojem su korišćene sve naredbe za direktni pristup:

```
      DEFINE FILE 8(1000,72,L,ID8)
      DIMENSION A(100),B(100),C(100),D(100),E(100),F(100)
      . . . . . .
      . . . . . .
   15 FORMAT (6F12.4)
      FIND (8'5)
      . . . . . .
      . . . . . .
      ID8 = 1
      DO 100 I = 1,100
      READ (8'ID8+4,15) A(I),B(I),C(I),D(I),E(I),F(I)
  100 CONTINUE
      . . . . . .
      . . . . . .
      . . . . . .
      DO 200 I = 1,100
      WRITE (8'ID8+4,15) A(I),B(I),C(I),D(I),E(I),F(I)
  200 CONTINUE
      . . . . . .
      . . . . . .
      END
```

Ovim primerom je ilustrovana sposobnost direktnih ulazno-izlaznih naredbi da u memoriji skupljaju i raspoređuju podatke po zadatom planu. Naredbe, koje pripadaju ciklusu prve naredbe DO u primeru (DO 100), upisuju vrednosti u polja svih članova matrica A do zaključno F, prenoseći podatke iz petog, desetog, petnaestog, ..., petstotog sloga datoteke sa pozivnom šifrom 8. Matrica A prima prvu vrednost iz svakog petog sloga, matrica B drugu vrednost itd., saglasno naredbi 15 FORMAT i listi naredbe READ. Kada se završi operacija ulaza, svi slogovi su raspoređeni po poljima matricâ A do zaključno F. Na kraju rada ciklusa prve naredbe DO, asocijativna promenljiva ID8 ima vrednost 501.

Naredbe, koje pripadaju ciklusu druge naredbe DO (DO 200), grupišu elemente podataka kao što je definisano listom naredbe WRITE i naredbom 15 FORMAT. Sve grupe elemenata prenose se u jednu istu datoteku, čiji je pozivni broj 8. Prenošenje počinje sa 505-tim slogom i nastavlja se u intervalima po pet sve dok ne bude upisan i 1000-ti slog.

VEŽBE

1. Kako će biti odštampani, prema datim specifikacijama, sledeći brojevi:

I6:	0, 1, 12, —634, 85476, 5239753
F6.2:	0.0, 1.0, 25.37, —456.78, 0.05, 146.00, 74.63
F5.0:	0.0, 1.0, 25.37, —74.63
E10.3:	$0.0\cdot10^{0}$, $0.0064\cdot10^{2}$, $54892.0\cdot10^{-4}$, $-248.05\cdot10^{-3}$
1PE10.2:	$0.0\cdot10^{0}$, $10.0\cdot10^{1}$, $0.00075\cdot10^{-2}$, $123456.78\cdot10^{6}$

2. Date su sledeće interne vrednosti promenljivih: I = 15, R = 275.4, S = —25.8, T = 134.5. Kako će tačno izgledati štampani red, ako je naredba za izlaz:

```
WRITE (7,17) I, R, S, T
```

a naredba FORMAT ima sledeće tri verzije:

* a.
```
17  FORMAT (' ','I= ',I3,'R= ',F6.1,'S= ',F6.1,'T= ',F6.1)
```
b.
```
17  FORMAT ('1','BROJ SERIJE: ', I3,'VISINA: ',F6.1,
    1'TEMPERATURA: ',F6.1, 'PRITISAK: ',F6.1)
```
c.
```
17  FORMAT (1H1,'BROJ SERIJE: ',I3/1H0, 'VISINA: ',F6.1/
    11H0, 'TEMPERATURA: ',F6.1/1H0, 'PRITISAK: ',F6.1)
```

* 3. U jednu karticu bušena su četiri broja; to su vrednosti za promenljive ALFA, BETA, GAMA i DELTA. Počev sa prvom kolonom, svaki broj zauzima polje od deset kolona; bušena je i decimalna tačka. Napisati naredbe READ i FORMAT za čitanje te kartice.

4. Isto kao 3., izuzev što nije bušena decimalna tačka. Pretpostavlja se da svi brojevi na kartici imaju po dva decimalna mesta.

5. Isto kao 3., izuzev što svaki broj zauzima polje od četrnaest kolona, a pisan je sa decimalnom tačkom i izložiocem E.

* 6. U jednu karticu bušeni su brojevi u sledećem obliku:

Kolone	*oblik*	*simbol*
1— 3	±xx	IKS
4— 6	xxx	JOT
7—20	±x.xxxxxxE±ee	ARC
21—34	±x.xxxxxxE±ee	DET

Malim slovima označena su mesta cifara. Napisati naredbe za čitanje ove kartice.

* 7. Data je naredba WRITE:

```
WRITE (7,13) I, J, X, Y
```

gde su I i J simboli celobrojnih, a X i Y realnih promenljivih. Kako treba da glasi naredba FORMAT da bi se dobili štampani redovi sledećeg sadržaja (*b* znači blanko):

a. bbbb—25bb50692bbb37.93bb528.78
b. bbbb—25bb50692bbb38.bb529.
c. bbb—25bbb50692bb0.37933Eb02bb0.52878Eb03
d. bbb—25bbb50692bb3.7933Eb01bb5.2878Eb02
e. I=bbb—25bbJ=b50692bbX=bb37.9bbY=b528.8
f. I=bbb—25

J=b50692
X=bb37.9
Y=b528.8

(Za zadatak pod f. treba napisati samo jednu naredbu FORMAT).

8. Isto kao 7, samo je naredba WRITE:

WRITE (7,17) N, U, V, W

gde je N celobrojna, a U, V i W realne promenljive.

a. b5bb—33.b2.6E—04bb1.528135Eb06
b. b5bb—32.71bb0.00026bb0.1528135Eb07
c. b5bb—32.71bb0.26E—03bb1528135.
d. JX EMAX VOLT WATT
b5bb—32.71bb2.6E—04bbbb1.528135Eb06
(Napisati samo jednu naredbu FORMAT).
e. JX=b5bbbEMAX=b—32.71bbbVOLT=b2.6E—04bbbWATT=
=bl.528135Eb06
(Napisati samo jednu naredbu FORMAT).

* 9. Simbol jedne jednodimenzionalne matrice, sa maksimalno deset članova, neka je TFE. U jednu karticu, u kolonama 1—2 bušen je stvarni broj članova matrice, N, dok su u ostalim poljima bušene vrednosti članova matrice (polja su po sedam kolona), kao obični realni brojevi sa decimalnom tačkom, bez izložioca E. Napisati naredbe za čitanje te kartice.

10. Isto kao 9., samo što sada realni brojevi predstavljaju vrednosti *neparnih* članova matrice TFE, pa ih prema tome ima najviše pet. N je sada *broj poslednjeg člana*, a ne ukupan broj članova.

* 11. U jednom redu treba da se odštampaju vrednosti realnih promenljivih P, Q, U i V. P i Q treba štampati bez izložioca, U i V sa izložiocem E. Za vrednosti P i Q treba ostaviti ukupno 12 pozicija u redu, od toga 4 za decimale. Za vrednosti U i V ostaviti po 20 pozicija, od toga 7 za decimale i ne koristiti faktor razmere. Napisati potrebne naredbe.

12. Isto kao 11., samo između P i Q treba štampati celobrojnu vrednost promenljive L sa ukupno 6 pozicija, a kod U i V decimalna tačka treba da bude pomerena za jednu poziciju udesno.

13. L je simbol jedne trodimenzionalne matrice, čije su dimenzije (maksimalne vrednosti indeksâ) date naredbom

DIMENSION L(2,2,3)

U jednu karticu bušeno je dvanaest članova matrice L, pri čemu svaki član (ceo broj) zauzima tri kolone. Napisati naredbe za čitanje vrednosti članova matrice te kartice. Kojim redom bi članovi matrice trebalo da budu bušeni u karticu, pa da lista u naredbi READ bude najjednostavnija?

* 14. Jedna dvodimenzionalna matrica, QSR, ima četrdeset članova, svrstanih u deset redova i četiri kolone. Napisati program po kojem će na spoljni nosilac biti upisana cela matrica, u uobičajenom rasporedu, s tim da u prvom redu na novoj stranici bude naslov »MATRICA QSR«. Vrednosti članova treba na spoljnom nosiocu da budu u obliku koji odgovara specifikaciji E20.7. (*Napomena*: paziti da u svakom redu budu samo četiri vrednosti).

* 15. Data je jedna celobrojna promenljiva I, čije su vrednosti $1 \leqslant I \leqslant 12$. Napisati program koji će štampati, u polju od tri pozicije, jednu od skraćenica za naziv meseca (JAN, FEB, MAR itd.), zavisno od vrednosti promenljive I. Program se piše za računar kod koga je dužina mašinske reči za alfanumeričke podatke jednaka četiri pozicije.

* 16. Dat je sledeći deo programa:

```
      DIMENSION SPEC(12), T(10)
      READ (3,55) (SPEC(I), I = 1,12)
   55 FORMAT (12A4)
      . . . . . .
      . . . . . .
      WRITE (5,SPEC) (T(I), I = 1,10)
```

Napisati šta treba da bude bušeno u karticu koja se čita naredbom READ, da bi deset vrednosti matrice T bilo odštampano na sledeći način:

a. svih deset u jednom redu, svaka po specifikaciji F10.2;
b. u dva reda po pet vrednosti, svaka po specifikaciji 1PE20.6;
c. u prvom redu T(1), po specifikaciji F10.2;
drugi red blanko;
u trećem, četvrtom i petom redu po tri vrednosti — T(2),T(3),T(4); T(5),T(6),T(7); T(8),T(9),T(10) — svaka po specifikaciji 1PE20.6.

* 17. U datoteci na disku, čija je pozivna šifra 8, ima 500 slogova maksimalne dužine 100 znakova (parametar L). U programu je definisana jedna jednodimenzionalna matrica od deset članova, čije su vrednosti celi brojevi (simbol matrice neka je M). Treba napisati program koji će u datoteku na disku, počev od njenog 16. sloga, preneti vrednosti svih deset članova matrice M, pod pretpostavkom da svaka vrednost ima 20 znakova (I20).

Glava 10

POTPROGRAMI

10.1 *UVOD*

U programiranju složenijih zadataka često se može zapaziti da se izvesne grupe naredbi ponavljaju, sa raznim vrednostima za promenljive u izrazima. Već smo videli kako se u takvim slučajevima koristi naredba DO. Međutim, pisanjem naredbe DO i ostalih naredbi u njenom ciklusu, mi ustvari rešavamo problem ponavljanja istih operacija samo za program koji upravo pišemo. Ako se ista grupa operacija javi kasnije u nekom drugom programu, moramo ponovo napisati sve naredbe koje smo već jedanput pisali i čiju smo tačnost proverili kroz rad programa.

Standardne matematičke funkcije, o čijoj smo primeni govorili u odeljku 3.3, jesu sredstvo koje u izvesnim okolnostima takođe olakšava pisanje programa: u aritmetičkom izrazu se samo napiše naziv funkcije, sa argumentima u zagradi, a na tom mestu će se automatski pojaviti njena izračunata vrednost. Te standardne funkcije se nalaze kao gotovi *potprogrami* u biblioteci programâ računara (vidi odeljak 1.3), odakle ih prevodilac uključuje u naš program u fazi prevođenja. Izbor tih funkcija različit je kod FORTRAN-prevodilaca za razne računare; jedan potpuniji pregled standardnih matematičkih funkcija dat je u odeljku 12.3*.

I za takve grupe operacija, koje nisu kao gotovi potprogrami uvrštene u biblioteku standardnih funkcija, u FORTRANu postoji mogućnost stvaranja jednostavnijih programskih jedinica — *aritmetičkih funkcija* ili složenijih programskih jedinica — *potprogramâ*, pomoću kojih se uprošćava rad na programiranju, a kako ćemo

* To je tabela standardnih funkcija za FORTRAN-prevodioce računarâ IBM 360; većina postoji i kod računara drugih tipova. Svaki računski centar ima sličnu tabelu za svoj računar.

videti, i štedi prostor u memoriji računara. Računski centri često imaju i sopstvene zbirke gotovih potprograma u biblioteci računara, tako da ih možemo koristiti za naš program, slično standardnim matematičkim funkcijama.

10.2 *ARITMETIČKE FUNKCIJE*

Ako se jedna funkcija, koja se ne nalazi u biblioteci standardnih matematičkih funkcija, često koristi u *jednom* programu, takvu funkciju možemo programirati pomoću jedne aritmetičke naredbe. Zato što se definiše posebnom aritmetičkom naredbom, takva funkcija se zove funkcija u vidu aritmetičke naredbe ili prostije *aritmetička funkcija.*

Aritmetička funkcija se definiše pisanjem naredbe u obliku $a = b$, gdje je a *naziv funkcije,* a b neki aritmetički izraz. Naziv funkcije je jedan simbol promenljive, izabran prema pravilima za takve simbole, koje smo videli u Glavi 2 i kasnije: dužina naziva je jedan do šest znakova (samo slova, cifre i znak blanko*), od kojih prvi mora biti slovo. Vrsta kojoj pripada naziv funkcije (INTEGER, REAL, DOUBLE PRECISION, COMPLEX ili LOGICAL) određuje se po unutrašnjoj konvenciji FORTRANa, pomoću naredbe IMPLICIT ili pomoću eksplicitnih naredbi za deklarisanje. Toj vrsti pripada i izračunata vrednost funkcije. Opšti oblik naredbe, kojom se definiše aritmetička funkcija, jeste:

$$naziv\ (a_1,\ a_2,\ \ldots,\ a_n) = izraz$$

gde su $a_1, a_2, \ldots, a_n$ fiktivni argumenti (vidi dalje), predstavljeni simbolima skalarnih promenljivih. U nizu fiktivnih argumenata, njihovi simboli moraju biti jedinstveni, tj. jedan isti simbol se ne sme upotrebiti dva puta, ali svaki od njih sme da bude identičan sa bilo kojim simbolom promenljive u programu. Suprotno tome, naziv funkcije mora biti različit od svih simbola za promenljive u programu, a ne sme da bude upotrebljen ni kao argument u sopstvenom nizu fiktivnih argumenata.

Na primer, naredbom:

```
F12 (A,B) = 3.0*A + B**2 + X + Y + Z
```

* Prevodilac ignoriše znak blanko u nazivu funkcije; na primer, BEbSEL prevodilac interpretira kao BESEL. U simbolu promenljive znaci blanko, kao što znamo, nisu dozvoljeni.

definisana je funkcija čiji je naziv F12, a simboli A i B su fiktivni argumenti te funkcije. Ako neka od kasnijih naredbi u programu glasi:

$$C = F*G + F12(D,E)$$

to je ekvivalentno naredbi:

$$C = F*G + 3.0*D + E**2 + X + Y + Z$$

Naredba kojom se definiše funkcija, čime se ujedno određuje i vrsta vrednosti koja se dobija njenim izračunavanjem kao i koje promenljive će imati ulogu fiktivnih argumenata, zove se *definiciona* naredba aritmetičke funkcije. Ona se u programu mora pisati na početku, pre svih aktivnih naredbi. Pisanje naziva funkcije, sa fiktivnim argumentima u zagradama, u aritmetičkom izrazu neke druge aritmetičke naredbe, zove se pozivanje ili *poziv* te funkcije.

Iz prethodnog primera vidimo zašto se simboli A i B zovu fiktivnim argumentima funkcije: prilikom pozivanja funkcije oni se zamenjuju pravim ili stvarnim argumentima (D,E), sa čijim vrednostima se izračunava vrednost funkcije. U gornjem primeru, X, Y i Z nisu fiktivni nego stvarni argumenti jer nisu, kao A i B, navedeni u zagradama na levoj strani definicione naredbe.

Prilikom pisanja poziva jedne aritmetičke funkcije, mora se voditi računa da se u zagradama iza naziva navede isti broj i vrsta stvarnih argumenata kao što su broj i vrsta fiktivnih argumenata u definicionoj naredbi. Na primer, definicionoj naredbi:

$$P\ (S,A,B,C) = SQRT\ (S*(S—A)*(S—B)*(S—C))$$

odgovara naredba koja sadrži poziv:

$$V = P\ (W,X,Y,Z)*H$$

ali ne odgovara naredba:

$$V = P\ (N,\ Q,\ R)*H$$

jer se ne slaže broj argumenata, a osim toga je stvarni argument N različite vrste od fiktivnog argumenta S (ukoliko prethodno nije data naredba REAL N).

Kao još jedan primer za ilustraciju, neka se u programu često traži izračunavanje jednog korena kvadratne jednačine $ax^2 + bx + c = 0$, za različite vrednosti koeficijenata a, b i c. Na početku programa možemo definisati funkciju:

$$X1\ (A,B,C) = (—B+SQRT(B**2—4.0*A*C))/(2.0*A)$$

Kasnije u istom programu, potreban je jedan koren kvadratne jednačine $dy^2 + ey + f = 0$ kao član nekog aritmetičkog izraza; neka je ta naredba:

```
Z = R**3.2 + X1 (D,E,F) + S
```

Pozivom funkcije biće izračunat jedan koren kvadratne jednačine (y_1) za koeficijente *d*, *e* i *f* i ta vrednost će biti sabrana sa ostalim članovima izraza.

Da istaknemo razliku između simbola stvarnih i fiktivnih argumenata, koji međusobno nemaju nikakve veze, uzmimo da se u istom programu traži vrednost jednog korena jednačine:

$$25.7\,x^2 + ax + b = 0$$

Napišimo ponovo definicionu naredbu aritmetičke funkcije X1, zajedno sa naredbom u kojoj se nalazi poziv te funkcije:

```
X1   (A, B, C) = (—B+ SQRT(B**2—4.0*A*C))/(2.0*A)
P = X1(25.7,A,B)
```

Vidimo da stvarnom argumentu A odgovara fiktivni argument b, a stvarnom argumentu B fiktivni argument C, dakle nema nikakve veze između simbola u programu i istih simbola u definicionoj naredbi aritmetičke funkcije.

Za razliku od fiktivnih argumenata, koji mogu biti samo simboli skalarnih promenljivih, stvarni argumenti mogu biti i indeksne promenljive, nazivi standardnih funkcija ili aritmetički izrazi. Na primer, za korišćenje prethodno definisane funkcije X1 (A, B, C), u naredbi koja sadrži poziv te funkcije:

```
X = X1 (1.0 + F, ABS(R — S), Q(1,4))
```

argumenti su, s leva na desno, prost aritmetički izraz, standardna funkcija i indeksna promenljiva. Pre prenošenja na mesta odgovarajućih fiktivnih argumenata, promenljive u svim tim izrazima moraju imati određene brojne vrednosti.

10.3 *POTPROGRAMI FUNCTION*

Nedostatak aritmetičke funkcije je u tome što se ona definiše uvek samo jednom aritmetičkom naredbom i što pripada samo onom izvornom programu u kome je napisana. Ona se ne može prevesti zasebno i posle, kao gotov potprogram, povezivati sa bilo kojim

programom, pozivanjem iz biblioteke. Za tu svrhu u FORTRANu postoji način za formiranje zasebnih programskih jedinica, tzv. *potprograma*. Njih ima dve vrste: potprogrami FUNCTION i potprogrami SUBROUTINE. Ulogu jednih i drugih videćemo iz daljeg izlaganja.

Jedan potprogram FUNCTION piše se, kao i svaki FORTRAN-program, u vidu niza naredbi. Prva naredba karakteriše tu programsku jedinicu kao potprogram FUNCTION i ima opšti oblik:

vrsta FUNCTION *naziv* (a_1, a_2, . . ., a_n)

gde je *vrsta* jedan od atributa INTEGER, REAL ili DOUBLE PRECISION, a kod većih računara još i COMPLEX i LOGICAL. *Vrsta* znači da će vrednost funkcije, koja se izračunava potprogramom FUNCTION, pripadati odgovarajućoj vrsti brojeva. Ovaj parametar nije obavezan; ako se ne napiše, vrsta broja-vrednosti funkcije biće određena po unutrašnjoj konvenciji, na osnovu početnog slova *naziva*. Kod većih računara može se u tu svrhu upotrebiti i naredba IMPLICIT. Što se tiče naziva, argumenata i načina pozivanja potprograma FUNCTION, važi sve što je rečeno za aritmetičku funkciju. Argumente u naredbi FUNCTION zovemo i ovde fiktivnim argumentima, iz istog razloga kao kod aritmetičke funkcije.

Poslednja u nizu naredbi potprograma FUNCTION mora biti naredba END. Između prve i poslednje naredbe potprograma mogu se pisati sve FORTRAN-naredbe kao i u programu. Najmanje jedna od tih naredbi mora biti jedna aritmetička naredba, kod koje je na levoj strani naziv potprograma FUNCTION. Obavezna je i najmanje jedna naredba RETURN, čiju ćemo ulogu videti u primerima.

Pogledajmo sada upotrebu jednog potprograma FUNCTION u sastavu FORTRAN-programa:

```
      Program                          Potprogram
                                       FUNCTION AFS (I,A,B,C)
                                       GO TO (3,4), I
                                     3 AFS = A/B
                                       RETURN
                                     4 AFS = ABS (C—A/B)
                                       RETURN
                                       END
    4 FORMAT (3E14.7)
      READ (5,4) X, Y, Z
      . . . . . .
      J = 1
    6 D = AFS(J,X,Y,Z)—1.0
      . . . . . .
      READ (5,4) Q,R,S
      . . . . . .
      K = 2
```

```
      Program
    7 E = 0.5*AFS (K,Q,R,S)
      . . . . . .
      READ (5,4) A,B,C
      . . . . . .
      I = 1
   10 F = SQRT(AFS(I,C,A,B))
      . . . . . .
      END
```

Program najpre čita tri vrednosti, za promenljive X, Y i Z. U naredbi 6 poziva se prvi put izvršavanje potprograma: vrednosti stvarnih argumenata J, X, Y i Z prenose se u potprogram na mesta fiktivnih argumenata I, A, B i C. Zato što je vrednost prvog argumenta 1, vrednost potprograma AFS biće računata po naredbi 3 (dakle X/Y) i vraćena (RETURN) programu na mesto odakle je pozvan potprogram AFS. Dalje se u programu izračunava vrednost D i nastavlja se izvršavanje programa, sve dok u naredbi 7 ne naiđe ponovo jedan poziv potprograma AFS, sada sa drugim vrednostima stvarnih argumenata. Pošto je sada vrednost prvog argumenta 2, vrednost AFS će biti izračunata po naredbi 4 (u ovom slučaju ABS (S—Q/R)) i opet vraćena programu (RETURN), čije će se izvršavanje nastaviti. Prilikom trećeg pozivanja potprograma AFS, u naredbi 10, fiktivni argumenti potprograma imaju istu grupu simbola kao stvarni argumenti, što pokazuje da se isti simboli mogu slobodno upotrebljavati u obe uloge — dejstvo neće biti ništa drukčije nego u prethodna dva poziva. Vrednost AFS će se opet računati po naredbi 3 i vratiti programu na mesto odakle je poziv upućen; promenljiva F dobiće vrednost kvadratnog korena od AFS = A/B.

Iz primera smo još mogli zapaziti da se u programu i potprogramu mogu upotrebiti isti brojevi naredbi (ovde: 4).

10.4 *NAREDBE COMMON I EQUIVALENCE*

Uzmimo sada drugi primer, u kome ćemo upoznati naredbu COMMON i njenu ulogu u potprogramima FUNCTION i SUBROUTINE. U ovom primeru, u programu se poziva potprogram SUMA, koji formira zbir kvadrata svih članova jednodimenzionalne realne matrice (maksimalno 100 članova):

```
Program                          Potprogram
DIMENSION A(100)                 FUNCTION SUMA(K,X)
1 FORMAT (I4, (E14.7))           DIMENSION X(100)
READ (5,1) N, (A(I), I=1,N)      SUMA = 0.0
S = SUMA(N,A)                    DO 3 I = 1,K
. . . . . .                    3 SUMA = SUMA + X(I)**2
. . . . . .                      RETURN
END                              END
```

Prevođenjem ove dve programske jedinice, biće rezervisan prostor u memoriji za maksimalne dimenzije matrica A i X, čiji su članovi realni brojevi obične tačnosti. Kada se u programu pozove potprogram SUMA, u potprogram će biti preneta vrednost argumenta N na mesto argumenta K, a svi članovi matrice A na odgovarajuća mesta u matrici X. Pošto je očigledno nepotrebno čuvati iste vrednosti na dva razna mesta u memoriji, gornje dve programske jedinice ćemo napisati drukčije:

```
Program                          Potprogram
COMMON A(100)                    FUNCTION SUMA (K,X)
1 FORMAT (I4,(E14.7))            COMMON X(100)
READ (5,1) N, (A(I),I=1,N)       SUMA = 0.0
S = SUMA(N,A)                    DO 3 I = 1,K
. . . . . .                    3 SUMA = SUMA + X(I)**2
. . . . . .                      RETURN
END                              END
```

Ovde smo upotrebili *naredbu COMMON*, čiji je opšti oblik:

$$\text{COMMON } a_1(k_1),\ a_2(k_2),\ \ldots,\ a_n(k_n)$$

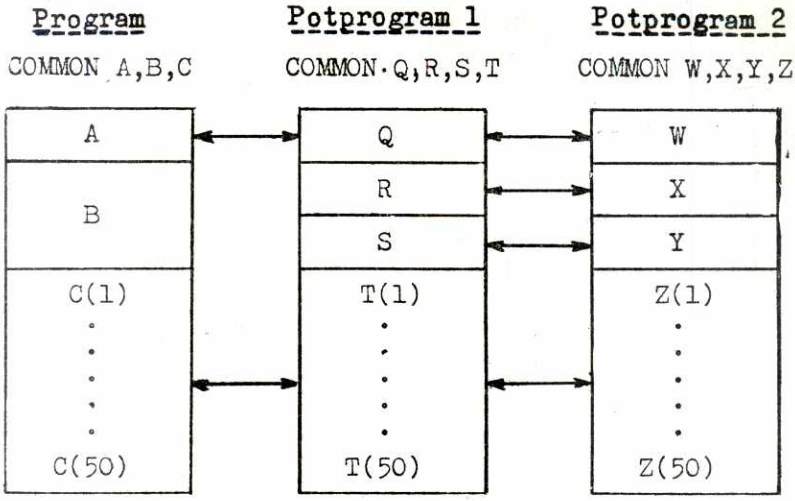
gde su $a_1, a_2, \ldots, a_n$ simboli promenljivih ili matrica, a $k_1, k_2, \ldots, k_n$ grupe od 1 do 3 (kod većih računara do 7) celobrojnih konstanti, odvojenih zarezima; te konstante su dimenzije matrica, tj. maksimalne vrednosti njihovih indeksa. Parametri k mogu biti i izostavljeni, kada se radi o skalarnim promenljivima.

Naredbom COMMON postiže se da matrice (ili promenljive) iz dve posebne programske jedinice zauzimaju u memoriji računara isti prostor. (Prostor koji u memoriji zauzima jedan podatak zovemo *polje*, dok više susednih polja čine *zonu;* ti nazivi su analogni nazivima *polje* i *slog* na spoljnim medijumima). Uslov je da se rezervisanje mesta izvrši naredbama COMMON u svim programskim jedinicama, koje će se u računaru zajedno izvršavati. U gornjem pri-

meru smo tako postupili za matrice A i X, u programu koji poziva potprogram SUMA i u samom potprogramu SUMA.

Dimenzije matrica, koje se definišu naredbama COMMON, moraju biti uzajamno jednake; jedino ako se radi o jednodimenzionalnim matricama, tada ona u potprogramu može biti veća (veći broj članova).

Poredak simbola u naredbama COMMON i vrste brojeva koje su tim simbolima predstavljene nisu bez značaja kada se dodeljuje isti prostor promenljivima i matricama u raznim programskim jedinicama. Uzmimo primer jednog programa sa dva potprograma:



U programu i u potprogramima postoji direktna korespondencija između realnih promenljivih A, Q i W i realnih matrica C, T i Z. Isto tako, samo u potprogramima, između realnih promenljivih R i X odnosno S i Y. Međutim, vrednost realne promenljive dvostruke tačnosti B ne može se koristiti u potprogramima jer deli isti prostor sa parovima R i S odnosno X i Y. O tome treba voditi računa kada se koristi naredba COMMON.

Naredba EQUIVALENCE ima sličnu ulogu kada se radi o skalarnim i indeksnim promenljivim u jednoj istoj programskoj jedinici. Njen opšti oblik je:

EQUIVALENCE $(a_1,a_2,\ldots.)$, $(b_1,b_2,\ldots.)$, ...

gde su u zagradama simboli skalarnih ili indeksnih promenljivih iz *jednog* programa ili potprograma. Vrednosti svih promenljivih u jednom paru zagrada zauzimaju u memoriji računara isto mesto. Dodeljivanje istog mesta u memoriji manjem broju skalarnih promenljivih nema naročitog značaja za uštedu prostora u memoriji. Međutim, u sledećem primeru:

```
DIMENSION A(50), B(5,10), C(2,5,5)
EQUIVALENCE (A(1), B(1,1), C(1,1,1))
```

izjednačavanjem prvih članova matrica A, B i C ustvari se implicitno dodeljuje zajednička zona celim matricama A, B i C.

Pogledajmo nekoliko praktičnih primena naredbe EQUIVALENCE. U početnom delu jednog programa koristi se dvodimenzionalna matrica C, čije su dimenzije 100 × 100. Kasnije u programu, matrica C više nije potrebna, ali se pojavljuje potreba za jednom dvodimenzionalnom matricom A (50 × 50) i jednom jednodimenzionalnom matricom B (100). Članovi svih matrica su realni brojevi. Ušteda prostora u memoriji postiže se ako se napiše:

```
DIMENSION C(100,100), A(50,50), B(100)
EQUIVALENCE (C(1),A(1)), (C(2501),B(1))
```

Svih 2500 članova matrice A deliće jednu zajedničku zonu sa prvih 2500 članova matrice C, pošto obe nisu u programu potrebne u isto vreme. Na isti način, članovi matrice B biće u zajedničkoj zoni sa članovima 2501 do 2600 matrice C.

U sledećem primeru imamo ove naredbe:

```
DIMENSION B(5), C(10,10), D(5,10,15)
EQUIVALENCE (A,B(1),C(5,3)), (D(5,10,2),E)
```

Time što je član B(1) izjednačen sa članom C(5,3), i ostali članovi matrice B sukcesivno dele ista polja sa članovima C(6,3) do C(9,3) matrice C. Posle ovoga, bilo bi nepravilno napisati na primer:

```
EQUIVALENCE (B(2), C(8,3))
```

jer bi to bilo u kontradikciji sa prethodnim. Umesto C(5,3) mogli smo napisati C(25), a umesto D(5,10,2) samo D(100); efekat bi bio isti.

U sledećem primeru vidimo kombinovani efekat naredbi EQUIVALENCE i COMMON u jednom programu:

```
DIMENSION D(3)
COMMON A, B, C
EQUIVALENCE (B, D(1))
```

Naredbom COMMON promenljive A, B i C zauzeće u memoriji susedna polja. Po naredbi EQUIVALENCE uspostaviće se sada jednakost u memoriji na sledeći način:

```
A          najniža adresa u opštoj zoni
B, D(1)
C, D(2)
   D(3)    najviša adresa u opštoj zoni
```

U istom primeru, naredba EQUIVALENCE se ne bi smela napisati ovako:

```
DIMENSION D(3)
COMMON A, B, C
EQUIVALENCE (B, D(3))
```

jer bi u tom slučaju trebalo da član D(1) zauzme u memoriji polje ispred polja promenljive A, sa kojim počinje definisana opšta zona:

```
   D(1)
A, D(2)    najniža adresa u opštoj zoni
B, D(3)
C          najviša adresa u opštoj zoni
```

10.5 *POTPROGRAMI SUBROUTINE I NAREDBA CALL*

Druga vrsta potprograma u FORTRANu, potprogrami SUBROUTINE, u mnogome je slična potprogramima FUNCTION: pravila za dužinu i sastav naziva, pravila za stvarne i fiktivne argumente kao i pravila za upotrebu naredbi RETURN i END, ista su kod obe vrste potprograma. Definiciona naredba, kojom počinje potprogram SUBROUTINE, ima opšti oblik:

SUBROUTINE *naziv* (a_1, a_2, . . ., a_n)

gde je naziv izabran po istim pravilima koja važe za simbol promenljive*, naziv aritmetičke funkcije ili naziv potprograma FUNCTION, a a_1, a_2, . . ., a_n su fiktivni argumenti, koji mogu biti i izostavljeni.

Glavna razlika sa gledišta programiranja između potprograma SUBROUTINE i potprograma FUNCTION jeste u tome što potprogram SUBROUTINE može da vrati programu odjednom *više* izra-

* vidi fusnotu u odeljku 10.2

čunatih vrednosti svojih *argumenata* (može i nijednu), dok potprogram FUNCTION uvek vraća samo *jednu* vrednost, onu koju ima *naziv* posle izvršavanja potprograma. Dok su u programu, koji poziva potprogram FUNCTION, stvarni argumenti morali prethodno da dobiju određene vrednosti, za pozivanje potprograma SUBROUTINE to nije neophodno.

Pozivanje potprograma SUBROUTINE vrši se pomoću naredbe CALL, čiji je opšti oblik:

CALL *naziv* (a_1, a_2, ..., a_n)

gde je *naziv* ime ili naziv potprograma, a a_1, a_2, ..., a_n su stvarni argumenti, koji mogu biti i izostavljeni.

Pogledajmo na primeru kako izgleda jedan potprogram SUBROUTINE i kako on funkcioniše na poziv iz programa:

```
Program                              Potprogram
. . . . . .                          SUBROUTINE SUBPOT(N,P,Q,R,S)
I = 2                                GO TO (8,10), N
A(I) = 2.0                         8 P = 5.0*R — 1.0
X = 0.3                              Q = Q + 1.0
Y = 0.7                              RETURN
CALL SUBPOT(1,Z,A(I),X+Y,W)       10 . . . . . .
S = Z + A(I)                         RETURN
. . . . . .                          END
END
```

Naredbom CALL prenosi se akcija na potprogram SUBPOT, čiji fiktivni argumenti N, Q i R dobijaju određene vrednosti stvarnih argumenata iz programa. U potprogramu se izračunavaju vrednosti fiktivnih argumenata P i Q, koje se zatim vraćaju programu kao vrednosti stvarnih argumenata Z i A(I). Sa tim vrednostima nastavlja se dalji rad programa, izvršavanjem prve naredbe posle naredbe CALL.

U sledećem primeru, u programu se poziva jedan potprogram SUBROUTINE bez argumenata:

```
Program                              Potprogram
COMMON K, A(500)                     SUBROUTINE SORT
1 FORMAT (I4,(E14.7))                COMMON N, X(500)
READ (5,1) K, (A(I),I=1,K)         3 J = 1
CALL SORT                            N = N — 1
```

```
      Program                    Potprogram
    2 . . . . . .                  IF (N) 1, 5, 1
      . . . . . .                1 DO 2 K = 1, N
      STOP 888                      L = K + 1
      END                           IF (X(K)—X(L)) 2, 2, 4
                                 4  T = X(K)
                                    X(K) = X(L)
                                    X(L) = T
                                    J = 2
                                 2  CONTINUE
                                 6  GO TO (5,3), J
                                 5  RETURN
                                    END
```

Program unosi vrednosti K članova u jednodimenzionalnu matricu A, koja zauzima istu zonu sa matricom X iz potprograma. Posle toga, aktivira se potprogram SORT, koji uređuje članove matrice X po rastućem* redosledu njihovih vrednosti. Kada je svih N = K članova u željenom redosledu, rad programa se nastavlja sa naredbom 2, pri čemu su sada članovi matrice A uređeni po rastućem redosledu vrednosti.

Još jedna razlika između potprograma SUBROUTINE, s jedne strane, i standardnih funkcija, aritmetičkih funkcija i potprograma FUNCTION, s druge strane, jeste što se vrste izračunatih vrednosti ne mogu deklarisati deklarisanjem vrste naziva. Kod potprograma SUBROUTINE sam naziv ne dobija vrednost, a argumenti mogu biti, isto kao i kod potprograma FUNCTION, veličine različite vrste.

Na kraju, da načinimo uporedni pregled karakteristika koje imaju četiri vrste programskih funkcija u FORTRANu: standardne matematičke funkcije, aritmetičke funkcije, potprogrami FUNCTION i potprogrami SUBROUTINE. Te karakteristike su date u tabeli 10.1, strana 191.

10.6 *OSTALE MOGUĆNOSTI*

Sve sledeće naredbe i mogućnosti postoje samo u FORTRAN-prevodiocima za veće računare. One su standardizovane u potpunom FORTRANu IV.

Promenljive dimenzije matrica u potprogramima omogućuju uštedu prostora u memoriji računara i bez upotrebe naredbe COMMON. Naredbama DIMENSION rezervišu se zone za matrice u pro-

* tačnije: neopadajućem, pošto dva ili više članova mogu imati iste vrednosti

TABELA 10.1

	Standardna funkcija	Aritmetička funkcija	Potprogram FUNCTION	Potprogram SUBROUTINE
Naziv	1-6 znakova, od kojih prvi slovo	1-6 znakova, od kojih prvi slovo	1-6 znakova, od kojih prvi slovo	1-6 znakova, od kojih prvi slovo
Vrsta	po prvom slovu naziva	po prvom slovu naziva, naredbom IMPLICIT ili eksplicitnim naredbama	po prvom slovu naziva, naredbom IMPLICIT ili eksplicitnim naredbama	nema odredjenu vrstu jer naziv ne dobija vrednost
Definicija	odredjena u FORTRAN-prevodiocu	jedna aritmetička naredba pre prve upotrebe funkcije	proizvoljan broj naredbi, na početku naredba FUNCTION	proizvoljan broj naredbi, na početku naredba SUBROUTINE
Poziv	pisanjem naziva gde je potrebna vrednost funkcije	pisanjem naziva gde je potrebna vrednost funkcije	pisanjem naziva gde je potrebna vrednost funkcije	naredba CALL
Broj argumenata	prema standardnoj definiciji	1 ili više, prema definiciji	1 ili više, prema definiciji	0 ili više, prema definiciji
Broj vrednosti	jedna	jedna	jedna	0 ili više

gramu i potprogramima, pa se zona u potprogramu višestruko koristi. Na primer:

```
      Program
      DIMENSION A(30), B(50)
      . . . . . .
    8 X = FUNKC(A,30,D)
      . . . . . .
      K = 50
    9 Y = FUNKC(B,K,G)
      . . . . . .
      . . . . . .
      END
```

```
      Potprogram
      FUNCTION FUNKC(Z,N,R)
      DIMENSION Z(N)
      FUNKC = R . . . .
      RETURN
      END
```

Prilikom pozivanja potprograma FUNKC u naredbi 8 programa, dimenzija matrice Z postaje 30, a njeni članovi trideset članova matrice A. Sledećim pozivom potprograma, u naredbi 9, dimenzija iste matrice Z biće 50, a u njenoj zoni nalaziće se pedeset članova matrice B.

Još jedan primer, sa potprogramom SUBROUTINE i dvodimenzionalnim matricama:

```
Program                          Potprogram
DIMENSION C(4,5), D(10,10)       SUBROUTINE SUBX(X,I,J,S,T)
. . . . . .                      DIMENSION X(I,J)
V = C(1,2)                       . . . . . .
M = 4                            W = X(I,J)
N = 5                            S = . . . .
CALL SUBX (C,M,N,Q,R)            T = . . . .
. . . . . .                      RETURN
M = 10                           END
CALL SUBX(D,M,M,U,V)
. . . . . .
END
```

Prvi argument programa definiše prenos elemenata matrice iz programa u potprogram (prvim pozivom C, drugim D), druga dva argumenta definišu dimenzije matrice (4×5 odnosno 10×10), a poslednja dva argumenta vraćaju u program izračunate vrednosti.

Naredba ENTRY omogućuje ulaze u potprograme FUNCTION i SUBROUTINE i počev od drugih naredbi u tim potprogramima, a ne samo od naredbe FUNCTION odnosno SUBROUTINE na početku. Njen opšti oblik je:

ENTRY *naziv* (a_1, a_2, . . ., a_n)

gde je *naziv* jedan jedinstven simbol, usvojen po opštim pravilima za nazive funkcija i potprograma, a a_1, a_2, . . ., a_n su fiktivni argumenti. U sledećem primeru imamo jedan normalan ulaz u potprogram i dva pomoću naredbi ENTRY:

```
   Program                       Potprogram
   . . . . . .                   SUBROUTINE POT1(A,B,G,K,N)
 5 CALL POT1(U,V,W,J1,J2)        . . . . . .
 6 IF . . . .                    Y = B + G . . . .
10 CALL POT2(X,J3)               ENTRY POT2(C,N)
11 IF . . . .                    . . . . . .
```

```
      Program                    Potprogram
15 CALL POT3(Q,S,J4,T)     ENTRY POT3(B,G,K,R)
16 IF . . . .              . . . . . .
   . . . . . .             RETURN
   END                     END
```

Ulaz u potprogram moguć je na tri mesta, POT1, POT2 i POT3. Posle naredbe RETURN, akcija u programu se nastavlja uvek od naredbe koja neposredno sledi tekuću naredbu CALL. U gornjem primeru, po naredbi 10 počinje izvršavanje potprograma od naredbe ENTRY POT2 i završava se naredbom RETURN, kojom se akcija prenosi na naredbu 11 programa. Naredbom 15 ponovo se ulazi u potprogram, sada od naredbe ENTRY POT3, a naredba RETURN prenosi akciju na naredbu 16 u programu. Od naredbi ENTRY, izvršava se uvek samo ona koja je pozvana naredbom CALL iz programa, ostale su neaktivne. Zato kada po naredbi 5 bude pozvan potprogram POT1, obe naredbe ENTRY će biti neaktivne i ceo potprogram će se izvršiti normalno do kraja.

Dodeljivanjem vrednosti stvarnih argumenata simbolima fiktivnih argumenata bilo koje naredbe ENTRY, te vrednosti se odnose na ceo potprogram, kako na promenljive ispod tako i na one iznad odnosne naredbe ENTRY. Tako na primer u gornjem potprogramu, posle poziva 15 CALL POT3(Q,S,J4,T) dodeljene su vrednosti promenljivih Q, S, J4 i T promenljivima B, G, K i R. To ujedno znači da su to sada vrednosti promenljivih B i G u celom potprogramu, počev od naredbe SUBROUTINE POT1(A,B,G,K,N) pa do kraja.

Naredba RETURN i, za razliku od naredbe RETURN, omogućuje povratak iz potprograma SUBROUTINE na *određenu* naredbu programa. U tom slučaju se lista stvarnih argumenata u pozivnoj naredbi CALL kao i lista fiktivnih argumenata u naredbi SUBROUTINE pišu nešto drukčije*. Pogledajmo to na primeru:

```
      Program                    Potprogram
   . . . . . .              SUBROUTINE GAMA(I,A,*,*,K)
 2 CALL GAMA(10,X,&6,&60,I)   . . . . . .
 3 IF . . . .                 GO TO (21,22,23), K
   . . . . . .              21 RETURN 2
 6 Z = . . . .              22 . . . . . .
   . . . . . .                 RETURN 1
```

* Važi kod prevodioca H za računare IBM 360 i prevodioca za računare ICL System 4

```
Program                    Potprogram
60 DO . . . .          23 . . . . . .
. . . . . .                RETURN
END                        END
```

Izvršavanjem naredbe 2 u programu, vrednosti argumenata se prenose u potprogram GAMA. Ovde treba uočiti da su na mesta dveju zvezdica u listi fiktivnih argumenata preneta dva broja naredbi, 6 i 60, i to 6 na mesto prve, a 60 na mesto druge zvezdice. Dalji tok programa zavisiće od vrednosti argumenta K: ako je K = 1, izvršiće se naredba RETURN 2 i program će se nastaviti sa naredbom 60; ako je K = 2, izvršiće se naredba RETURN 1 i program će se nastaviti sa naredbom 6; ako je K = 3, izvršiće se naredba RETURN i program će se nastaviti normalno, sa naredbom 3.

Naredba DATA je jedna neaktivna naredba, kojom se daju početne vrednosti skalarnim i indeksnim promenljivima i celim matricama. Njen opšti oblik je:

DATA *lista* /d_1,d_2, . . ., d_n/, *lista* /d_1,d_2, . . ., d_n/, . . .

Lista može da sadrži niz simbola skalarnih i indeksnih promenljivih i matrica, odvojenih zarezima. Indeksne promenljive i matrice moraju prethodno biti dimenzionisane naredbama DIMENSION*. Indeksi mogu biti samo celobrojne konstante. Na mestima d_1, d_2, . . ., d_n pišu se konstante-početne vrednosti promenljivih iz prethodne liste, istim redom s leva na desno. Na primer:

```
DIMENSION A(3), B(3,3), L(4)
DATA A/1.0,1.0,1.0/,B/9*2.0/,L/4*T/,C/'NASLOV'/
```

Prethodnim eksplicitnim naredbama (ili po unutrašnjoj konvenciji) deklarisane su promenljive A i B kao realne, a promenljiva, L kao logička promenljiva. Vidimo da se iste vrednosti za niz promenljivih ne moraju pisati pojedinačno: dozvoljen je faktor ponavljanja, odvojen od vrednosti operatorom množenja. Tako će svi članovi matrice A dobiti početnu vrednost 1.0 (to se moglo pisati i A/3*1.0/), a svi članovi matrice B vrednost 2.0. Logička konstanta .TRUE. može se pisati i samo sa slovom T (.FALSE. sa slovom F), a literal kao vrednost promenljive piše se između apostrofa** ili sa specifikacijom *w*H, kao u naredbi FORMAT.

---

* Onima koje su dimenzionisane naredbom COMMON, početne vrednosti se ne mogu dodeliti naredbom DATA

** Vidi fusnotu za naredbu RETURN *i*

Vrednosti date promenljivima na ovaj način mogu
gramu menjati pomoću drugih naredbi. Pošto je naredb
jedna od neaktivnih naredbi (kao FORMAT, DIMENSION it
se ne može dati broj i na nju prenositi akcija u programu, da
promenljivima dale druge početne vrednosti.

Zajednička ili opšta zona u memoriji računara za dve ili v...e promenljivih ili matrica u raznim programskim jedinicama, kao što smo videli u odeljku 10.4, definiše se pomoću naredbe COMMON. Takva zona se zove blanko ili *neimenovana zajednička zona* jer sve veličine koje se u njoj nalaze nemaju u programu nikakav zajednički naziv. U FORTRAN-prevodiocima za veće računare dozvoljeno je da se, osim blanko zajedničke zone, za neke promenljive i matrice definiše jedna ili više *imenovanih zajedničkih zona*. Pogledajmo jednu naredbu COMMON, kojom je za izvesne veličine određeno zajedničko mesto u neimenovanoj, a za druge u dvema posebnim, imenovanim zonama:

```
COMMON A,B,C /ZONA1/ D,E,F // G,H /ZONA2/ P
```

Po toj naredbi, raspored veličina u memoriji biće:

```
neimenovana zona: A,B,C,G,H
ZONA1: D,E,F
ZONA2: P
```

Simbolima, koji su u ovakvoj naredbi navedeni iza dve kose crte, dodeljuju se mesta u neimenovanoj zajedničkoj zoni.

Potprogrami BLOCK DATA su specijalni potprogrami, sastavljeni isključivo od neaktivnih naredbi; pomoću tih potprograma dodeljuju se početne vrednosti samo promenljivima čija su polja u imenovanim zajedničkim zonama. Potprogram BLOCK DATA počinje jednom naredbom BLOCK DATA, bez argumenata. Zatim dolaze eksplicitne naredbe za deklarisanje, naredbe DIMENSION, COMMON i DATA i na kraju jedna naredba END. Evo jednog primera:

```
BLOCK DATA
LOGICAL LT, LF
DOUBLE PRECISION B, E
DIMENSION A(25), B(50), C(25)
COMMON /ZONA1/A,C,D /ZONA2/B,E /ZONA3/LT,LF
DATA A,D/25*0,0,1.0/,LT,LF/T,F/,E/0.135792246809D—02/
END
```

Posle izvršavanja potprograma, promenljive u imenovanim zonama imaće sledeće početne vrednosti:

ZONA1 (realne konstante)	ZONA2 (dvostruka tačnost)	ZONA3 (logičke konstante)
$A(1) = 0.0 \cdot 10^0$	B(1) neodređeno	LT = .TRUE.
. . . . .	. . . . .	LF = .FALSE.
$A(25) = 0.0 \cdot 10^0$	B(50) neodređeno	
C neodređeno	$E = 0.13579246809 \cdot 10^{-2}$	
$D = 0.1 \cdot 10^1$		

Naredba EXTERNAL primenjuje se u vezi sa potprogramima FUNCTION i SUBROUTINE. Da bi se *naziv* nekog potprograma mogao preneti kao argument u druge potprograme, on se mora definisati kao takav naredbom EXTERNAL na početku programa iz koga se prenosi. Opšti oblik naredbe EXTERNAL je:

EXTERNAL a, b, c, ...

gde su *a, b, c,* ..... nazivi potprograma, odvojeni zarezima. Upotreba naredbe EXTERNAL biće najbolje objašnjena na jednom primeru:

```
  Program
  EXTERNAL POT2, POT3
  COMMON X(100)
1 FORMAT (I4,(E14.7))
  . . . . . .
  READ (5,1) N, (A(I),I=1,N)
2 Y = POT1(POT2,N)
  . . . . . .
3 W = POT1(POT3,N)
  . . . . . .
  END
```

```
  Potprogrami
  FUNCTION POT1(ARG,M)
  COMMON E(100)
  . . . . . .
100 R = ARG(M)
  . . . . . .
101 POT1 = . . . .
  RETURN
  END
  FUNCTION POT2(K)
  COMMON F(100)
  POT2 = 0.0
  DO 3 I = 1,K
3 POT2 = POT2 + F(I)
  RETURN
  END
  FUNCTION POT3(K)
  COMMON G(100)
  POT3 = 0.0
  DO 3 I = 1,K
3 POT3 = POT3 + G(I)**2
  RETURN
  END
```

Naredbom 2 u programu poziva se potprogram POT1. Naziv drugog potprograma, POT2, i vrednost promenljive N prenose se tom prilikom kao argumenti u potprogram POT1. Izvršavanjem potprograma, nailazi naredba 100, u kojoj se traži potprogram ARG. Pošto je naziv ARG već zamenjen nazivom POT2, to se ovde ustvari poziva potprogram POT2. Posle izvršavanja celog potprograma POT2, akcija se vraća na naredbu 100 u potprogramu POT1. Kada se izvrši i potprogram POT1 do kraja, njegovom naredbom RETURN će akcija biti preneta natrag na naredbu 2 u programu.

U naredbi 3 programa ponovo se poziva potprogram POT1. Ponoviće se ceo malopređašnji proces, izuzev što se sada kao argument u potprogram POT1 prenosi naziv potprograma POT3.

Eksplicitne naredbe za deklarisanje vrste promenljivih, koje smo upoznali u odeljku 2.4, služe kod većih računara istovremeno i za dimenzionisanje matrica. Znamo već da se dimenzije matrice određuju pomoću naredbe DIMENSION, a da je to takođe moguće i pomoću naredbe COMMON, ovo poslednje samo za matrice koje će deliti iste zone sa matricama u drugim programskim jedinicama. Na primer:

```
INTEGER ALFA(7), B, C(3,6)
REAL NR(10), IKS(2,6,5), L18
COMMON ALFA, IKS
```

mogli smo napisati i na poznati način:

```
INTEGER ALFA, B, C
REAL NR, IKS, L18
DIMENSION ALFA(7), C(3,6), NR(10), IKS(2,6,5)
COMMON ALFA, IKS
```

Logičan je samo takav redosled ovih naredbi po kojem FORTRAN-prevodilac najpre dobija informaciju o *vrsti brojeva*-članova matrice, pa tek zatim informaciju o *dimenzijama* matrice, što je u gornjim primerima zadovoljeno. Ovo je važno zato što celi i obični realni brojevi zauzimaju u memoriji računara polja određene dužine, izražene brojem pozicija (taj broj je najčešće 4*), dvostruko tačni brojevi najčešće dva puta duža polja; za kompleksne brojeve i za logičke veličine postoje takođe određene dužine polja.

* Računari IBM 360, ICL System 4, CII 10070 i dr. Ove podatke za druge računare treba videti iz FORTRAN-priručnika u računskim centrima

Zato prevodilac ne može da rezerviše zonu za matricu ako već prethodno ne raspolaže podatkom o dužini polja za jedan njen član.

Za one FORTRAN-prevodioce, koji dozvoljavaju ovakvu upotrebu eksplicitnih naredbi za deklarisanje vrste, dozvoljeno je pisanje atributa vrste u definicionim naredbama aritmetičkih funkcija i potprogramâ FUNCTION. Izračunata vrednost biće u tom slučaju pretvorena u oblik deklarisane vrste. Na primer, da bi vrednost aritmetičke funkcije X1A bila celobrojna, možemo alternativno napisati sledeće naredbe:

```
INTEGER X1A
X1A(A,B) = A + B*X**2.35
```

ili

```
INTEGER X1A(A,B) = A + B*X **2.35
```

VEŽBE

* 1. Napisati definicionu naredbu za aritmetičku funkciju:

$$f(x) = x^2 + \sqrt{1 + 2x + 3x^2}$$

a zatim pomoću te funkcije programirati izračunavanje vrednosti sledećih izraza:

$$a = \frac{5.8 + y}{y^2 + \sqrt{1 + 2y + 3y^2}}$$

$$b = \frac{3.2z + z^4}{z^2 + \sqrt{1 + 2z + 3z^2}}$$

$$c = \frac{\sin y}{y^4 + \sqrt{1 + 2y^2 + 3y^4}}$$

$$d = \frac{1}{\sin^2 y + \sqrt{1 + 2\sin y + 3\sin^2 y}}$$

2. Napisati definicionu naredbu za aritmetičku funkciju zapremine loptinog odsečka (kalote):

$$V = h^2 \left(R - \frac{h}{3} \right)$$

gde je h visina odsečka, a R poluprečnik lopte, a zatim pomoću te funkcije programirati izračunavanje zapremine za deset parova vrednosti h i R.

3. Analizirati sledeće delove programâ, u kojima su upotrebljene razne aritmetičke funkcije, i ispraviti one koji su pogrešno napisani:

a.
```
1 FORMAT (2E14.7)
ROM(0.5,C,D) = 0.5*C*D
READ (5,1) D1, D2
ROM(0.5,C,D) = 0.5*C*D
```

b.
```
   DIMENSION A(100), B(100)
1  FORMAT (2E14.7)
   RAZ(A(I),B(I)) = ABS(A(I)—B(I))
   DO 8 I = 1,100
   READ (5,1) A(I), B(I)
8  X(I) = RAZ(A(I),B(I))**3
```

c.
```
1 FORMAT (2E14.7)
READ (5,1) A, B, C
S(A,B) = 0.5*(A + B + C)
P = S(A,B)
```

d.
```
1 FORMAT (2E14.7)
RAD(STEP) = STEP*3.14159/180.0
ALUK(R,TETA) = R*TETA
READ (5,1) UGSTEP, POLUPR
```

```
      Y = SIN(RAD(UGSTEP))
      X = ALUK(POLUPR,RAD(UGSTEP))
e.  1 FORMAT (2E14.7)
      CS = 3.14159*R*SQRT(R*R + H*H)
      READ (5,1) R, H
      P = CS(R,H)
```

* 4. Napisati jedan potprogram **FUNCTION** za izračunavanje funkcije:

$$y(x) = \begin{cases} 1 + \sqrt{1 + x^2} & \text{za } x < 0 \\ 0 & \text{za } x = 0 \\ 1 - \sqrt{1 + x^2} & \text{za } x > 0 \end{cases}$$

a zatim pomoću tog potprograma napisati aritmetičke naredbe za izračunavanje sledećih veličina:

$$f = 2 + y(a + z)$$

$$g = \frac{y[x(k)] + y[x(k+1)]}{2}$$

$$h = y[\cos(2x)] + \sqrt{1 + y(2\pi x)}$$

5. Napisati jedan potprogram **FUNCTION** za izračunavanje funkcije:

$$TAU(A,B,N) = \frac{A}{2\pi} \sum_{i=1}^{N} Bi$$

gdje je *B* jednodimenzionalna matrica sa najviše 50 članova ($N \leqslant 50$). Taj potprogram upotrebiti za izračunavanje vrednosti STR, koja je jednaka ½ puta vrednost zbira prvih 36 članova matrice A. Program i potprogram napisati tako da se koriste matrice promenljivih dimenzija.

* 6. Napisati jedan potprogram **FUNCTION** sa argumentima A, M i N, gde je A simbol jedne dvodimenzionalne matrice, M broj redova, a N broj kolona. Vrednost funkcije treba da bude zbir apsolutnih vrednosti svih članova matrice A. Dimenzije matrice treba da budu promenljive.

* 7. A je simbol jedne jednodimenzionalne matrice sa 50 članova. Napisati jedan potprogram **SUBROUTINE** za izračunavanje srednje vrednosti (proseka) prvih N članova i broja tih članova čija je vrednost nula. Za naziv i argumente potprograma usvojiti simbole PROSEK(A,N,SRED,NULA).

8. Napisati potprogram **FUNCTION** ZBIR(REDKOL,L,A,M,N), po kome se izračunava algebarski zbir svih vrednosti *reda L* ili *kolone L* neke matrice promenljivih dimenzija M x N.

* 9. Napisati potprogram **SUBROUTINE** koji, koristeći potprogram **FUNCTION** iz prethodne vežbe, nalazi *red* matrice M x N sa najvećom vrednošću algebarskog zbira članova; izlazne veličine su broj reda i vrednost algebarskog zbira. Sa definisanim potprogramom **SUBROUTINE** naći među redovima matrice PSI (15,29) broj reda (stvarni argument NRED) sa najvećom vrednošću algebarskog zbira članova (stvarni argument PSIL). (Promenljive dimenzije matrice prenose se kao argumenti iz jednog potprograma u drugi!).

Glava 11

PRIMERI FORTRAN-PROGRAMA

11.1 *DIFERENCIJALNE JEDNAČINE PRVOG REDA*

Metoda Runge-Kutta je poznata metoda za numeričko rešavanje diferencijalnih jednačina prvog reda, na bazi aproksimiranja funkcije pomoću Tejlorovog (Taylor) reda. Nasuprot egzaktnom izračunavanju članova Tejlorovog reda, po ovoj metodi se ne izračunavaju izvodi višeg reda od prvog. Osim toga, za početak računanja dovoljna je samo jedna vrednost funkcije, ostale se dobijaju računom sukcesivno. Veličina intervala nezavisno promenljive može se menjati u toku procesa. Po metodi, koja je upotrebljena u programu, greška je reda h^4, gde je h interval nezavisno promenljive. Iako bi se, prema tome, greška mogla smanjiti usvajanjem manjeg intervala, time se ujedno povećava broj iteracija, pa i greška usled zaokruživanja.

Postoji više varijanata metode Runge-Kutta. Program je rađen na osnovi tzv. klasične varijante: ako je data diferencijalna jednačina prvog reda:

$$\frac{dy}{dx} = f(x,y)$$

sa početnim vrednostima x_0 i y_0, sledeća vrednost funkcije se dobija po obrascu:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

gde je:

$$k_1 = h \cdot f(x_n, y_n)$$

$$k_2 = h \cdot f(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$$

$$k_3 = h \cdot f(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$$

$$k_4 = h \cdot f(x_n + h, y_n + k_3)$$

Za ilustraciju, programirano je rešavanje diferencijalne jednačine:

$$\frac{dy}{dx} = x - y^2$$

sa početnim uslovima $x_0 = 0$ i $y_0 = 0$. Kao ulazne veličine, programu se daju početna i krajnja vrednost nezavisno promenljive (XP i XK), interval H i početna vrednost funkcije, YP. Na slikama 11.1 i 11.2 vidimo dijagram toka i FORTRAN-program. Na slici 11.3 prikazani su rezultati rada programa za interval H=0.05 i krajnju vrednost nezavisno promenljive XK=1.0.

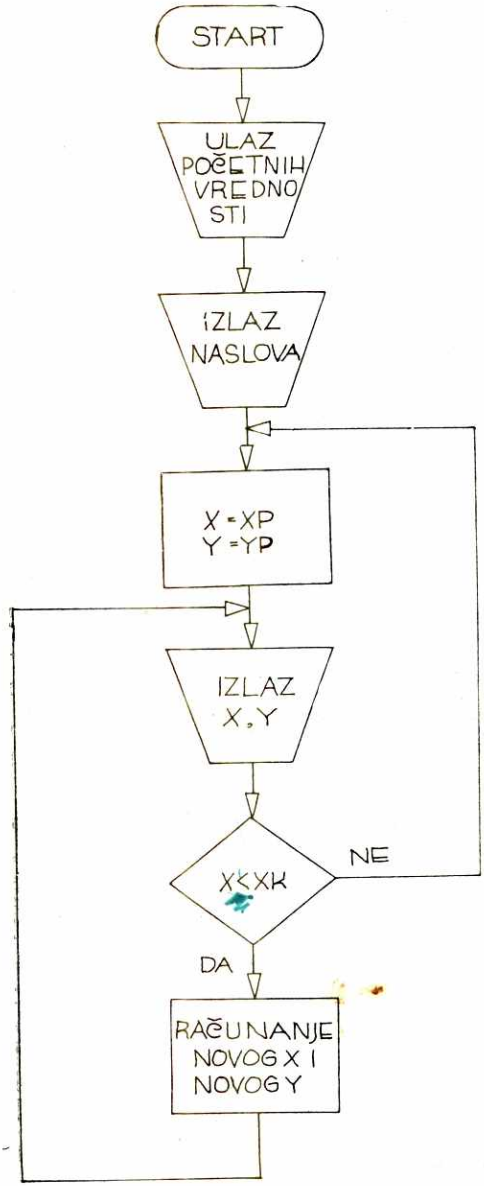
Kada program završi sa izračunavanjem funkcije prema zadatim ulaznim podacima, on se ne zaustavlja na normalan način (naredba STOP) nego traži nove ulazne podatke. Videli smo da ovakav način zaustavljanja programa, iako moguć, nije pravilan sa gledišta iskorišćenja računara (odeljak 5.6). Kako bi trebalo promeniti dijagram, a zatim i program, pa da operator računara dobije poruku o završetku rada programa i da upravljački program računara nastavi sa izvođenjem sledećeg programa?

11.2 *NJUTNOV ITERACIONI OBRAZAC*

Njutnov (Newton) iteracioni obrazac služi za približno numeričko traženje nulâ realnih funkcija jedne promenljive. Ako je data neka funkcija od x u obliku $f(x) = 0$, ovaj metod se sastoji u tome da, polazeći od nekog približnog rešenja x_i, daje još približnije rešenje x_{i+1} pomoću obrasca:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Na primer, neka je data funkcija $f(x) = x^2 - 25$. Kao prvo približno rešenje uzmimo $x_0 = 2$. Pošto je $f'(x) = 2x$, drugo približno



Slika 11.1 Dijagram za rešavanje diferencijalne jednačine

```
      PROGRAM RUNGE-KUTTA
      DY/DX = X-Y**2
      X(0)=0, Y(0)=0
      F(X,Y) = X - Y*Y
    1 READ(5,2)H,YP,XP,XK
    2 FORMAT (4F20.7)
      WRITE(6,3)
    3 FORMAT (4X,'METODA RUNGE-KUTTA'/
     11H ,'RESENJE ZA  DY/DX=X-Y**2'/
     21H ,4X,1HX,14X,1HY/)
      X = XP
      Y = YP
    5 WRITE(6,4)X,Y
    4 FORMAT (F10.7,F15.7)
      IF (X-XK) 6,1,1
    6 RK1 = F(X,Y)
      RK2 = F(X+0.5*H,Y+0.5*H*RK1)
      RK3 = F(X+0.5*H,Y+0.5*H*RK2)
      RK4 = F(X+0.5*H,Y+H*RK3)
      YRK = (RK1+2.0*(RK2+RK3)+RK4)/6.0
      X = X + H
      Y = Y + H*YRK
      GO TO 5
      END
```

Slika 11.2 Program za rešavanje diferencijalne jednačine

rešenje biće:

$$x_1 = x_0 - \frac{x_0^2 - 25}{2x_0} = 2 - \frac{4 - 25}{4} = 7.25$$

Ponavljajući sada isti postupak po gornjem opštem obrascu, dobićemo niz približnih rešenja:

$x_0 = 2$
$x_1 = 7.25$
$x_2 = 5.35$
$x_3 = 5.0114$
$x_4 = 5.00001$
$x_5 = 5.0000000$

```
   METODA RUNGE-KUTTA
RESENJE ZA  DY/DX=X-Y**2
     X                Y

0.0              0.0
0.0500000        0.0010416
0.1000000        0.0045829
0.1499999        0.0106216
0.1999999        0.0191520
0.2499998        0.0301629
0.2999998        0.0436361
0.3499997        0.0595441
0.3999997        0.0778489
0.4499996        0.0985007
0.4999996        0.1214364
0.5499995        0.1465784
0.5999995        0.1738346
0.6499994        0.2030977
0.6999994        0.2342454
0.7499993        0.2671409
0.7999993        0.3016343
0.8499992        0.3375635
0.8999992        0.3747565
0.9499992        0.4130334
0.9999991        0.4522088
1.0499983        0.4920948
```

Slika 11.3 Rešenje diferencijalne jednačine (interval $h = 0.05$, krajnja vrednost nezavisno promenljive $x_k = 1.0$)

Približno rešenje na kojem ćemo se zaustaviti zavisi od tačnosti koju želimo. Taj uslov se obično postavlja tako što se zadaje određena, obično vrlo mala razlika između dva uzastopna približna rešenja.

Program, koji ćemo napisati po ovoj metodi, nalazi kvadratni koren pozitivnog realnog broja. Za taj slučaj, Njutnov obrazac glasi:

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{A}{x_i}\right)$$

gde je A proizvoljni pozitivan realni broj čiji kvadratni koren tražimo, x_i tekuća aproksimacija, a x_{i+1} sledeća aproksimacija kvadratnog korena tog broja.

Uslov tačnosti ćemo ovde postaviti tako da se iteracija prekida kada bude:

$$\left| \frac{(x_{i+1})^2 - A}{A} \right| < t$$

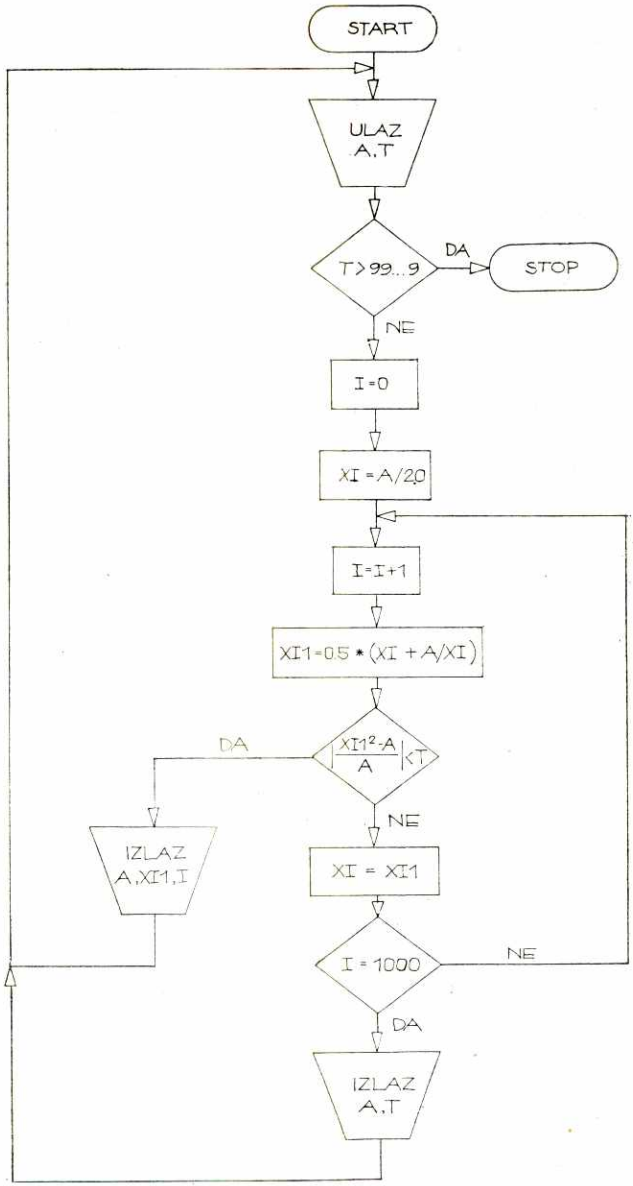
Oba broja, A i t, nalaze se u jednom ulaznom slogu, gde imaju oblik realnih brojeva sa izložiocem E, a zauzimaju prva dva polja po 14 kolona. Pošto program treba da daje kvadratne korene za neograničen broj pozitivnih realnih brojeva, sa posebno zadatom tačnošću za svaki pojedini broj, to će na kraju ulazne datoteke biti jedan slog koji u drugom polju ima neku konstantu veću od 9.E10 (slika 11.4).

```
0.0400000E 02 0.0001000E 00
0.0900000E 02 0.0001000E 00     C
0.1600000E 02 0.0001000E 00     C
0.2500000E 02 0.0001000E 00     C
0.3600000E 02 0.0001    E 25
```

Slika 11.4 Ulazni podaci za program sa slike 11.6

Dijagram toka za taj program dat je na slici 11.5. Rezultat rada programa je jedan izlazni slog od tri polja: prvo i drugo polje sadrže broj i njegov kvadratni koren kao realne brojeve u obliku sa izložiocem E, a treće polje broj izvršenih iteracija (sl. 11.7). FORTRAN-program prikazan je na slici 11.6.

Prvom naredbom IF otkriva se slog na kraju datoteke, kojim se završava rad programa. Početna aproksimacija je polovina datog realnog broja. Programom je predviđeno maksimalno 1000 iteracija, što je veoma ekstreman slučaj; ako se ipak ne postigne željena tačnost, u izlazni slog će izaći zadati broj A i veličina t. U normalnom slučaju, kada se postigne zadata tačnost, doći će do izlaza jednog sloga sa zadatim brojem A, njegovim kvadratnim korenom i brojem izvršenih iteracija. Posle toga se nastavlja rad programa ulazom novog sloga iz ulazne datoteke.



Slika 11.5 Dijagram za kvadratni koren pozitivnog realnog broja

```
C       PROGRAM ZA KVADRATNI KOREN POZITIVNOG
C       REALNOG BROJA
      2 FORMAT (2E14.7,I4)
      1 READ (5,2) A,T
        IF (T.GE.0.9999999E10) GO TO 8
        XI = A/2.0
        DO 5 I=1,1000
        XI1 = 0.5*(XI+A/XI)
        IF (ABS(XI1**2/A-1.0).LT.T) GO TO 6
      5 XI = XI1
        WRITE (6,2) A,T
        GO TO 1
      6 WRITE (6,2) A,XI1,I
        GO TO 1
      8 STOP 666
        END
```

Slika 11.6 Program za kvadratni koren pozitivnog realnog broja

```
0.4000000E 01 0.2000000E 01    1
0.9000000E 01 0.3000015E 01    3
0.1600000E 02 0.4000000E 01    4
0.2500000E 02 0.5000012E 01    4
0.3600000E 02 0.6000000E 01    5
```

Slika 11.7 Rezultati programa sa slike 11.6 (za ulazne podatke sa slike 11.4)

11.3 *LINEARNA INTERPOLACIJA*

U praktičnim proračunima je čest slučaj da analitički izraz neke funkcije jedne nezavisno promenljive, koja se vrlo često upotrebljava, nije poznat nego je data tablica sa parovima vrednosti funkcije i nezavisno promenljive-argumenta. Takve tablice su obično rezultat empiričkog ispitivanja neke fizičke pojave, za koju analitički izraz ne postoji ili je vrlo složen, tako da je upotreba funkcije u daljim proračunima jedino moguća u ovom obliku.

Tablice ovakvih funkcija nikada ne sadrže sve vrednosti funkcije koje u proračunima mogu biti potrebne, pa se zato pribegava

interpolaciji. Mi ćemo ovde ostaviti po strani slučajeve kada je funkcija prekidna ili kada se njena aproksimacija mora vršiti nekim polinomom višeg stepena. Zadatak će biti da se napiše FORTRAN-program za linearnu interpolaciju tablično date funkcije, što u praksi vrlo često predstavlja zadovoljavajuće rešenje. Ovakav program bi se mogao napisati i kao potprogram, koji posle može služiti kao pomoćno sredstvo raznim programima, u kojima postoji potreba za linearnom interpolacijom.

Pretpostavićemo da su redovi tablice, u kojima su parovi vrednosti argumenta i funkcije (x i y), dati u slogovima i to tako da je u svakom slogu jedan par, u obliku koji odgovara za ulaz u program po specifikaciji 2F10.0. Slogova ima najviše 200, a uređeni su po rastućim vrednostima argumenta x. Intervali između dva susedna argumenta ne moraju biti jednaki. Na kraju datoteke tabličnih vrednosti nalazi se jedan slog koji, osim vrednosti x i y, ima u svojoj 21. koloni jednu cifru različitu od nule (slika 11.8).

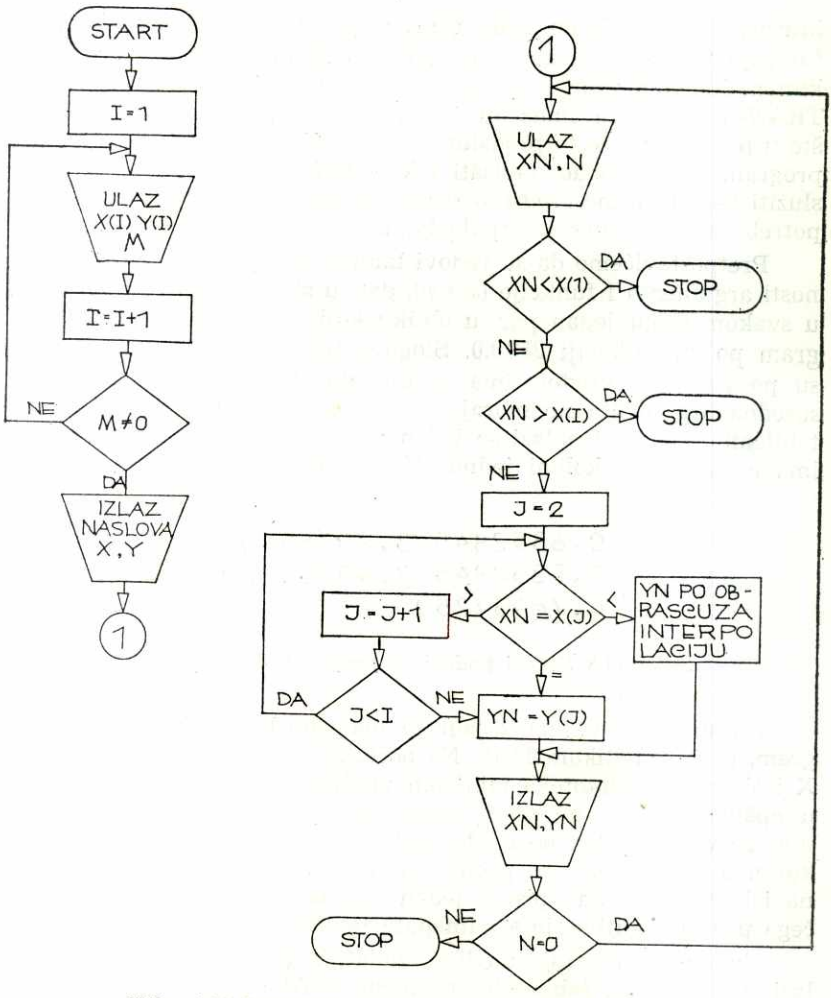
```
0.6352845 3.2976540
0.8352845 3.49765409
0.76437259
```

Slika 11.8 Ulazni podaci za program sa slike 11.10

Na slici 11.9 vidimo dijagram toka, po kojem je napisan program, prikazan slikom 11.10. Na početku programa, u zone matrice X i Y unosi se datoteka tabličnih vrednosti funkcije. Pošto tablica u opštem slučaju ne mora imati tačno 200 redova nego manje, ulaz se vrši sve dok ne naiđe poslednji par tabličnih vrednosti, sa kojim ulazi vrednost za promenljivu M, različita od nule. Tada se na izlazni medijum upisuje jedan slog sa naslovima (X, Y), posle čega počinje ispitivanje argumenata tablice.

Najpre se utvrđuje da li se zadati argument (XN) uopšte nalazi u granicama tabličnih argumenata. Ako je on manji ili veći od graničnih vrednosti, program će se zaustaviti, dajući za oba slučaja istu poruku. Pošto je upotrebljena naredba STOP, upravljački program računara će prekinuti dalji rad ovog programa i on se više ne može nastaviti.

Ako je vrednost XN u opsegu tabličnih vrednosti, on će u ciklusu DO 7 biti sistematski upoređivan sa tabličnim argumentima, počev od drugog pa sve do prvog većeg ili jednakog. Ako se upo-



Slika 11.9 Dijagram toka za linearnu interpolaciju

ređivanje zaustavi kod prvog većeg argumenta, izračunavanje interpolisane vrednosti funkcije vršiće se po obrascu (naredba 5):

$$y_n = y_{j-1} + \frac{y_j - y_{j-1}}{x_j - x_{j-1}} (x_n - x_{j-1})$$

```
C       PROGRAM ZA LINEARNU INTERPOLACIJU
        DIMENSION X(200),Y(200)
        DO 1 I=1,200
        READ (5,2) X(I),Y(I),M
    2   FORMAT (2F10.0,I1)
    1   IF(M.NE.0) GO TO 10
   10   WRITE (6,4)
    4   FORMAT (8X,1HX,16X,1HY/)
   11   READ (5,3) XN,N
    3   FORMAT (F10.0,I1)
        IF(XN.LT.X(1)) STOP 333
        IF(XN.GT.X(I)) STOP 333
        DO 7 J=2,I
        IF(XN-X(J)) 5,6,7
    7   CONTINUE
    6   YN = Y(J)
        GO TO 8
    5   YN = Y(J-1)+(Y(J)-Y(J-1))/
       1(X(J)-X(J-1))*(XN-X(J-1))
    8   WRITE (6,9) XN,YN
    9   FORMAT (1H ,E14.7,E17.7)
        IF(N.EQ.0) GO TO 11
        STOP 777
        END
```

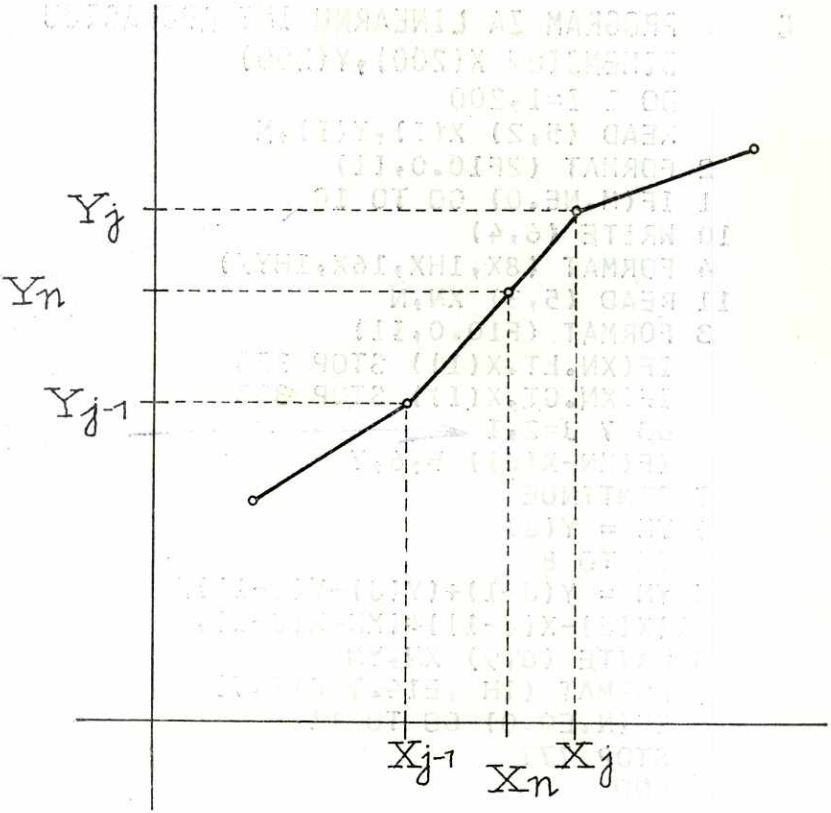
IF na kraju DO-ciklusa

poslednje I

Slika 11.10 Program za linearnu interpolaciju

Taj postupak je prikazan dijagramom na slici 11.11. Ako se upoređivanje zaustavi na argumentu koji je jednak zadatom, tada se po naredbi 6 direktno uzima odgovarajuća vrednost funkcije. Još pre ulaska u ciklus DO 7 isključena je mogućnost da zadati argument, posle upoređivanja sa svim tabličnim argumentima, bude još uvek veći. U dijagramu toka, izlaz NE iz pretposlednjeg romba ($J < I$) mogao je biti upućen bilo gde: do tog izlaza neće nikada doći.

Kada nađe odgovarajuću vrednost funkcije na jedan ili drugi način, program upisuje vrednost zadatog argumenta i pripadajuću vrednost funkcije kao jedan slog na izlaznom medijumu (slika



Slika 11.11 Grafički prikaz linearne interpolacije

11.12). Na sličan način kao pri unošenju tabličnih vrednosti, program zatim ispituje da li ima još zadatih argumenata; ako ima (N=0), ponavlja se gore opisani proces od naredbe 11, a ako nema (N≠0), program se završava sa porukom o normalnom kraju posla (STOP 777).

```
          X                    Y

 0.7643725E 00      0.3426742E 01
```

Slika 11.12 Rezultati programa sa slike 11.10 (za ulazne podatke sa sl. 11.8)

11.4 *DISTRIBUCIJA FREKVENCIJA*

Distribucija frekvencija je pojam iz matematičke statistike. Distribucije frekvencija se često upotrebljavaju za izradu sumarnih pregleda jedinica posmatranja, grupisanjem prema onim njihovim obeležjima koja se ne moraju pojedinačno prikazivati. Na primer, ako se radi o težinama lica, sva lica sa težinama između 61 i 65 kilograma mogu se grupisati zajedno jer za mnoge praktične svrhe nije potrebno praviti razliku između lica čije su težine 61.5, 62.5, 64 itd. kilograma. Jedinice posmatranja gube svoju individualnost grupisanjem, a daje se samo njihov broj-frekvencija po intervalima. Takav sumarni pregled naziva se *distribucija frekvencija,* a intervali se zovu *grupni* ili *klasni intervali.* Postoji nekoliko opštih pravila za formiranje distribucija frekvencija: (a) grupni intervali ne smeju se poklapati niti uzajamno isključivati u pogledu vrednosti obeležja, (b) ukoliko je to moguće, intervali treba da budu jednaki, (c) broj intervala ne treba da bude ni suviše veliki ni suviše mali, što zavisi od veličine mase (broja jedinica posmatranja), (d) sredine intervala treba da budu u tačkama nagomilavanja frekvencija, ako takve postoje, i (e) ekstremne veličine, bilo suviše velike ili suviše male, treba staviti u krajnje otvorene intervale oblika »manje od« i »veće od«.

Zadatak je da se napiše FORTRAN-program za formiranje distribucije frekvencija po jednakim intervalima obeležja. Pretpostavlja se da je iz prethodne kontrole posmatranih obeležja poznat broj jedinica (N), kao i najmanja i najveća vrednost obeležja. Na osnovu toga određen je broj intervala (NINT), širina intervala (DINT) i donja granica prvog intervala koji nije otvoren (XINT(1)). Sve te veličine unose se u program kao podaci. Program treba da prima do 250 jedinica posmatranja ($N \leqslant 250$), a da formira distribucije sa najviše 25 intervala ($NINT \leqslant 25$).

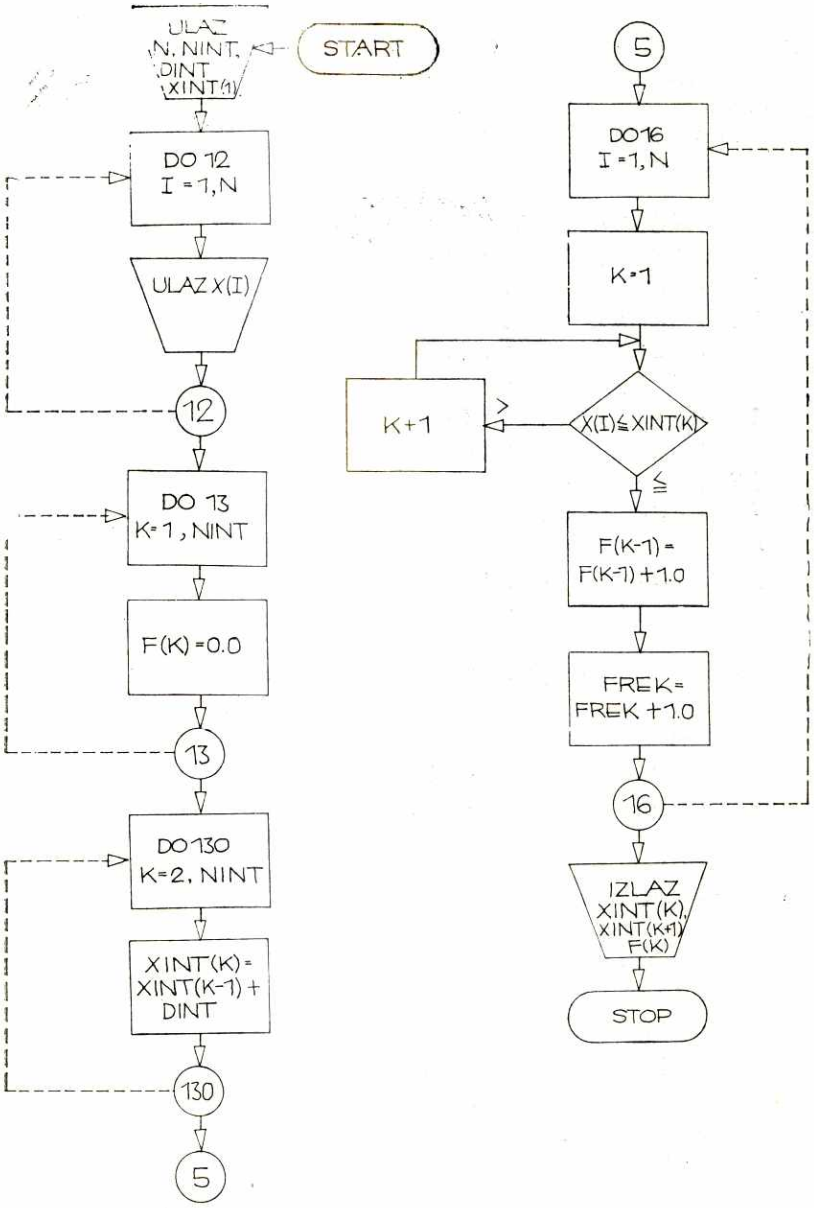
Dijagram toka, koji je dat na slici 11.14, objašnjava ostale detalje zadatka i ujedno ilustruje usvojeno rešenje. Na početku, čita se jedan slog sa vrednostima za ukupan broj jedinica (N), broj intervala (NINT), širinu intervala (DINT) i donju granicu prvog intervala koji nije otvoren (XINT(1)). Zatim ulazi N vrednosti obeležja na osnovu kojih se distribuiraju jedinice posmatranja (X(I)), pri čemu je u ovom rešenju usvojeno da su te pojedinačne vrednosti u posebnim slogovima (karticama); promenom naredbe 11 FORMAT može se rešiti da više vrednosti budu u jednom slogu. Posle dodeljivanja vrednosti nula svim članovima matrice F (frekvencije po intervalima) i promenljivoj FREK (zbir svih frekvencija), izra-

čunavaju se donje granice svih intervala (XINT(K)), koje će biti potrebne za formiranje distribucije frekvencija po intervalima. Ciklus naredbe DO 16 ispituje kojem intervalu pripada svaka pojedina vrednost i na osnovu toga kumulira frekvencije po interva-

```
02608015.0130.95
 137.0
 137.2
 142.5
 148.3
 149.2
 149.2
 149.3
 153.0
 153.2
 154.6
 154.6
 155.0
 155.0
 155.4
 157.6
 158.3
 161.4
 162.6
 164.7
 166.8
 168.7
 172.0
 177.8
 178.6
 225.0
 230.0
```

Slika 11.13 Ulazni podaci za program sa slike 11.15

lima i zbir frekvencija. Prvom naredbom WRITE upisuje se na izlazni medijum samo naslov; nad dvema kolonama brojeva, koje će predstavljati donje i gornje granice intervala, upisuju se reči



Slika 11.14 Dijagram za distribuciju frekvencija

DONJA i GORNJA, a iznad ovih, kao zajednički naslov za obadve, reč GRANICA. Posle izlaza naslova u dva reda (dva sloga), izlaze jedan za drugim slogovi sa po tri vrednosti: donja granica intervala, gornja granica intervala i vrednost frekvencije za taj interval. Poslednjom naredbom WRITE izlazi, u koloni FREKVENCIJA,

```
C       PROGRAM ZA DISTRIBUCIJU FREKVENCIJA
        DIMENSION XINT(25),X(250),F(25)
   10   FORMAT (I3,I2,F5.1,F6.2)
   11   FORMAT (F5.1)
        READ(5,10)N,NINT,DINT,XINT(1)
        DO 12 I=1,N
   12   READ (5,11)X(I)
        DO 13 K=1,NINT
   13   F(K) = 0.0
        FREK = 0.0
        DO 130 K=2,NINT
  130   XINT(K) = XINT(K-1) + DINT
        DO 16 I=1,N
        K = 1
  131   IF (X(I).LE.XINT(K)) GO TO 15
        K = K + 1
        GO TO 131
   15   F(K-1) = F(K-1) + 1.0
   16   FREK = FREK + 1.0
        WRITE (6,20)
   20   FORMAT (1H ,10X,'GRANICA'/1H ,5X,
       1'DONJA',7X,'GORNJA',5X,'FREKVENCIJA'/)
        L = NINT - 1
        DO 21 K=1,L
   21   WRITE (6,22) XINT(K),XINT(K+1),F(K)
   22   FORMAT (1H ,F10.2,F13.2,F11.0)
        WRITE (6,23) FREK
   23   FORMAT (1H ,22X,F12.0)
        STOP 777
        END
```

Slika 11.15 Program za distribuciju frekvencija

zbir svih frekvencija, koji može poslužiti kao kontrola da su u intervale razvrstane sve jedinice (FREK = N). Nije predviđena mogućnost da program posle toga počne automatski ispočetka, sa novim ulaznim podacima. To je, međutim, vrlo jednostavno i prepušta se zainteresovanom čitaocu da sam uradi.

Na slici 11.13 prikazana je jedna grupa ulaznih podataka, pripremljenih prema zahtevu ulaznih naredbi programa, na slici 11.15 sam FORTRAN-program, a na slici 11.16 tabela sa rezultatima za zadate ulazne podatke.

GRANICA		
DONJA	GORNJA	FREKVENCIJA
130.95	145.95	3.
145.95	160.95	13.
160.95	175.95	6.
175.95	190.95	2.
190.95	205.95	0.
205.95	220.95	0.
220.95	235.95	2.
		26.

Slika 11.16 Rezultati programa sa slike 11.15 (za ulazne podatke sa slike 11.13)

11.5 *PRETHODNA KONTROLA STATISTIČKIH PODATAKA*

U istraživačkom radu se često vrši dugoročno prikupljanje podataka o nekoj pojavi. Podaci se obično prikupljaju sa namerom da se kasnije primenom matematičko-statističkih metoda donesu izvesni zaključci o posmatranoj pojavi. U toku prikupljanja se, međutim, po pravilu ne posvećuje dovoljna pažnja tačnosti snimanja ili šifriranja podataka; vrlo često se podaci buše u kartice bez naročite kontrole. Na taj način dolazi kasnije do grešaka, koje isključivo potiču iz faze pripreme podataka. Pored toga, podaci često ostanu nepotpuni i tada se postavlja pitanje kako treba postupati sa nepotpunim materijalom.

Zadatak prethodne kontrole podataka je, prema tome, da otkrije izostavljene podatke i greške u šifriranju i bušenju, što bi se inače sve odrazilo na tačnost zaključaka u kasnijoj analizi. Statističko-analitičke metode su jako zavisne od tačnosti i potpunosti

podataka; netačan i nepotpun materijal dovodi do pogrešnih rezultata u analizi. Dešava se da se netačni rezultati rđavo interpretiraju, što nužno dovodi do pogrešnih zaključaka. U nekim slučajevima se može posumnjati da su netačni rezultati posledica usvojene analitičke metode, pa se uzalud gubi vreme na isprobavanju metoda koje će dati verodostojne rezultate. Greške u originalnom materijalu ostaju, međutim, neotkrivene i zato se nepotrebno troši vreme računara.

Da se izbegnu takve teškoće, pouzdanost podataka se može proveriti prethodnom kontrolom, još pre početka stvarne statističke analize. Ta kontrola ne mora da bude komplikovana, radi se samo o nekoliko prostih operacija, sa kojima treba da se ustanovi:

— da li izvesni podaci nedostaju,
— da li su i koji podaci pogrešno šifrirani,
— broj posmatranja u kojima je vrednost promenljive različita od nule,
— srednja vrednost i standardna devijacija niza vrednosti koje su različite od nule (kao prethodni podatak za kasniju statističku analizu),
— za svaku promenljivu (obeležje) broj onih čije vrednosti padaju u određene granice,
— priprema za prikazivanje podataka u vidu histograma,
— izrada pojedinačnog pregleda onih ulaznih podataka koji sadrže greške,
— mogućnost da se podaci razvrstaju u nekoliko grupa.

Radi izrade dijagrama toka, koji vidimo na slici 11.18, najpre ćemo zadatak formulisati u nekoliko tačaka:

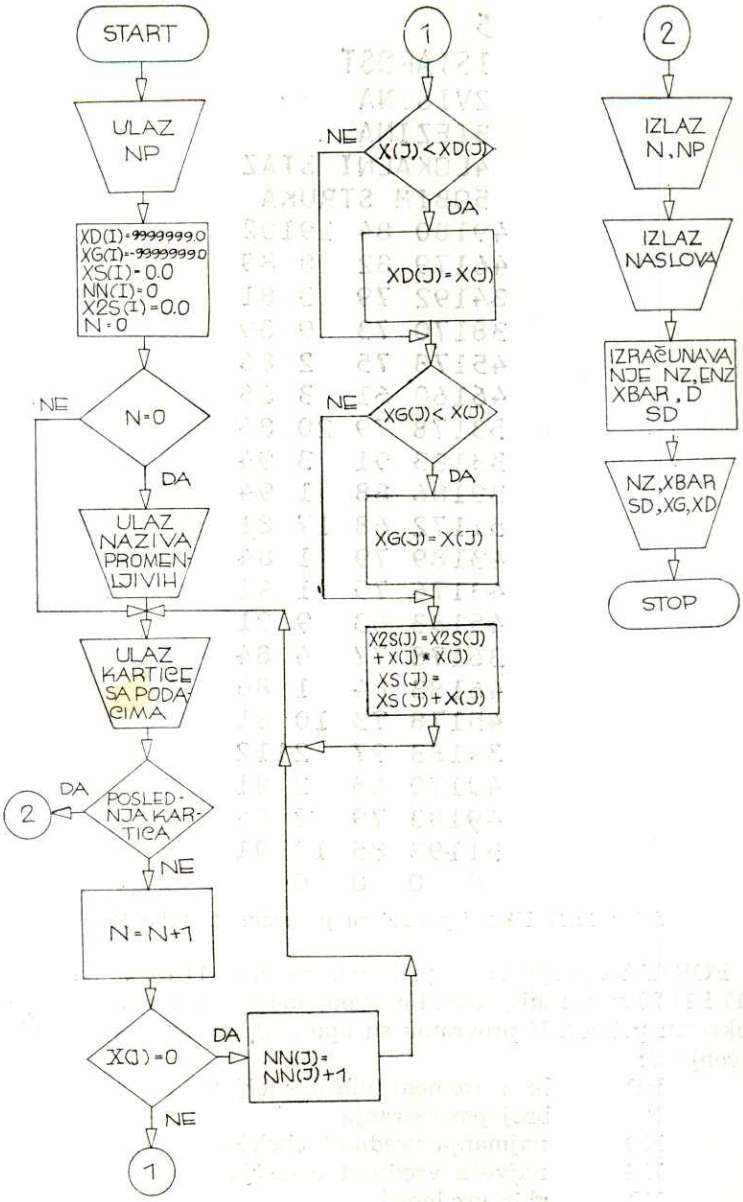
1. iz prve kartice treba da se pročita podatak o broju promenljivih (obeležja), koje će se programom obrađivati;
2. iz sledećih kartica čitaće se nazivi promenljivih; svaka kartica ove grupe treba da sadrži naziv jedne promenljive;
3. posle grupe kartica sa nazivima čitaće se kartice sa vrednostima promenljivih u jednom posmatranju; poslednja kartica ove grupe treba da bude jedna blanko kartica;
4. za svaku promenljivu treba odrediti najveću i najmanju vrednost, broj posmatranja sa vrednošću različitom od nule, srednju vrednost i standardnu devijaciju;
5. na izlazni medijum treba upisati broj posmatranja i broj promenljivih, a zatim potrebne naslove i vrednosti iz tačke 4.

```
 5
 1STAROST
 2VISINA
 3TEZINA
 4LOKALNI STAZ
 5OBIM STRUKA
49180 84 19102
46179 82  8 89
34192 79  3 81
38170 73  9 89
45178 75  2 86
46160 61  3 66
53178 79 20 86
33183 91  3 94
39186 88  1 94
51172 68 17 81
43189 79  1 84
43176 75  1 91
45183 83  9 91
36178 77  4 84
46183 84  1 86
45178 73 10 81
34183 97  2112
40170 68  2 81
49183 79  2 84
51193 85 10 91
 0  0  0  0
```

Slika 11.17 Ulazni podaci za program sa slike 11.19

FORTRAN-program je prikazan na slici 11.19, a na slikama 11.17 i 11.20 su ulazni podaci i odštampana tabela rezultata za jedan konkretan primer. U programu su upotrebljeni simboli sa sledećim značenjima:

NP	broj promenljivih (obeležja)
N	broj posmatranja
XD	najmanja vrednost obeležja
XG	najveća vrednost obeležja
XS	zbir vrednosti



Slika 11.18 Dijagram za prethodnu kontrolu podataka

```
C      PROGRAM ZA PRETHODNU KONTROLU
C      PODATAKA
       DIMENSION XD(99),XG(99),XS(99),
      1NN(99),X2S(99),B(99,3),X(99)
       READ (5,20)NP
   20  FORMAT (I2)
       N = 0
       DO 1 I=1,NP
       XD(I) =  9999999.0
       XG(I) = -9999999.0
       XS(I) = 0.0
       NN(I) = 0
    1  X2S(I) = 0.0
       IF (N) 5,3,5
    3  DO 2 I=1,NP
    2  READ (5,4)(B(I,J),J=1,3)
    4  FORMAT (2X,3A4)
    5  READ(5,6)(X(J),J=1,NP)
    6  FORMAT (F2.0,4F3.0)
       IF (X(1)+X(2)+X(3)+X(4)) 7,8,7
    7  N = N+ 1
       DO 9 J=1,NP
       IF (X(J)) 10,11,10
   10  IF (X(J)-XD(J)) 12,13,13
   12  XD(J) = X(J)
   13  IF (XG(J)-X(J)) 14,15,15
   14  XG(J) = X(J)
   15  X2S(J) = X2S(J) + X(J)*X(J)
       XS(J) =XS(J) + X(J)
    9  CONTINUE
       GO TO 5
   11  NN(J) = NN(J) + 1
       GO TO 9
    8  WRITE (6,16) N,NP
   16  FORMAT (19H BROJ POSMATRANJA =,
      1I3,23H,  BROJ PROMENLJIVIH = ,I2///)
       WRITE (6,17)
   17  FORMAT (10X,8HOBELEZJE,9X,8HRAZLICI-/
      128X,5HTO OD,13X,10HSTANDARDNA/8X,
      23HBR.,3X,5HNAZIV,9X,4HNULE,4X,7HSREDINA,
      33X,10HDEVIJACIJA,4X,3HMAX,7X,3HMIN//)
       DO 18 J=1,NP
       NZ = N - NN(J)
       ENZ = NZ
       XBAR = XS(J)/ENZ
       D = X2S(J) - XS(J)*XS(J)/ENZ
       SD = SQRT(D/(ENZ-1.0))
   18  WRITE (6,19)J,(B(J,I),I=1,3),NZ,XBAR,SD,
      2XG(J),XD(J)
   19  FORMAT (I10,3X,3A4,I7,4F11.4)
       STOP 888
       END
```

Slika 11.19 Program za prethodnu kontrolu podataka

NN	broj posmatranja sa vrednošću nula
X2S	zbir kvadrata vrednosti
B	naziv promenljive
NZ,ENZ	broj posmatranja sa vrednošću različitom od nule
XBAR	srednja vrednost
SD	standardna devijacija
D	međurezultat

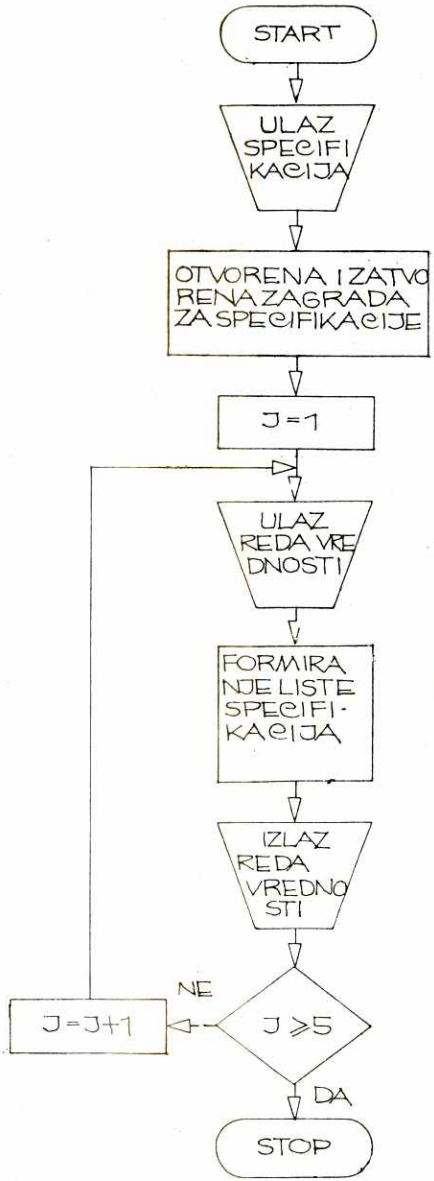
OBELEZJE BR.	NAZIV	RAZLICITO OD NULE	SREDINA	STANDARDNA DEVIJACIJA	MAX	MIN
1	STAROST	20	43.3000	6.0620	53.0000	33.0000
2	VISINA	20	179.7000	7.8481	193.0000	160.0000
3	TEZINA	20	79.0000	8.3792	97.0000	61.0000
4	LOKALNI STAZ	20	6.3500	6.2093	20.0000	1.0000
5	OBIM STRUKA	20	87.6500	9.2069	112.0000	66.0000

Slika 11.20 Rezultati programa sa slike 11.19 (za ulazne podatke sa slike 11.17)

11.6 *BLANKIRANJE POLJÂ KOJA SADRŽE VREDNOST NULA*

Kada se na izlazni medijum upisuju vrednosti promenljivih koje su jednake nuli, po specifikaciji F one imaju oblik 0.00000, a po specifikaciji E ili D oblik 0.00000E 00 odnosno 0.00000D 00, gde će broj decimalnih nula biti jednak vrednosti parametra *d* u specifikacijama. Da bi se u štampanim tabelama bolje uočavale vrednosti nula, pomešane sa drugim vrednostima, potrebno je da se prilikom izlaza polja sa takvim vrednostima ostave prazna, tj. da se ispune znacima blanko. U tu svrhu koristi se program kao što je onaj čiji dijagram vidimo na slici 11.21, a FORTRAN-program na slici 11.22; takve naredbe mogu biti i deo nekog većeg programa ili samostalnog potprograma. Prikazani način je, razume se, moguć samo ako FORTRAN-prevodilac našeg računara dozvoljava promenljive FORMAT-specifikacije (vidi 9.5).

Sve vrednosti za jedan štampani red nalaze se u obliku jednog sloga u datoteci 5. Po naredbi za ulaz one će biti unete na mesta članova matrice VRED (10). Na početku iste datoteke nalazi se jedan slog sa alfanumeričkim podacima, od kojih ćemo u zoni matrice SPEC (12) formirati listu specifikacija za izlaz sloga koji se štampa. Polja alfanumeričkog sloga vezana su sa simbolima u programu na sledeći način:



Slika 11.21 Dijagram za blankiranje polja

Kolone	Sadržaj	Simbol
1— 8	(bbbbbbb	LEVO
9—16	//)bbbbb	DESNO
17—24	A8,bbbbb	A
25—32	F8.2,bbb	F
33—40	bbbbbbbb	BLANKO

```
C       PROGRAM ZA BLANKIRANJE POLJA
C       KOJA SADRZE VREDNOST NULA
        DIMENSION VRED(10)
        DOUBLE PRECISION SPEC(12),
       1LEVO, DESNO, A, F, BLANKO
    1   FORMAT (5A8)
        READ (5,1) LEVO,DESNO,A,F,BLANKO
        SPEC(1) = LEVO
        SPEC(12) = DESNO
        DO 4 J = 1,5
        READ (5,2) VRED
    2   FORMAT (10F8.2)
        DO 3 I = 1,10
        IF(VRED(I).EQ.0.0 ) GO TO 5
        SPEC(I+1) = F
        GO TO 3
    5   SPEC(I+1) = A
        VRED(I) = BLANKO
    3   CONTINUE
    4   WRITE (6,SPEC) VRED
        STOP 999
        END
```

Slika 11.22 Program za blankiranje polja

```
a) 32561      832      1462  - 327      2811      -396
b) 325.61  0  83.20  0.  14.62  -3.27  0.  28.11  0.  -3.96
c) 325.61     83.20      14.62  -3.27      28.11      -3.96
```

Slika 11.23 Ulazni podaci (a) i rezultati (c) za program sa slike 11.22

Naredbom DIMENSION u programu na slici 11.22 rezervisan je prostor za deset članova matrice VRED. Vrednosti koje se tu smeštaju su obični realni brojevi, koji u memoriji računara zauzimaju polja po 4 pozicije*. Naredbom DOUBLE PRECISION deklarišu se sve pomenute promenljive kao realne promenljive dvostruke tačnosti (po 8 pozicija u memoriji*) i istovremeno se rezerviše zona za 12 članova matrice SPEC. Prvom naredbom READ unosi se slog sa sadržajem specifikacija, kao pet alfanumeričkih podataka dužine po osam znakova. Aritmetičkim naredbama SPEC(1) = LEVO i SPEC(12) = DESNO fiksiraju se sadržaji prvog i poslednjeg člana matrice SPEC, koji će uvek biti leva odnosno desna zagrada (vidi primere u odeljku 9.5). Posle toga, po drugoj naredbi READ ulazi svih deset vrednosti članova matrice VRED. Ako se ispitivanjem utvrdi da je vrednost nekog od članova nula, specifikacija za izlaz te vrednosti biće A8, inače F8.2. Sam član matrice VRED, koji je nula, takođe će dobiti alfanumerički sadržaj od osam znakova blanko, umesto realne nule dvostruke tačnosti. Posle te pripreme, ceo niz vrednosti članova matrice VRED biće upisan na spoljni medijum (datoteka 6) i to po listi specifikacija iz matrice SPEC, koja je formirana u toku ispitivanja.

Opisani postupak ponoviće se pet puta (DO 4 J = 1,5) jer je u ovom programu pretpostavljeno da u ulaznoj datoteci 5 ima pet slogova istog sastava. Na slici 11.23 prikazan je pod a) sadržaj i oblik jednog takvog sloga na ulazu u ovaj program, pod b) oblik koji bi slog imao posle izlaza po specifikaciji 10F8.2, a pod c) oblik koji slog dobija po ovom programu.

11.7 *GRAFIČKO PRIKAZIVANJE TOKA FUNKCIJE*

U prethodnim fazama istraživanja neke složene funkcije, često je potrebnije da se ima opšta predstava o njenom toku i o uticaju parametarskih veličina nego da se odmah poznaju njene tačne vrednosti za određene veličine nezavisno promenljive. U takvim okolnostima, grafičko prikazivanje funkcije je najbolje rešenje.

Primer koji će biti prikazan je jedan takav slučaj. Treba da se grafički predstavi promenljivi intenzitet naizmenične struje u serijskom oscilatornom kolu, koje sadrži i element otpornosti, a u kojem je stalna elektromotorna sila zamenjena početnim statičkim opterećenjem serijski vezanog kondenzatora. Pošto nema stalnog izvora

* na pr. računari IBM 360, ICL System 4, CII 10070

energije, oscilacije će biti prigušene, utoliko jače ukoliko je veći aktivni otpor R.

Trenutna vrednost struje u takvom oscilatornom kolu data je izrazom:

$$i = I_m e^{-Rt/2L} \sin 2\pi f_1 t$$

gde je I_m amplituda struje, R aktivni otpor, L induktivnost i f_1 frekvencija oscilovanja kola. Postoje još i sledeći funkcionalni odnosi, koji se koriste pri izračunavanju osnovne funkcije *i*:

$$I_m = \frac{2\pi f_0^2 \cdot Q}{f_1} .$$

$$f_0 = \frac{1}{2\pi} \sqrt{\frac{1}{LC}}$$

$$f_1 = \frac{1}{2\pi} \sqrt{\frac{1}{LC} - \frac{R^2}{4L^2}} \quad \text{gde je } R^2 < 4\frac{L}{C}$$

gde je f_0 frekvencija neprigušenog oscilovanja, Q početno električno opterećenje serijskog kondenzatora, a C njegov kapacitet. Iz poslednjeg izraza se vidi da pri numeričkom radu, potrebnom za crtanje dijagrama, možemo koristiti samo onu kombinaciju parametara R, L i C koja zadovoljava uslov $R^2 < 4L/C$. Inače, kada nema prigušenja ($R = 0$), tada je $f_0 = f_1$ i $I_m = 2\pi f_0 Q$. Sa ovim objašnjenjima je jasno da *i* ima tok jedne periodične funkcije (sa periodom $T = 1/f_1$), čije vrednosti opadaju eksponencijalno kada raste vrednost nezavisno promenljive *t*.

Radi dimenzija hartije u štampaču, apscisna osa dijagrama će biti normalna na štampane redove. Obe ose će biti odštampane kao nizovi tačaka. Tačke na krivoj, za koje se po programu računaju ordinate, biće štampane zvezdicom. Ti znaci . i *, zajedno sa jednim znakom blanko, čitaju se i dodeljuju kao alfanumeričke vrednosti (specifikacija 3A1) promenljivima BLANKO, TACKA i X *(slika 11.24)*.

```
.*
 0.0001      0.01     0.00001      0.002 320
 0.0001      0.06     0.00001      0.002 320
 0.0001      0.18     0.00001      0.002 320
 0.0001      0.25     0.00001      0.002 320
```

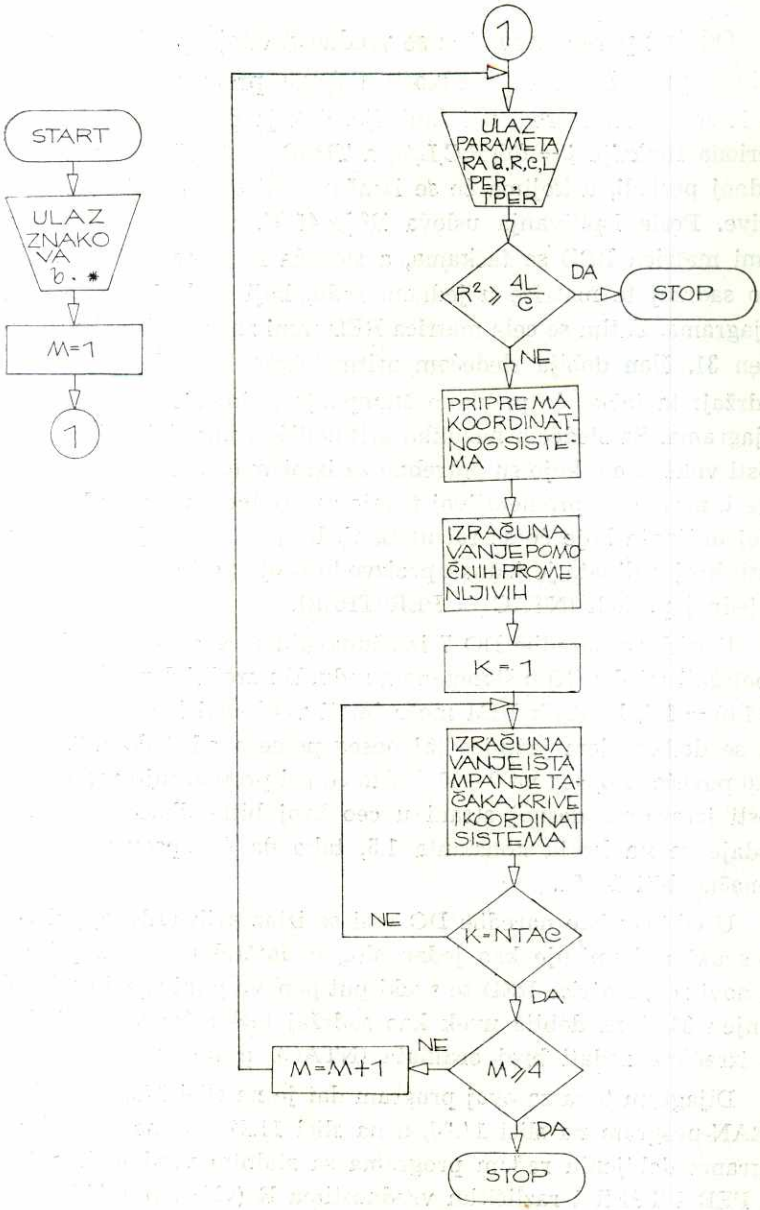
Slika 11.24 Ulazni podaci za program sa slike 11.26

Od šest parametara, čije se vrednosti čitaju po drugoj naredbi READ, prva četiri su vrednosti fizičkih promenljivih, potrebnih za izračunavanje vrednosti funkcije, dok je promenljiva PER broj perioda funkcije $i = f(Q,R,C,L,t)$, a TPER broj tačaka na apscisi u jednoj periodi, u kojima će se izračunavati ordinate i crtati tačke krive. Posle ispitivanja uslova $R^2 \geqslant 4L/C$, ciklus naredbe DO 2 puni matricu RED sa tačkama, a sledeća naredba WRITE štampa ceo sadržaj te matrice u jednom redu, koji će biti ordinatna osa dijagrama. Zatim se cela matrica RED puni znacima blanko, a samo njen 31. član dobija sledećom aritmetičkom naredbom tačku kao sadržaj; ta tačka će prilikom štampanja redova dati apscisnu osu dijagrama. Sa sledećih nekoliko aritmetičkih naredbi daju se vrednosti veličinama koje su potrebne za izračunavanje vrednosti funkcije *i*, nezavisno promenljivoj *t* daje se vrednost nula i određuje se broj ordinata koje će se računati, tj. broj tačaka koje će se štampati; broj ordinata je jednak proizvodu broja perioda i broja tačaka u jednoj periodi (NTAC = PER*TPER).

U ciklusu naredbe DO 7 izračunavaju se vrednosti funkcije (I) i položaji tačaka (X) u štampanom redu. U naredbi J = 30.0*(I/IM + + 1.0) + 1.5, količnik I/IM može imati vrednosti između —1 i +1, pa se dodavanjem jedinice taj opseg pomera od 0 do +2. Faktor 30.0 povećava opseg na 0—60. Pošto će pri pretvaranju realne vrednosti izraza na desnoj strani u ceo broj biti odbačene decimale, dodaje se na kraju konstanta 1.5, tako da će opseg vrednosti J konačno biti 1—61.

U ciklusu iste naredbe DO vrši se izlaz svih redova, pri čemu se svaki red upisuje kao jedan slog u datoteku 6. Kao priprema za novi red, matrica RED se svaki put ponovo puni znacima blanko, a njen 31. član dobija uvek kao sadržaj tačku (za apscisu). Kada se izračuna zadati broj ordinata (NTAC), program se zaustavlja.

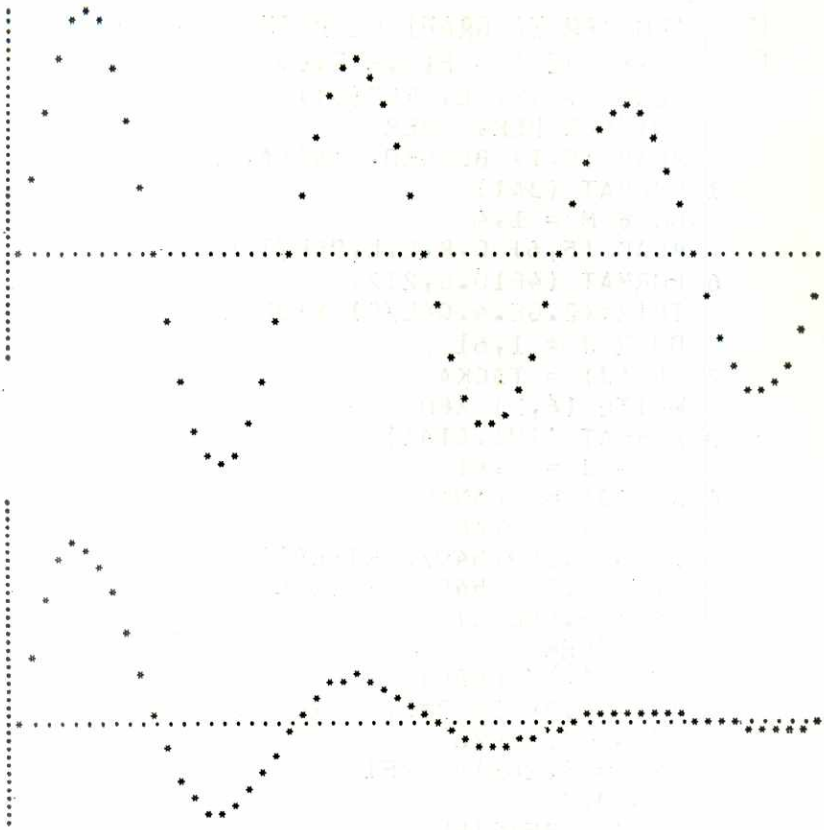
Dijagram toka za ovaj program dat je na slici *11.25*, sam FORTRAN-program na slici *11.26*, a na slici *11.27* vidimo nekoliko dijagrama dobijenih radom programa sa stalnim vrednostima Q, C, L, PER i TPER i različitim vrednostima R (vidi sliku 11.24).



Slika 11.25 Dijagram za grafičko prikazivanje

```
C     PROGRAM ZA GRAFICKO PRIKAZIVANJE
C     FUNKCIJE I = F(Q,R,C,L,T)
      REAL I, IM, L, RED(61)
      INTEGER PER, TPER
      READ (5,1) BLANKO, TACKA, X
    1 FORMAT (3A1)
      DO 8 M = 1,4
      READ (5,6) Q,R,C,L,PER,TPER
    6 FORMAT (4F10.0,2I2)
      IF(R**2.GE.4.0*L/C) STOP 111
      DO 2 J = 1,61
    2 RED(J) = TACKA
      WRITE (6,3) RED
    3 FORMAT (1H1,61A1)
      DO 4 J = 1,61
    4 RED(J) = BLANKO
      RED(31) = TACKA
      F0 = 0.1591549/SQRT(L*C)
      F1 = 0.1591549*SQRT(1.0/(L*C) -
     2R*R/(4.0*L*L))
      P = TPER
      DT = 1.0/(P*F0)
      IM = 6.2831853*F0*F0*Q/F1
      VK = -0.5*R/L
      OM1 = 6.2831853*F1
      T = 0.0
      NTAC = PER*TPER
      DO 7 K = 1,NTAC
      I = IM*EXP(VK*T)*SIN(OM1*T)
      J = 30.0*(I/IM+1.0) + 1.5
      RED(J) = X
      WRITE (6,5) RED
    5 FORMAT (1H ,61A1)
      RED(J) = BLANKO
      RED (31) = TACKA
    7 T = T + DT
    8 CONTINUE
      STOP 555
      END
```

Slika 11.26 Program za grafičko prikazivanje funkcije

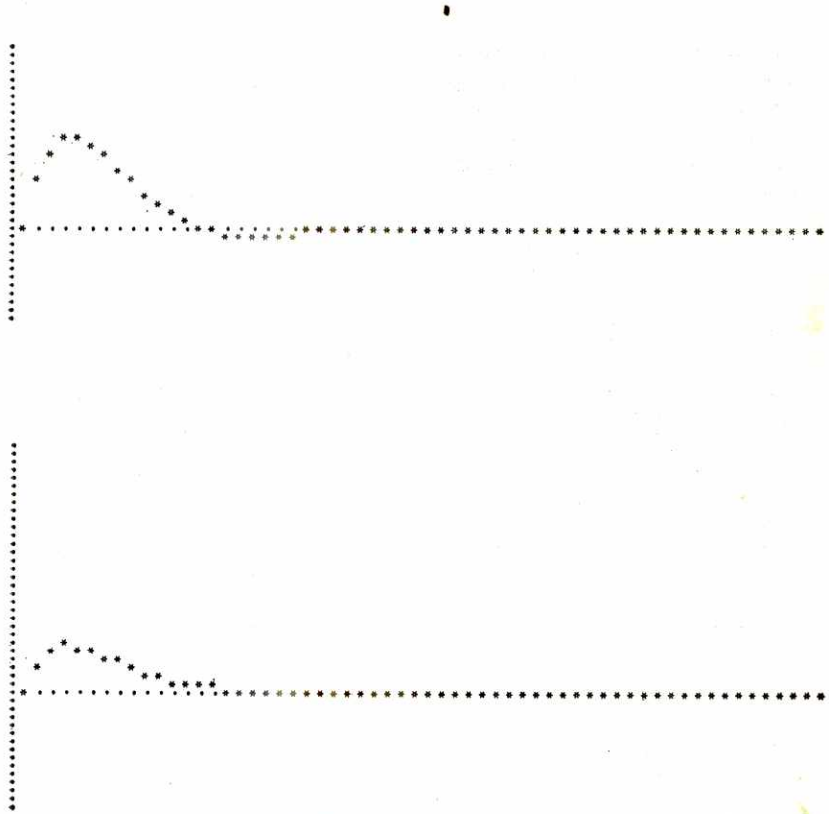


Slika 11.27 Rezultati programa sa slike 11.26 (1. deo)

11.8 *SISTEM LINEARNIH ALGEBARSKIH JEDNAČINA*

FORTRAN-programa za rešavanje sistema linearnih algebarskih jednačina ima više, pošto je takav matematički aparat veoma potreban u problemima iz različitih oblasti i često se upotrebljava u svim računskim centrima. Ovde će biti izložen jedan od tih programa*, koji je od posebnog interesa zato što ilustruje rešenje sa većim brojem potprograma. Što se tiče primenjenog matematičkog

* Rešenje je uzeto iz priručnika IBM System/360 Scientific Subroutine Package, H20-0205-3, str. 449—451



Slika 11.27 Rezultati programa sa slike 11.26 (2. deo)

postupka, rešenje sistema se dobija metodom eliminacije. Program štampa i ulazne podatke i rešenja i ponavlja se automatski sve dok ima podataka u ulaznoj datoteci.

Ulazni podaci se sastoje od četiri vrste slogova (vidi sliku *11.28*). Na čelu grupe nalazi se jedan slog sa sledećim podacima: identifikacija zadatka, broj redova matrice koeficijenata, broj kolona matrice koeficijenata, kôd oblika matrice. Ovoliki broj parametara nije bezuslovno potreban za rešavanje sistema linearnih jednačina (broj redova i kolona je uvek jednak, članovi matrice su u opštem slučaju različiti); međutim, dva upotrebljena potprograma (MATIN i MXOUT) su opšti parametarski programi, koji se u biblioteci programâ ne nalaze samo zbog rešavanja sistema linearnih jedna-

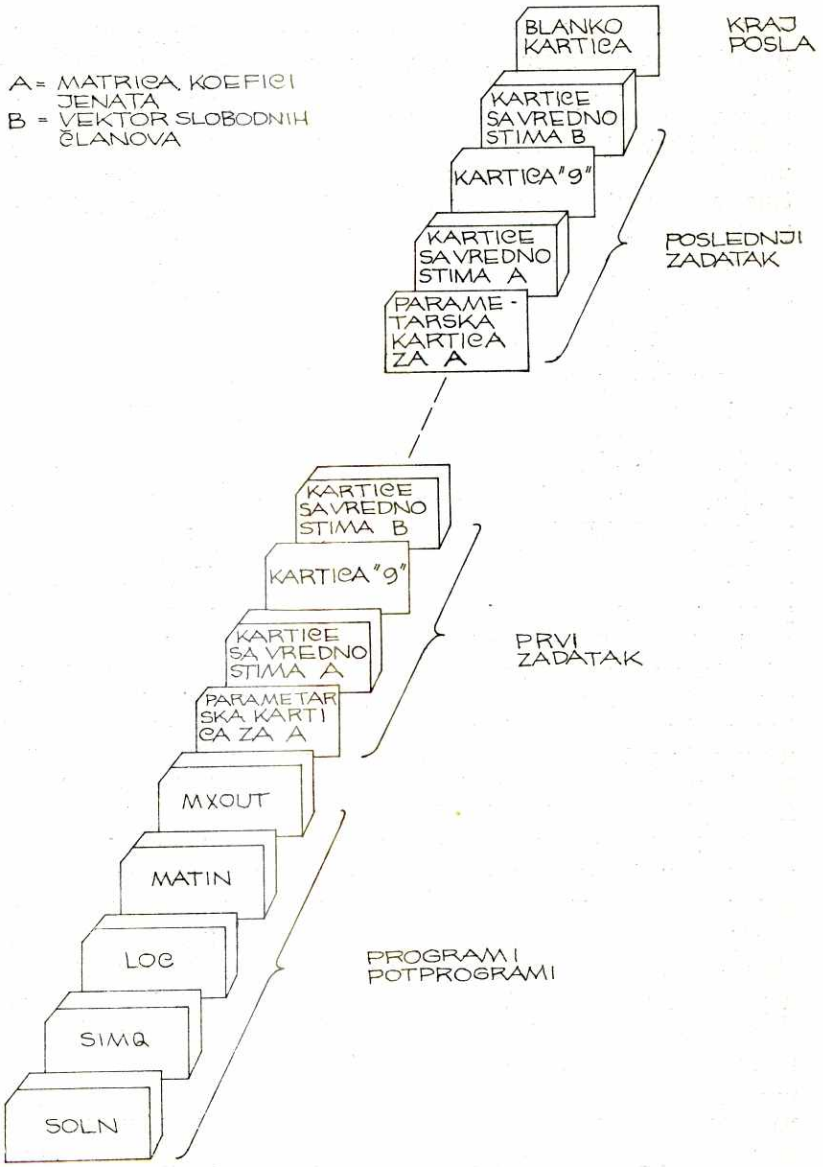
čina nego služe kao potprogrami za ulaz odnosno izlaz matrica bilo kakvog oblika i veličine (na pr. u programima za operacije matrične algebre i sl.). Zato se u ovom konkretnom slučaju moraju u prvom slogu navesti jednake vrednosti za broj redova i kolona, a polje za kôd vrste matrice treba ostaviti prazno (vrednost nula), što znači da se radi o opštem obliku matrice. (Kôd 1 znači da su na ulazu dati samo članovi gornjeg trougla i članovi na dijagonali (simetrična matrica), dok kôd 2 znači da se na ulazu nalazi samo vektor dijagonalnih članova). Specifikacija za ovaj slog je data u prvoj naredbi READ potprograma MATIN, tj. u pripadajućoj naredbi FORMAT.

```
    1  10                                                                    
1.0000000 0.6644085 0.7601008 0.7507505 0.4299425 0.0033291 0.7284786
0.6751766 0.8635910 0.7446845
0.6644085 1.0000000 0.6271802 0.6194650 0.3547574 0.0027470 0.6010878
0.5571068 0.7125728 0.6144597
0.7601008 0.6271802 1.0000000 0.7086843 0.4058519 0.0031426 0.6876602
0.6373449 0.8152021 0.7029582
0.7507505 0.6194650 0.7086843 1.0000000 0.4008593 0.0031039 0.6792011
0.6295047 0.8051740 0.6943108
0.4299425 0.3547574 0.4058519 0.4008593 1.0000000 0.0017776 0.3889673
0.3605070 0.4611099 0.3976204
0.0033291 0.0027470 0.0031426 0.0031039 0.0017776 1.0000000 0.0030119
0.0027915 0.0035705 0.0030789
0.7284786 0.6010878 0.6876602 0.6792011 0.3889673 0.0030119 1.0000000
0.6108296 0.7812874 0.6737132
0.6751766 0.5571068 0.6373449 0.6295047 0.3605070 0.0027915 0.6108296
1.0000000 0.7241215 0.6244183
0.8635910 0.7125728 0.8152021 0.8051740 0.4611099 0.0035705 0.7812874
0.7241215 1.0000000 0.7986682
0.7446845 0.6144597 0.7029582 0.6943108 0.3976204 0.0030789 0.6737132
0.6244183 0.7986682 1.0000000
9
     110.0     -120.0       10.0      145.0      -50.0       44.2      -14.0
      38.5       22.0     1650.0
```

Slika 11.28 Ulazni podaci za program sa slike 11.31

Koeficijenti matrice moraju za ulaz biti pripremljeni po sedam u jednom slogu, pri čemu svaki koeficijent zauzima polje od deset kolona. Na 80-kolonskim karticama to znači da se koristi prvih 70 kolona, dok poslednjih 10 mogu biti iskorišćene za identifikaciju zadatka i tsl. Specifikacija uz naredbu za ulaz koeficijenata (to je naredba READ (5,1) u potprogramu MATIN) jeste 7F10.0, što znači da u poljima kartice treba bušiti i decimalnu tačku na potrebnom mestu. Isti oblik sloga važi i za ulaz vektora slobodnih članova (naredba READ (5,20) u programu SOLN).

Koeficijenti se u kartice buše *po redovima* matrice, tako da se u grupi uzastopnih kartica nalaze svi koeficijenti jedne jednačine;



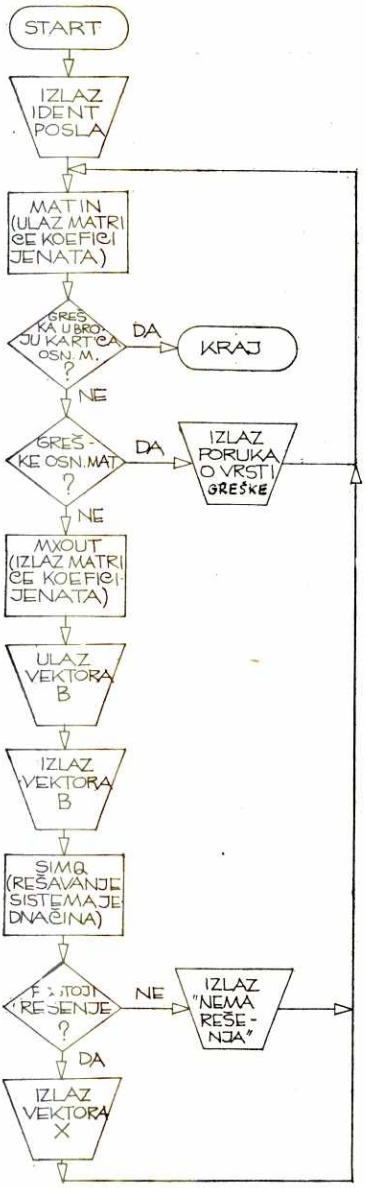
Slika 11.29 Sastav paketa kartica za rešavanje više sistema jednačina jednog za drugim

novi red matrice (nova jednačina) mora započeti sa prvim poljem prve kartice nove grupe. Isto važi za slobodne članove jednačina, čiji slogovi se u ulaznoj datoteci nalaze neposredno iza sloga sa cifrom 9 u prvoj koloni (vrednost $9.0 \cdot 10^9$), kojom se završavaju slogovi koeficijenata. Na slici *11.28* vidimo jednu potpunu grupu ulaznih slogova, za jedan konkretan sistem od 10 linearnih jednačina, a na slici 11.29 sastav jednog niza takvih grupa, zajedno sa programima koji će ih rešavati jednu za drugom.

Skup programskih jedinica, koje serijski rešavaju sisteme linearnih jednačina, sastoji se od programa SOLN i četiri potprograma SUBROUTINE: MATIN, SIMQ, MXOUT i LOC. Sve jedinice su napisane tako da se mogu neposredno upotrebiti za rešavanje sistema od 2 do maksimalno 50 jednačina, ali se mogu lako modifikovati i za rešavanje većih sistema; o tome će biti reči kasnije.

Program SOLN, čiji dijagram vidimo na slici *11.30*, a izvorni oblik na slici *11.31*, jeste ona programska jedinica sa čijim izvršavanjem počinje ceo proces rada. Odmah na početku, on poziva potprogram MATIN (slike *11.33* i *11.34*) radi ulaza matrice koeficijenata A_{ij} iz ulazne datoteke. Potprogram MATIN, posle čitanja prvog sloga iz ulazne datoteke, poziva potprogram LOC (slika *11.35* i *11.36*) i to u dva razna slučaja: po prvoj naredbi CALL potprogram LOC daje podatke o potrebnom prostoru za smeštaj matrice koeficijenata, na osnovu čega potprogram MATIN konstatuje da li je zona A dovoljna za sve članove A_{ij}. Ako nije, MATIN indicira grešku, a ako jeste, počinje ulaz članova matrice iz ulazne datoteke u zonu A; pri tome, potprogram LOC, pozvan drugom naredbom CALL, uspostavlja za svaki član korespondenciju između njegovih indeksa i »unutrašnjeg« indeksa.

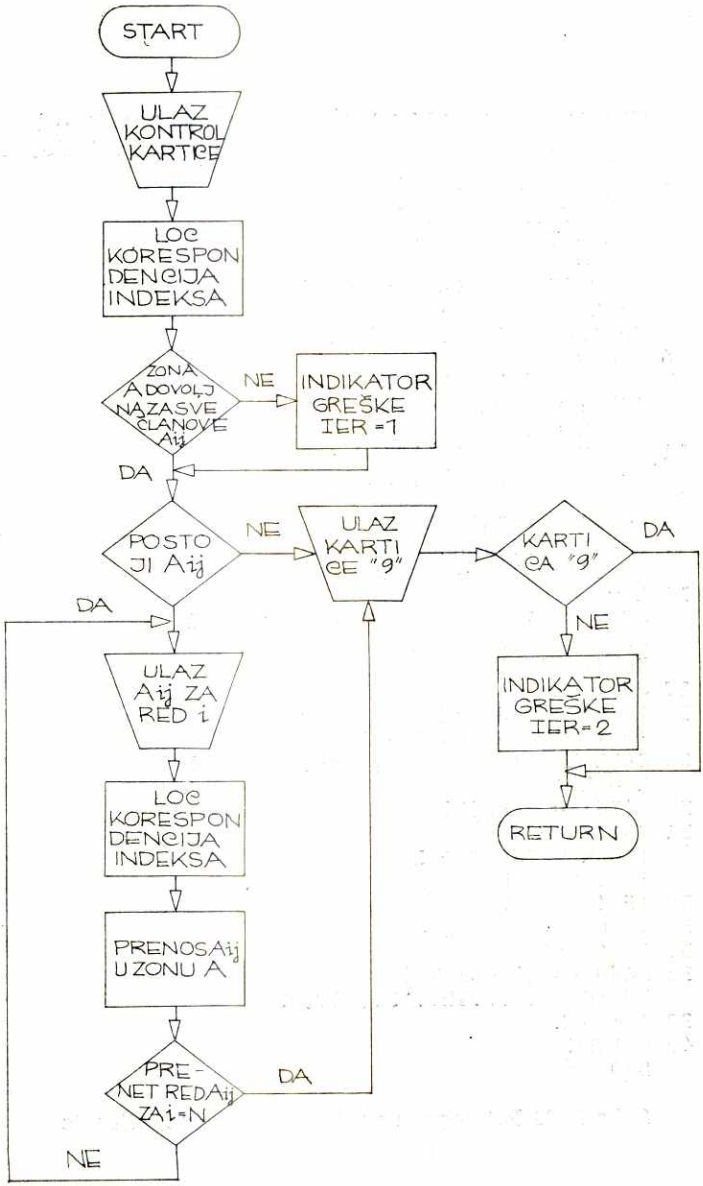
Posle ulaza svih članova matrice A_{ij}, nastavlja se rad programa SOLN, koji odmah poziva potprogram MXOUT (slike *11.37* i *11.38*) radi izlaza svih članova. Potprogram MXOUT najpre štampa zaglavlje, pa zatim poziva potprogram LOC, koji obavlja funkciju obrnutu od one prilikom pozivanja drugom naredbom CALL LOC u potprogramu MATIN. Potprogram MXOUT, zajedno sa potprogramom LOC, prenosi na izlazni medijum sve članove matrice A_{ij}, u rasporedu koji je prikazan na slici *11.32a*. Po završetku prenosa, program SOLN čita sa ulaznog medijuma slobodne članove (vektor B) i iste prenosi na izlazni medijum, u rasporedu koji vidimo na slici *11.32b*. Posle toga, on poziva potprogram SIMQ (slike *11.39* i *11.40*).



Slika 11.30 Dijagram za program sa slike 11.31

```
      PROGRAM ZA RESAVANJE SISTEMA LINEARNIH JEDNACINA - SOLN
      DIMENSION A(2500),B(50)
      COMMON A, B
      WRITE (6,10)
   10 FORMAT (1H1,25HRESENJE SISTEMA JEDNACINA)
   25 CALL MATIN(ICOD,A,2500,N,M,MS,IER)
      IF (N) 30, 95, 30
   30 IF (IER-1) 45, 35, 40
   35 WRITE (6,11) ICOD
   11 FORMAT (1HO,34HZONA NEDOVOLJNA ZA ULAZNU MATRICU ,I4)
      GO TO 90
   40 WRITE (6,14) ICOD
   14 FORMAT (1HO,32HNETACAN BROJ KARTICA ZA MATRICU ,I4)
      GO TO 95
   45 IF (N-M) 50, 55, 50
   50 WRITE (6,13) ICOD
   13 FORMAT (1HO,47HNE SLAZU SE DIMENZIJE REDA I KOLONE ZA MATRICU ,I4)
      GO TO 90
   55 IF (MS) 60, 65, 60
   60 WRITE (6,16) ICOD
   16 FORMAT (1HO,35HKOD STRUKTURE NIJE NULA ZA MATRICU ,I4)
      GO TO 90
   65 CALL MXOUT (ICOD,A,N,M,MS,60,120,2)
      READ (5,20) (B(I),I=1,N)
   20 FORMAT (7F10.0)
      WRITE (6,17)
   17 FORMAT (1H1,19HORIGINALNI VEKTOR B,////)
      DO 70 I = 1,N
   70 WRITE (6,21) I, B(I)
   21 FORMAT (I3,10X,E16.6)
      CALL SIMQ (A,B,N,KS)
      IF (KS-1) 80, 75, 80
   75 WRITE (6,19)
   19 FORMAT (1HO,21HMATRICA JE SINGULARNA)
      WRITE (6,15)
   15 FORMAT (1HO,37HRAD SE NASTAVLJA SA SLEDECIM ZADATKOM)
      GO TO 25
   80 WRITE (6,18)
   18 FORMAT (1H1,13HR E S E N J A,////)
      DO 85 I = 1,N
   85 WRITE (6,21) I, B(I)
      WRITE (6,22)
   22 FORMAT (1HO,12HKRAJ ZADATKA)
      GO TO 25
   90 READ (5,20) (B(I),I=1,N)
      WRITE (6,15)
      GO TO 25
   95 WRITE (6,12)
   12 FORMAT (1HO,20HIZVRSAVANJE ZAVRSENO)
      RETURN
      END
```

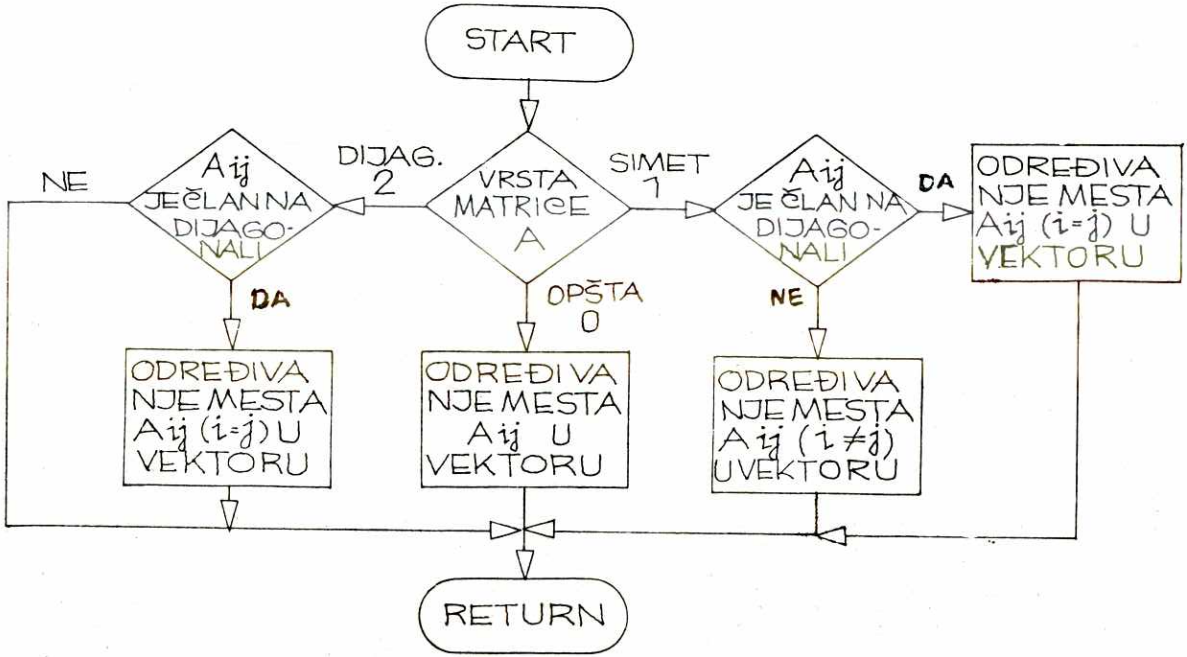
Slika 11.31 Program za rješavanje sistema linearnih jednačina (sa četiri potprograma)



Slika 11.33 Dijagram za potprogram sa slike 11.34

```
C     POTPROGRAM ZA ULAZ MATRICE KOEFICIJENATA - MATIN
      SUBROUTINE MATIN (ICOD,A,ISIZE,IROW,ICOL,IS,IER)
      DIMENSION A(2500), CARD(8)
    1 FORMAT (7F10.0)
    2 FORMAT (I6,2I4,I2)
      IDC = 7
      IER = 0
      READ (5,2) ICOD,IROW,ICOL,IS
      CALL LOC(IROW,ICOL,ICNT,IROW,ICOL,IS)
      IF (ISIZE-ICNT) 6, 7, 7
    6 IER = 1
    7 IF (ICNT) 38, 38, 8
    8 ICOLT = ICOL
      IRDCR = 1
   11 IRCDS = (ICOLT-1)/IDC + 1
      IF (IS-1) 15, 15, 12
   12 IRCDS = 1
   15 DO 31 K = 1,IRCDS
      READ (5,1) (CARD(I), I=1,IDC)
      IF (IER) 16, 16, 31
   16 L = 0
      JS = (K-1)*IDC + ICOL - ICOLT + 1
      JE = JS + IDC - 1
      IF (IS-1) 19, 19, 17
   17 JE = JS
   19 DO 30 J = JS,JE
      IF (J-ICOL) 20, 20, 31
   20 CALL LOC(IRDCR,J,IJ,IROW,ICOL,IS)
      L = L + 1
   30 A(IJ) = CARD(L)
   31 CONTINUE
      IRDCR = IRDCR + 1
      IF (IROW-IRDCR) 38, 35, 35
   35 IF (IS-1) 37, 36, 36
   36 ICOLT = ICOLT - 1
   37 GO TO 11
   38 READ (5,1) CARD(1)
      IF (CARD(1)-9.E9) 39, 40, 39
   39 IER = 2
   40 RETURN
      END
```

Slika 11.34 Potprogram za ulaz matrice koeficijenata



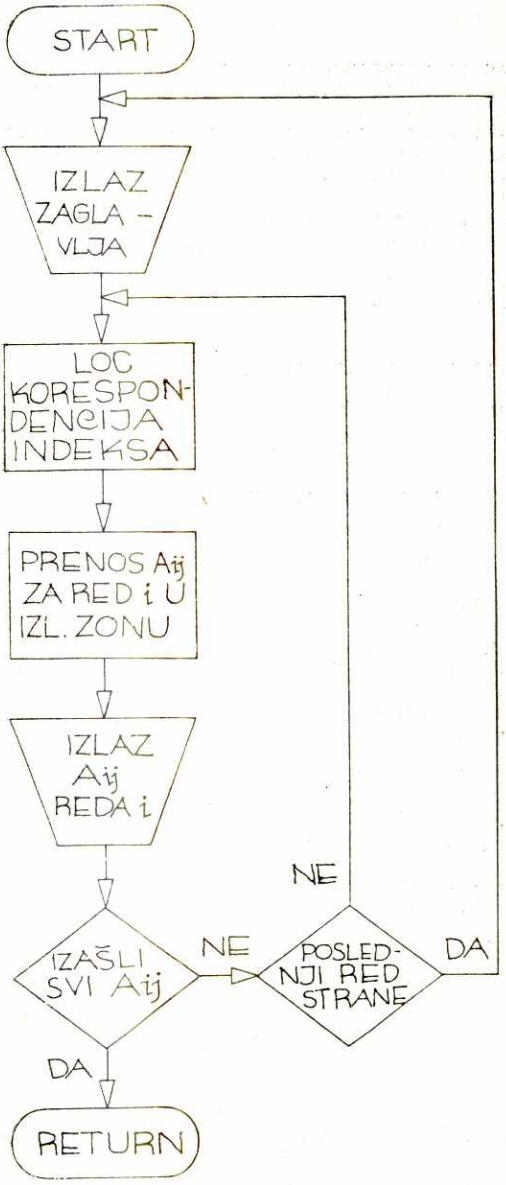
Slika 11.35 ***Dijagram*** *za potprogram sa slike 11.36*

```
C     POTPROGRAM ZA RACUNANJE INDEKSA MESTA - LOC
      SUBROUTINE LOC(I,J,IR,N,M,MS)
      IX = I
      JX = J
      IF (MS-1) 10, 20, 30
   10 IRX = N*(JX-1) + IX
      GO TO 36
   20 IF (IX-JX) 22, 24, 24
   22 IRX = IX + (JX*JX-JX)/2
      GO TO 36
   24 IRX = JX + (IX*IX-IX)/2
      GO TO 36
   30 IRX = 0
      IF (IX-JX) 36, 32, 36
   32 IRX = IX
   36 IR = IRX
      RETURN
      END
```

Slika 11.36 Potprogram za »unutrašnje« indekse

Potprogram SIMQ ustvari nalazi rešenja sistema jednačina, po poznatoj Gausovoj metodi eliminacije. Osim toga, radi poboljšanja tačnosti računanja, postupak je dopunjen deljenjem najvećim koeficijentom po kolonama. Počinje s tim što u datom sistemu jednačina traži član najveće apsolutne vrednosti u prvoj koloni i upoređuje ga sa veličinom ε, koja je u ovom slučaju nula (može se i promeniti, pre prevođenja potprograma). Ako je taj član jednak ili manji od ε, daje se indikacija da je matrica koeficijenata singularna. Zavisno od mesta u koloni koeficijenata, na kojem je nađen koeficijent sa najvećom apsolutnom vrednošću, vrši se promena mesta svih koeficijenata tog reda sa koeficijentima prvog reda. Zatim sledi prvi korak eliminacionog postupka. Isti proces se ponavlja u sledećem koraku nad formiranim sistemom od N-1 jednačina, sa isto toliko nepoznatih (jer je u prvom koraku eliminisana jedna nepoznata), itd. Kada su na taj način eliminisane sve nepoznate, računaju se unatrag vrednosti rešenja i smeštaju u zonu jednodimenzionalne matrice B, gde su bile vrednosti slobodnih članova jednačine.

Po završenom radu potprograma SIMQ, program SOLN na osnovu indikacije iz potprograma SIMQ utvrđuje da li postoji rešenje sistema. Ako postoji, program SOLN prenosi na izlazni me-



Slika 11.37 Dijagram za potprogram sa slike 11.38

```
C       POTPROGRAM ZA STAMPANJE MATRICA - MXOUT
        SUBROUTINE MXOUT (ICOD,A,N,M,MS,LINS,IPOS,ISP)
        DIMENSION A(2500), B(8)
    1 FORMAT (1H1,4X,7HMATRICA,I5,4X,I3,7H REDOVA,7X,I3,7H KOLONA,
       115X,16HNACIN SMESTANJA ,I1,6X,7HSTRANA ,I2/)
    2 FORMAT (12X,8HKOLONA  ,7(3X,I3,10X))
    3 FORMAT (1H )
    4 FORMAT (1H ,7X,4HRED ,I3,7(E16.6))
    5 FORMAT (1HO,7X,4HRED ,I3,7(E16.6))
        J = 1
        NEND = IPOS/16 - 1
        LEND = (LINS/ISP) - 2
        IPAGE = 1
   10 LSTRT = 1
   20 WRITE (6,1) ICOD,N,M,MS,IPAGE
        JNT = J + NEND - 1
        IPAGE = IPAGE + 1
   31 IF (JNT-M) 33, 33, 32
   32 JNT = M
   33 CONTINUE
        WRITE (6,2) (JCUR,JCUR=J,JNT)
        IF (ISP-1) 35, 35, 40
   35 WRITE (6,3)
   40 LTEND = LSTRT + LEND - 1
        DO 80 L = LSTRT,LTEND
        DO 55 K = 1,NEND
        KK = K
        JT = J + K - 1
        CALL LOC(L,JT,IJNT,N,M,MS)
        B(K) = 0.0
        IF (IJNT) 50, 50, 45
   45 B(K) = A(IJNT)
   50 CONTINUE
        IF (JT-M) 55, 60, 60
   55 CONTINUE
   60 IF (ISP-1) 65, 65, 70
   65 WRITE (6,4) L, (B(JW),JW=1,KK)
        GO TO 75
   70 WRITE (6,5) L, (B(JW),JW=1,KK)
   75 IF (N-L) 85, 85, 80
   80 CONTINUE
        LSTRT = LSTRT + LEND
        GO TO 20
   85 IF (JT-M) 90, 95, 95
   90 J = JT + 1
        GO TO 10
   95 RETURN
        END
```

Slika 11.38 Potprogram za izlaz matrice koeficijenata

MATRICA 1 10 REDOVA 10 KOLONA NACIN SMESTANJA 0 STRANA 1

	KOLONA	1	2	3	4	5	6
RED	1	0.100000E 01	0.664408E 00	0.760101E 00	0.750750E 00	0.429942E 00	0.332910E-02
RED	2	0.664408E 00	0.100000E 01	0.627180E 00	0.619465E 00	0.354757E 00	0.274700E-02
RED	3	0.760101E 00	0.627180E 00	0.100000E 01	0.708684E 00	0.405852E 00	0.314260E-02
RED	4	0.750750E 00	0.619465E 00	0.708684E 00	0.100000E 01	0.400859E 00	0.310390E-02
RED	5	0.429942E 00	0.354757E 00	0.405852E 00	0.400859E 00	0.100000E 01	0.177760E-02
RED	6	0.332910E-02	0.274700E-02	0.314260E-02	0.310390E-02	0.177760E-02	0.100000E 01
RED	7	0.728479E 00	0.601088E 00	0.687660E 00	0.679201E 00	0.388967E 00	0.301190E-02
RED	8	0.675177E 00	0.557107E 00	0.637345E 00	0.629505E 00	0.360507E 00	0.279150E-02
RED	9	0.863591E 00	0.712573E 00	0.815202E 00	0.805174E 00	0.461110E 00	0.357050E-02
RED	10	0.744684E 00	0.614460E 00	0.702958E 00	0.694311E 00	0.397620E 00	0.307890E-02

MATRICA 1 10 REDOVA 10 KOLONA NACIN SMESTANJA 0 STRANA 2

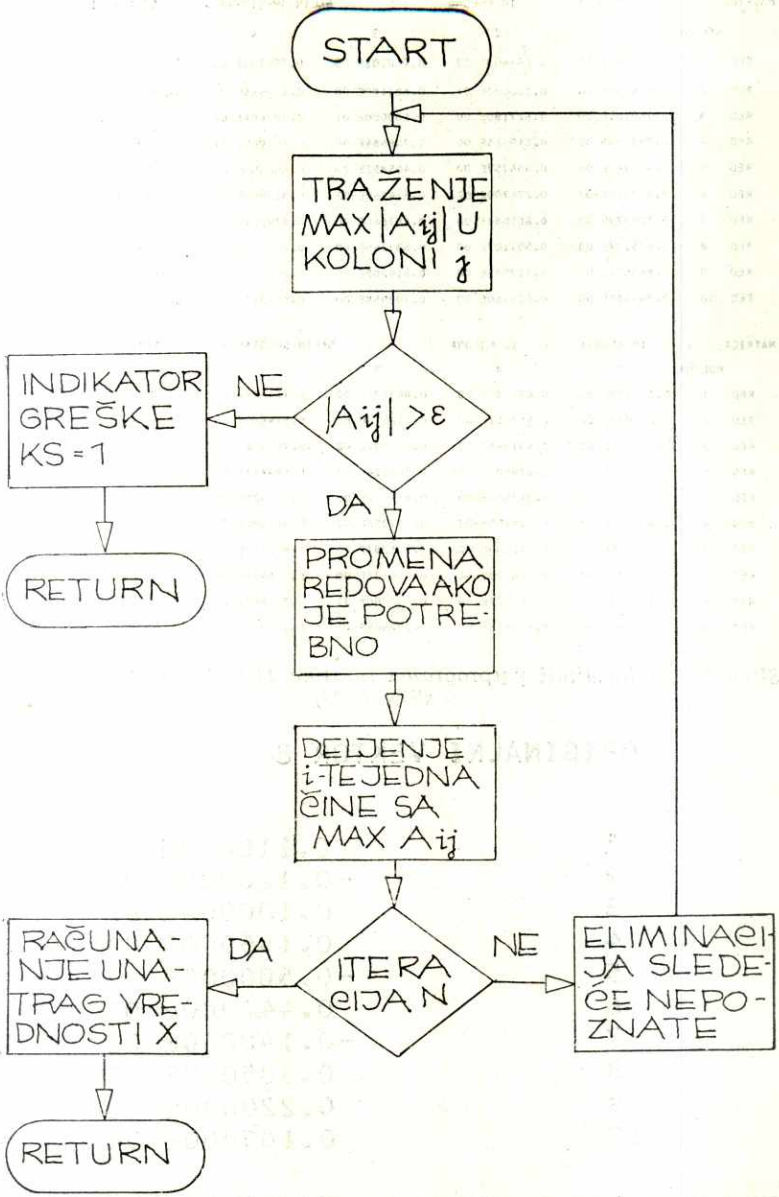
	KOLONA	7	8	9	10
RED	1	0.728479E 00	0.675177E 00	0.863591E 00	0.744684E 00
RED	2	0.601088E 00	0.557107E 00	0.712573E 00	0.614460E 00
RED	3	0.687660E 00	0.637345E 00	0.815202E 00	0.702958E 00
RED	4	0.679201E 00	0.629505E 00	0.805174E 00	0.694311E 00
RED	5	0.388967E 00	0.360507E 00	0.461110E 00	0.397620E 00
RED	6	0.301190E-02	0.279150E-02	0.357050E-02	0.307890E-02
RED	7	0.100000E 01	0.610830E 00	0.781287E 00	0.6 3713E 00
RED	8	0.610830E 00	0.100000E 01	0.724121E 00	0.6 4418E 00
RED	9	0.781287E 00	0.724121E 00	0.100000E 01	0.7 8668E 00
RED	0	0.673713E 00	0.624418E 00	0.798668E 00	0.1 0000E 01

Slika 11.32a Rezultati potprograma sa slike 11.38 (za ulazne podatke sa slike 11.28)

ORIGINALNI VEKTOR B

1	0.110000E 03
2	-0.120000E 03
3	0.100000E 02
4	0.145000E 03
5	-0.500000E 02
6	0.442000E 02
7	-0.140000E 02
8	0.385000E 02
9	0.220000E 02
10	0.165000E 04

Slika 11.32b Rezultati programa sa slike 11.31 (za ulazne podatke sa slike 11.28)



Slika 11.39 Dijagram za potprogram sa slike 11.40

```
C   -  POTROGRAM ZA RESENJA (VEKTOR X) - SIMQ
      SUBROUTINE SIMQ (A,B,N,KS)
      DIMENSION A(2500), B(50)
      TOL = 0.0
      KS = 0
      JJ = -N
      DO 65 J = 1,N
      JY = J + 1
      JJ = JJ + N + 1
      BIGA = 0.0
      IT = JJ - J
      DO 30 I = J,N
      IJ = IT + I
      IF (ABS(BIGA)-ABS(A(IJ))) 20, 30, 30
   20 BIGA = A(IJ)
      IMAX = I
   30 CONTINUE
      IF (ABS(BIGA)-TOL) 35,35, 40
   35 KS = 1
      RETURN
   40 I1 = J + N*(J-2)
      IT = IMAX - J
      DO 50 K = J,N
      I1 =I1 + N
      I2 = I1 + IT
      SAVE = A(I1)
      A(I1) = A(I2)
      A(I2) = SAVE
   50 A(I1) = A(I1)/BIGA
      SAVE = B(IMAX)
      B(IMAX) = B(J)
      B(J) = SAVE/BIGA
      IF (J-N) 55, 70, 55
   55 IQS = N*(J-1)
      DO 65 IX = JY,N
      IXJ = IQS + IX
      IT = J - IX
      DO 60 JX = JY,N
      IXJX = N*(JX-1) + IX
      JJX = IXJX + IT
   60 A(IXJX) = A(IXJX) - (A(IXJ)*A(JJX))
   65 B(IX) = B(IX) - (B(J)*A(IXJ))
   70 NY = N - 1
      IT = N*N
      DO 80 J = 1,NY
      IA = IT - J
      IB = N - J
      IC = N
      DO 80 K = 1,J
      B(IB) = B(IB) - A(IA)*B(IC)
      IA = IA - N
   80 IC = IC - 1
      RETURN
      END
```

Slika 11.40 Potprogram za nalaženje rešenja sistema

dijum vrednosti rešenja, u rasporedu koji vidimo na slici 11.32c. Ako ne postoji, izlaze poruke »matrica je singularna« i »rad se nastavlja sa sledećim zadatkom«. U oba slučaja, naredbom 25 CALL MATIN traži se ulaz nove matrice koeficijenata (sledeći zadatak).

Naredba COMMON A,B u programu SOLN nepotrebna je jer nema odgovarajućih naredbi u potprogramima. Zone za matrice A i B rezervisane su samo u programu SOLN, dok u potprogramima prevodilac definiše samo jedno polje, u koje će biti smešten jedan po jedan član kada se, pozivanjem potprograma, simbol matrice prenese kao stvarni argument na mesto fiktivnog argumenta u definicionoj naredbi potprograma. Definisanje zajedničkih zona pomoću naredbi COMMON bilo bi potrebno kada simboli matrica ne bi bili korišćeni kao argumenti u definicionim naredbama i naredbama za pozivanje.

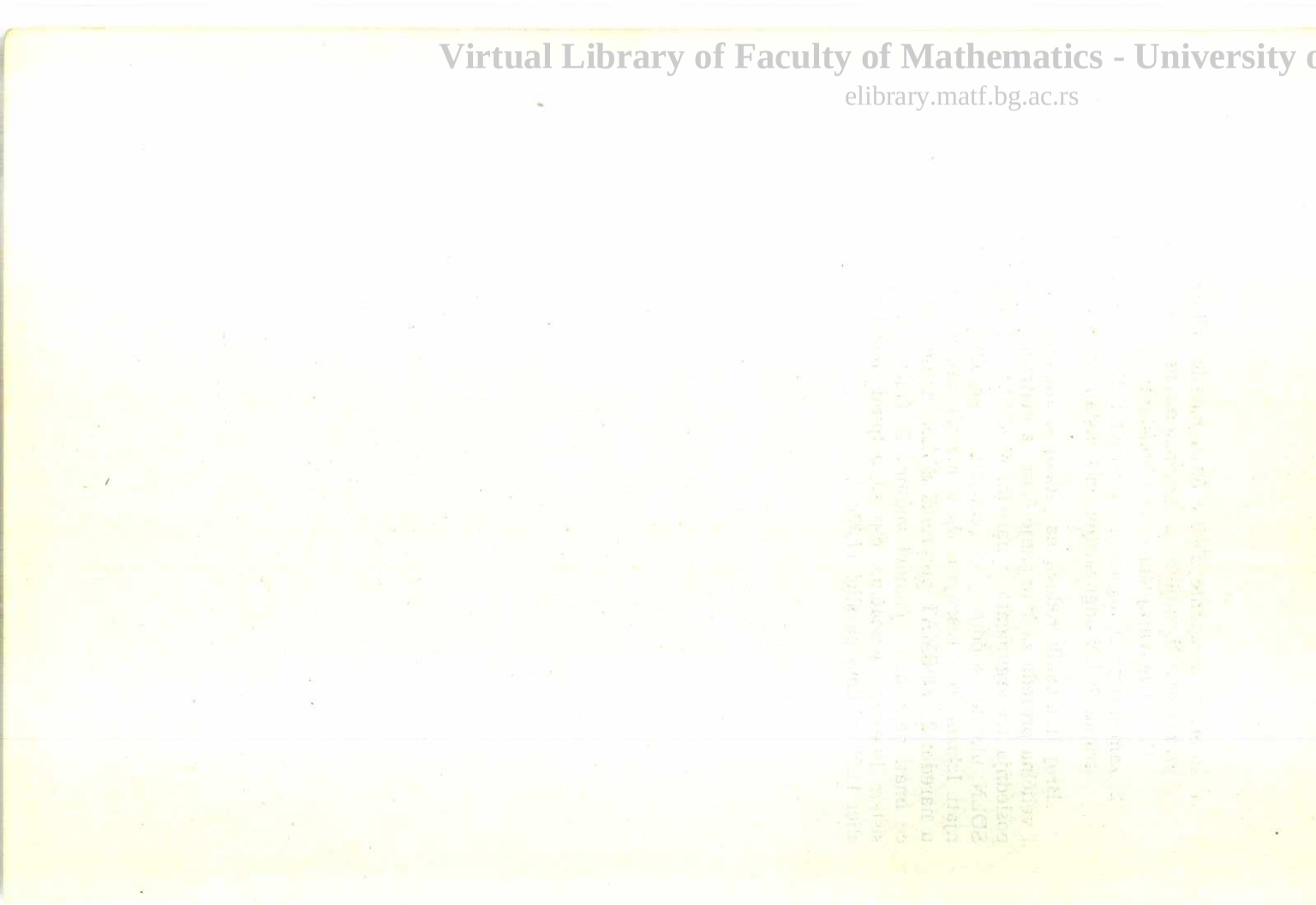
Ako se želi da se sa ovom grupom programskih jedinica rešavaju sistemi jednačina veći od 50, tada se moraju izvršiti sledeće modifikacije u izvornim naredbama:

```
R E S E N J A


 1          -0.283124E 03
 2          -0.567240E 03
 3          -0.516456E 03
 4          -0.299155E 02
 5          -0.179352E 03
 6           0.435176E 02
 7          -0.479274E 03
 8          -0.230430E 03
 9          -0.210172E 04
10           0.480974E 04

KRAJ ZADATKA
```

Slika 11.32c Rezultati potprograma sa slike 11.40 (za ulazne podatke sa slike 11.28)

1. zameniti konstante 2500 i 50 u naredbi DIMENSION programa SOLN, kojom su dimenzionisane zone za matrice A i B, sa odgovarajućim većim konstantama,
2. zameniti treći argument u naredbi 25 CALL MATIN programa SOLN odgovarajućom konstantom (kao pod 1.).

Broj štampanih redova na jednoj strani, broj pozicija u redu i veličina proreda za štampanje članova matrice A regulisani su sa poslednja tri argumenta u naredbi 65 CALL MXOUT programa SOLN; ukoliko je potrebno, vrednosti tih argumenata se mogu menjati. Izlazni oblik rešenja može se menjati promenom specifikacija u naredbi 21 FORMAT programa SOLN; u tom slučaju, isti oblik će imati na izlazu i članovi vektora B. Odštampane rezultate za sistem linearnih jednačina, čiji su potpuni ulazni podaci dati na slici 11.28, vidimo na slici *11.32c*.



Glava 12

PRILOZI

U ovoj glavi dat je raznovrstan priručni materijal, neophodan za praktičan rad na programiranju. Tu je pre svega uporedni pregled FORTRAN-naredbi u prevodiocima za razne tipove računara, uporedni pregled najvažnijih karakteristika FORTRANa za razne tipove računara i tabela standardnih matematičkih funkcija. Zatim, ovde su opisane nestandardne naredbe za deklarisanje vrste i dužine promenljivih, kakve postoje samo u prevodiocima za veće računare, pomoćne naredbe za ispitivanje indikatora i pomoćne naredbe za testiranje programa. Na kraju su data rešenja zadataka s zvezdicom, iz »Vežbi« na kraju svake glave.

12.1 *POTPUN REPERTOAR NAREDBI FORTRANa IV*

Sve naredbe date su u tabeli *12.1*, u abecednom redosledu. U prvim dvema kolonama zvezdicama su označene aktivne (A) odnosno neaktivne (N) naredbe; prema definiciji iz prethodnih glava, neaktivne su one naredbe koje služe samo za davanje određenih informacija FORTRAN-prevodiocu, a ne prevode se u mašinske instrukcije koje vrše neke operacije. U ostalim kolonama označene su zvezdicama, za pojedine tipove računara, one naredbe koje postoje u pripadajućem prevodiocu.

Tabela 12.1, str. 250. i 251, može nam korisno poslužiti za orijentaciju koje naredbe možemo koristiti kada pripremamo program za prevođenje i izvršavanje u konkretnom računskom centru. Takođe, ako raspolažemo gotovim FORTRAN-programom, koji je pisan za određeni tip računara i na tom računaru proveren, pomoću ove tabele možemo videti da li isti program možemo neposredno upotrebiti na računaru sa kojim raspolažemo odnosno koje naredbe moramo prethodno promeniti.

Tabela 12.1 Potpun repertoar naredbi FORTRANa IV

	Naredba		IBM 360		UNIVAC	CII	NCR	ICL	CDC	IBM	UNIVAC	Honey-	Bull-	IBM	IBM
	aktiv- na	neak- tivna	prevo- dilac D i E	prevo- dilac H	9300	10070	315	System 4	3300	1130	1050	well 200	GE Gamma 30	1401	705
a = b	x		x	x	x	x	x	x	x	x	x	x	x	x	x
ASSIGN i TO n	x			x		x		x	x		x	x	x		x
BACKSPACE i	x		x	x	x	x	x	x	x		x	x	x	x	x
BLOCK DATA		x		x		x		x			x				
CALL	x		x	x	x	x	x	x	x	x	x	x	x	x	
COMMON		x	x	x	x	x	x	x	x	x	x	x	x	x	
COMPLEX		x		x		x		x	x		x				
CONTINUE	x		x	x	x	x	x	x	x	x	x	x	x	x	x
DATA		x		x		x		x	x		x	x		x	
DEFINE FILE		x	x	x						x				x	
DIMENSION		x	x	x	x	x	x	x	x	x	x	x	x	x	x
DO	x		x	x	x	x	x	x	x	x	x	x	x	x	x
DOUBLE PRECISION		x	x	x		x	x	x	x		x	x			
END		x	x	x	x	x	x	x	x	x	x	x	x	x	x
END FILE i	x		x	x	x	x	x	x	x		x	x	x	x	x
ENTRY		x		x		x		x	x						
EQUIVALENCE		x	x	x	x	x	x	x	x	x	x	x	x	x	
EXTERNAL		x	x	x	x	x		x	x	x	x	x		x	
FIND	x		x	x						x				x	
FORMAT		x	x	x	x	x	x	x	x	x	x	x	x	x	x
FUNCTION		x	x	x	x	x	x	x	x	x	x	x	x	x	
GO TO n	x		x	x	x	x	x	x	x	x	x	x	x	x	x
GO TO n, $(n_1,n_2,\ldots,n_m)$	x			x		x		x	x		x	x	x		x
GO TO $(n_1,n_2,\ldots,n_m)$, i	x		x	x	x	x	x	x	x	x	x	x	x	x	x
IF (a) n_1,n_2,n_3	x		x	x	x	x	x	x	x	x	x	x	x	x	x
IF (t) s	x			x		x	x	x	x		x	x		x	

IMPLICIT		x		x		x		x						
INTEGER		x	x	x	x	x	x	x	x	x	x	x	x	
LOGICAL		x		x		x	x	x	x		x	x	x	
NAMELIST		x		x		x		x						
PAUSE	x		x	x	x	x	x	x	x	x	x	x	x	x
PAUSE n	x		x	x	x	x	x	x	x	x	x	x	x	x
PAUSE 'poruka'	x			x		x		x						
PRINT n, lista	x			x		x		x	x		x		x	x
PUNCH n, lista	x			x		x		x	x		x		x	x
READ (i,n) lista	x		x	x	x	x	x	x	x	x	x	x	x	x
READ n, lista	x			x				x	x		x			
READ (i), lista	x		x	x		x		x	x		x	x	x	x
READ (j'e,n) lista	x			x									x	
READ (j'e) lista	x			x						x			x	
READ (i,x)	x			x		x		x						
REAL		x	x	x	x	x	x	x	x	x	x	x		x	
RETURN	x		x	x	x	x	x	x	x	x	x	x	x	x	
RETURN i	x			x		x	x	x							
REWIND i	x		x	x	x	x	x	x	x		x	x	x	x	x
STOP	x		x	x	x	x	x	x	x	x	x	x	x	x	x
STOP n	x		x	x	x	x	x	x	x		x	x	x		x
SUBROUTINE		x	x	x	x	x	x	x	x	x	x	x	x	x	
WRITE (i,n) lista	x		x	x	x	x	x	x	x	x	x	x	x	x	x
WRITE n, lista	x			x				x			x		x		
WRITE (i), lista	x		x	x		x		x	x		x	x	x	x	x
WRITE (j'e,n) lista	x			x										x	
WRITE (j'e) lista	x			x						x				x	
WRITE (i,x) lista	x			x		x		x							

1) samo u obliku READ INPUT TAPE i,n,lista odnosno WRITE OUTPUT TAPE i,n,lista (samo za magnetnu traku)
2) odnosi se samo na čitač kartica
3) samo u obliku READ TAPE i,lista odnosno WRITE TAPE i,lista (samo za magnetnu traku)

12.2 *UPOREDNI PREGLED GLAVNIH KARAKTERISTIKA*

Tabela 12.2 daje podatke o karakteristikama FORTRAN-prevodilaca istih tipova računara, za koje je naveden repertoar naredbi u tabeli 12.1. Karakteristike koje su propisane standardima* za osnovni i potpuni FORTRAN IV vidimo u prvim dvema kolonama tabele.

Tabela 12.2 dopunjuje informacije iz tabele 12.1 i služi u istu svrhu. Treba ipak ponovo napomenuti, kao što je već ranije naglašeno, da pored svih ovih informacija mora biti poznat i sastav (konfiguracija) računara za koji se piše ili adaptira FORTRAN-program. Takve informacije mogu se dobiti neposredno u računskom centru.

12.3 *DRUGI OBLIK NAREDBI ZA DEKLARISANJE VRSTE*

FORTRAN-prevodioci za neke veće računare** dozvoljavaju da se u poznatim eksplicitnim naredbama za deklarisanje vrste promenljivih i u naredbi IMPLICIT ujedno definiše i dužina (tačnost) promenljive, tj. broj pozicija koje zauzima vrednost promenljive u memoriji. Za svaku vrstu postoje dve dužine, od kojih je jedna standardna i FORTRAN-prevodilac je podrazumeva i ako nije napisana u naredbi. Time je omogućeno da se gotovi izvorni programi, pisani za manji računar, mogu bez izmena prevoditi i izvršavati na većim računarima, čiji FORTRAN-prevodioci raspolažu sa ovakvim naredbama.

Definicija vrste, zajedno sa odgovarajućim varijantama dužine, je sledeća:

Vrsta	Faktor *s* (dužina)
INTEGER*s (ceo broj)	2 ili 4 (standardno je 4)
REAL*s (realan broj)	4 ili 8 (standardno je 4)
COMPLEX*s (kompleksan broj)	8 ili 16 (standardno je 8)
LOGICAL*s (logička veličina)	1 ili 4 (standardno je 4)

REAL*8 je ekvivalentno atributu DOUBLE PRECISION. Ako se, prema tome, napiše sledeća eksplicitna naredba:

REAL*8 MATR, GREDA, VRED*4, NRART(5,5)

tada su kao realni brojevi dvostruke tačnosti deklarisane promenljive MATR, GREDA i svi članovi matrice NRART. Promenljiva

* standardi ASA (American Standards Association)
** na pr. prevodilac H za IBM 360 ili prevodilac za ICL System 4

Tabela 12.2 Uporedni pregled glavnih karakteristika FORTRAN-prevodilaca

	Standardni FORTRAN IV osnovni	Standardni FORTRAN IV potpuni	IBM 360 prevodilac D i E	IBM 360 prevodilac H	UNIVAC 9300	CII 10070	NCR 315	ICL System 4	CDC 3300	IBM 1130	UNIVAC 1050	Honeywell 200	Bull-GE Gamma 30	IBM 1401	IBM 705
Broj naredbe, max	9999	99999	99999	99999	99999	99999	99999	99999	99999	99999	99999	99999	99999	99999	99999
Produžne kartice, max	5	19	19	19	19	19	19	19	19[1]	5	5	9	5	9	9
Specifikacione naredbe obavezno ispred prve aktivne naredbe	da	da	da	ne	da	ne	da	ne	da	da	da	da	da	ne	da
INTEGER konstanta, max cifara			10	10	5	10	11	10	15	5	5	20	7	20	10
INTEGER, max vrednost			$2^{31}-1$	$2^{31}-1$	$2^{15}-1$	$2^{31}-1$	$10^{11}-1$	$2^{31}-1$	$2^{47}-1$	$2^{15}-1$	$10^{5}-1$	$2^{119}-1$	10^{7}	$10^{20}-1$	10^{10}
REAL konstanta, max cifara			7	7	9	7	12	7	10	7	8	20	8	20	8
DOUBLE PRECISION konstanta, max cifara			16	16	25[2]	16	21	16	25	10	14	20	-	-	-
REAL, DOUBLE PRECISION, max vrednost			10^{75}	10^{75}	10^{49}	10^{75}	10^{150}	10^{75}	10^{308}	10^{38}	10^{49}	10^{99}	10^{99}	10^{99}	10^{99}
Simbol promenljive, max znakova	5	6	6	6	6	6	nema ograničenja	6	8	5	6	6	6	6	10
Mešoviti aritmetički izrazi			da	da	da	da	da	da	da	da	ne	ne	ne	ne	ne
GO TO sa zadatim prelazom		da	ne	da	ne	da	da	da	da	ne	da	da	da	ne	da
Logička naredba IF		da	ne	da	ne	da	da	da	da	ne	da	da	ne	da	ne
DOUBLE PRECISION, operacije		da	da	da	ne	da	da	da	da	da	da	ne	ne	ne	ne
COMPLEX, operacije		da	ne	da	ne	da	da	da	da	ne	ne	ne	ne	ne	ne
LOGICAL, operacije		da	ne	da	ne	da	da	da	da	ne	da	da	ne	da	ne
Dimenzije promenljivih u naredbama za deklarisanje vrste		da	da	da	ne	da	da	da	ne	da	da	da	ne	ne	ne
COMMON, imenovana zona		da	ne	da	ne	da	da	da	da	ne	da	da	ne	ne	ne
Matrice, max dimenzije	2	3	3	7	3	nema ogranič.	nema ogranič.	7	3	3	3	3	3	3	3
Matrice, promenljive dimenzije		da	ne	da	ne	da	da	da	da	ne	ne	ne	ne	da	ne
Indeksi negativni i nula			ne	ne	ne	ne	da	ne	ne	ne	ne	ne	ne	ne	ne
Indeks - proizvoljan izraz, sa indeksnim promenljivima			ne	ne	ne	da	da	ne	ne	ne	ne	ne	ne	ne	ne
SUBROUTINE sa više ulaza (ENTRY) i izlaza (RETURN i)			ne	da	ne	da	da	da	ne	ne	ne	ne	ne	ne	ne
DATA, naredba			ne	da	ne	da	da	da	da	ne	da	da	ne	da	ne
FORMAT-specifikacije, promenljive			ne	da	ne	da	da	da	da	ne	ne	da	ne	da	ne

1) i više, zavisi od broja operatora, broja naredbi i broja delimitera (vidi FORTRAN-priručnik CDC 3300)
2) pretvara se automatski u realnu konstantu sa 9 dekadnih cifara

VRED je jedina deklarisana kao realan broj obične tačnosti; vidimo da se zadata dužina odnosi na sve promenljive u naredbi izuzev onu, čija dužina je izričito naznačena. Sve ostale promenljive u programu, koje nisu deklarisane ovom ili nekom drugom naredbom, biće u pogledu vrste deklarisane po unutrašnjoj konvenciji, a dužina će im biti standardna (prema gornjoj definiciji).

Za računare čiji FORTRAN-prevodioci dozvoljavaju deklarisanje vrste i dužine na ovaj način, tabela 2.2 glasi:

+ - * /	INTEGER (2)	INTEGER (4)	REAL (4)	REAL (8)	COMPLEX (8)	COMPLEX (16)
INTEGER (2)	Integer (2)	Integer (4)	Real (4)	Real (8)	Complex (8)	Complex (16)
INTEGER (4)	Integer (4)	Integer (4)	Real (4)	Real (8)	Complex (8)	Complex (16)
REAL (4)	Real (4)	Real (4)	Real (4)	Real (8)	Complex (8)	Complex (16)
REAL (8)	Real (8)	Real (8)	Real (8)	Real (8)	Complex (16)	Complex (16)
COMPLEX (8)	Complex (8)	Complex (8)	Complex (8)	Complex (16)	Complex (16)	Complex (16)
COMPLEX (16)	Complex (16)	Complex (16)	Complex (16)	Complex (16)	Complex (16)	Complex (16)

12.4 *STANDARDNE MATEMATIČKE FUNKCIJE*

U tabeli *12.3* dat je potpun pregled standardnih matematičkih funkcija. Svi podaci u tabeli odgovaraju pravilima FORTRAN-prevodioca H za računar IBM 360*, tj. važe kod većih računara. FORTRAN-prevodioci za manje računare ne raspolažu sa funkcijama koje su u koloni »Funkcija« obeležene zvezdicom. Osim toga, kod manjih računara ne mogu se primeniti varijante svih funkcija za

* i FORTRAN-prevodioca za računar ICL System 4

kompleksne i nestandardne celobrojne argumente; atribut REAL*8 znači DOUBLE PRECISION, atribut REAL*4 znači REAL, a atribut INTEGER*4 znači INTEGER.

Gotovi potprogrami za izračunavanje svih standardnih funkcija iz tabele 12.3 jesu potprogrami FUNCTION. Oni se dele na dve vrste: one koji u koloni IO imaju oznaku I, FORTRAN-prevodilac uključuje u mašinski program na mestima gde je funkcija pozvana, dok one sa oznakom O uključuje jedanput za ceo program.

Za druge tipove računara treba videti tabelu standardnih funkcija u odgovarajućem FORTRAN-priručniku. Simboli i definicije se najčešće poklapaju sa onima u tabeli 12.3.

TABELA 12.3 STANDARDNE MATEMATIČKE FUNKCIJE

Funkcija	Naziv	Definicija	I O	Broj arg.	Vrsta argumenata	Vrsta funkcije
Eksponencijalna	EXP	e^x	O	1	REAL*4	REAL*4
	DEXP	e^x	O	1	REAL*8	REAL*8
	CEXP	e^x	O	1	COMPLEX*8	COMPLEX*8
	CDEXP	e^x	O	1	COMPLEX*16	COMPLEX*16
Prirodni logaritam	ALOG	ln x	O	1	REAL*4	REAL*4
	DLOG	ln x	O	1	REAL*8	REAL*8
	CLOG	ln x	O	1	COMPLEX*8	COMPLEX*8
	CDLOG	ln x	O	1	COMPLEX*16	COMPLEX*16
Dekadni logaritam	ALOG10	log x	O	1	REAL*4	REAL*4
	DLOG10	log x	O	1	REAL*8	REAL*8
Arkus-sinus*	ARSIN	arcsin x	O	1	REAL*4	REAL*4
	DARSIN	arcsin x	O	1	REAL*8	REAL*8
Arkus-kosinus*	ARCOS	arccos x	O	1	REAL*4	REAL*4
	DARCOS	arccos x	O	1	REAL*8	REAL*8
Arkus-tangens	ATAN	arctg x	O	1	REAL*4	REAL*4
	ATAN2	arctg x_1/x_2	O	2	REAL*4	REAL*4

(nastavak)

Funkcija	Naziv	Definicija	I 0	Broj arg.	Vrsta argumenata	Vrsta funkcije
	DATAN	arctg x	0	1	REAL*8	REAL*8
	DATAN2	arctg x_1/x_2	0	2	REAL*8	REAL*8
Sinus (arg. u radijanima)	SIN	sin x	0	1	REAL*4	REAL*4
	DSIN	sin x	0	1	REAL*8	REAL*8
	CSIN	sin x	0	1	COMPLEX*8	COMPLEX*8
	CDSIN	sin x	0	1	COMPLEX*16	COMPLEX*16
Kosinus (arg. u radijanima)	COS	cos x	0	1	REAL*4	REAL*4
	DCOS	cos x	0	1	REAL*8	REAL*8
	CCOS	cos x	0	1	COMPLEX*8	COMPLEX*8
	CDCOS	cos x	0	1	COMPLEX*16	COMPLEX*16
Tangens (arg. u radijanima)*	TAN	tg x	0	1	REAL*4	REAL*4
	DTAN	tg x	0	1	REAL*8	REAL*8
Kotangens (arg. u radijanima)*	COTAN	ctg x	0	1	REAL*4	REAL*4
	DCOTAN	ctg x	0	1	REAL*8	REAL*8
Kvadratni koren	SQRT	$x^{1/2}$	0	1	REAL*4	REAL*4
	DSQRT	$x^{1/2}$	0	1	REAL*8	REAL*8
	CSQRT	$x^{1/2}$	0	1	COMPLEX*8	COMPLEX*8
	CDSQRT	$x^{1/2}$	0	1	COMPLEX*16	COMPLEX*16
Hiperbol. tangens	TANH	tgh x	0	1	REAL*4	REAL*4
	DTANH	tgh x	0	1	REAL*8	REAL*8
Najveća vrednost	AMAX0	max($x_1, x_2, ..$)	0	≥ 2	INTEGER*4	REAL*4
	AMAX1		0	≥ 2	REAL*4	REAL*4
	MAX0		0	≥ 2	INTEGER*4	INTEGER*4
	MAX1		0	≥ 2	REAL*4	INTEGER*4
	DMAX1		0	≥ 2	REAL*8	REAL*8
Najmanja vrednost	AMIN0	min($x_1, x_2, ..$)	0	≥ 2	INTEGER*4	REAL*4
	AMIN1		0	≥ 2	REAL*4	REAL*4
	MIN0		0	≥ 2	INTEGER*4	INTEGER*4
	MIN1		0	≥ 2	REAL*4	INTEGER*4
	DMIN1		0	≥ 2	REAL*8	REAL*8

(nastavak)

Funkcija	Naziv	Definicija	I O	Broj arg.	Vrsta argumenata	Vrsta funkcije
Pokretni zarez	FLOAT	pretvaranje celog broja	I	1	INTEGER*4	REAL*4
	DFLOAT	u decimalan	I	1	INTEGER*4	REAL*8
Fiksni zarez	IFIX	pretvaranje decimalnog	I	1	REAL*4	INTEGER*4
	HFIX	broja u ceo	I	1	REAL*4	INTEGER*2
Prenos algebar. znaka	SIGN	algebarski	I	2	REAL*4	REAL*4
	ISIGN	znak od x_2	I	2	INTEGER*4	INTEGER*4
	DSIGN	ispred $\|x_1\|$	I	2	REAL*8	REAL*8
Pozitivna razlika	DIM	$x_1-\min(x_1,x_2)$	I	2	REAL*4	REAL*4
	IDIM		I	2	INTEGER*4	INTEGER*4
Hiperbol. sinus*	SINH	sinh x	O	1	REAL*4	REAL*4
	DSINH	sinh x	O	1	REAL*8	REAL*8
Hiperbol. kosinus*	COSH	cosh x	O	1	REAL*4	REAL*4
	DCOSH	cosh x	O	1	REAL*8	REAL*8
Funkcija greške*	ERF	$\frac{2}{\sqrt{\pi}}\int_0^x e^{-u^2} du$	O	1	REAL*4	REAL*4
	DERF		O	1	REAL*8	REAL*8
Komplem. funkcije greške*	ERFC	1 - erf (x)	O	1	REAL*4	REAL*4
	DERFC		O	1	REAL*8	REAL*8
Gama-funkcija*	GAMMA	$\int_0^\infty u^{x-1} e^{-u} du$	O	1	REAL*4	REAL*4
	DGAMMA		O	1	REAL*8	REAL*8
Ln(gama-funkcija)*	ALGAMA	$\ln \Gamma(x)$	O	1	REAL*4	REAL*4
	DLGAMA		O	1	REAL*8	REAL*8
Modul	MOD	$x_1 (\mathrm{mod}\ x_2)$	I	2	INTEGER*4	INTEGER*4
	AMOD	$x_1 - \left[\frac{x_1}{x_2}\right] * x_2$	I	2	REAL*4	REAL*4
	DMOD	gde je $[X]$ = max ceo br. $\leq \|X\|$	I	2	REAL*8	REAL*8

(nastavak)

Funkcija	Naziv	Definicija	I O	Broj arg.	Vrsta argumenata	Vrsta funkcije
Apsolutna vrednost	IABS	|x|	I	1	INTEGER*4	INTEGER*4
	ABS		I	1	REAL*4	REAL*4
	DABS		I	1	REAL*8	REAL*8
	CABS	$(a^2+b^2)^{1/2}$	O	1	COMPLEX*8	REAL*4
	CDABS	za a + bi	O	1	COMPLEX*16	REAL*8
Odbacivanje decimala	INT	max ceo br.	I	1	REAL*4	INTEGER*4
	AINT	<|x| sa alg.	I	1	REAL*4	REAL*4
	IDINT	znakom od x	I	1	REAL*8	INTEGER*4
Značajniji deo broja REAL*8	SNGL		I	1	REAL*8	REAL*4
Realni deo kompleksnog broja *	REAL		I	1	COMPLEX*8	REAL*4
Imaginarni deo kompleksnog broja *	AIMAG		I	1	COMPLEX*8	REAL*4
Broj REAL*4 u obliku REAL*8	DBLE		I	1	REAL*4	REAL*8
Dva realna broja u obliku kompleksnog *	CMPLX	$c = x_1 + x_2 i$	I	2	REAL*4	COMPLEX*8
	DCMPLX		I	2	REAL*8	COMPLEX*16
Konjugovani kompleksni broj *	CONJG	$c_c = a - bi$ za $c = a + bi$	I	1	COMPLEX*8	COMPLEX*8
	DCONJG		I	1	COMPLEX*16	COMPLEX*16

12.5 *POMOĆNE NAREDBE ZA ISPITIVANJE INDIKATORA*

Indikatori su određene pozicije u memoriji računara, koje pod određenim uslovima dobijaju odgovarajući sadržaj. Ispitivanjem sadržaja indikatora može se usmeriti tok programa.

Indikatora opšte namene ima u FORTRANu četiri. Njihov sadržaj može biti samo 0 ili 1; ako je sadržaj nula, za indikator se kaže da je isključen, a ako je jedinica, indikator se smatra uključenim. Stanje ovih indikatora (sadržaj 0 ili 1) menja se naredbom:

```
CALL SLITE (i)
```

gde je i jedan izraz, čija vrednost je ceo broj. Ako je $i=0$, sva četiri indikatora se isključuju; ako je $i=1,2,3$ ili 4, uključuje se odgovarajući indikator. Naredbom:

```
CALL SLITET (i,j)
```

gde je i izraz prema gornjoj definiciji, vrši se ispitivanje određenog indikatora (1, 2, 3 i 4). Ako je taj indikator prilikom ispitivanja bio uključen, celobrojna promenljiva j dobija vrednost 1, inače 2. Posle ispitivanja, indikator se isključuje.

Na primer, kada se u programu:

```
      . . . . . .
      CALL SLITE (0)
      . . . . . .
      CALL SLITE (3)
      . . . . . .
      CALL SLITET (3,NR)
      GO TO (5,12), NR
   12 WRITE (6,22) (RES(K), K = 1,8)
      . . . . . .
    5 . . . . . .
```

bude izvršavala naredba CALL SLITET (3,NR), promenljiva NR će imati vrednost 1 ili 2, zavisno od stanja indikatora 3: ako je indikator bio uključen (NR=1), program će se nastaviti izvršavanjem naredbe 12 i indikator 3 će biti isključen, a ako je bio isključen (NR=2), izvršiće se naredba 5 i indikator 3 će ostati isključen.

Indikator prekoračenja kapaciteta može u izvesnim slučajevima da spreči automatsku indikaciju greške (praćenu neopozivim prekidom rada programa) kada u programu neka vrednost treba da premaši najveću ili podbaci najmanju dozvoljenu veličinu realnog broja. Stanje ovog indikatora menja se automatski, po bilo kojoj naredbi u programu, a ispituje se naredbom:

CALL OVERFL (j)

gde je j celobrojna promenljiva, koja u normalnom slučaju ima vrednost $j=2$ (nema pokušaja prekoračenja), pri pokušaju prekoračenja najvećeg dozvoljenog broja $j=1$, a pri pokušaju prekoračenja najmanjeg dozvoljenog broja $j=3$. Posle utvrđivanja vrednosti j, indikator se automatski isključuje.

Indikator deljenja nulom može da indicira pokušaj deljenja nulom, na analogni način kao prethodno opisani indikator prekoračenja. Ispituje se naredbom:

CALL DVCHK (j)

gde je j celobrojna promenljiva, koja dobija vrednost 1 ako je indikator uključen, a vrednost 2 ako je isključen. Posle utvrđivanja vrednosti j, indikator se automatski isključuje. (Ovaj indikator se na primer mogao koristiti u slučaju koji je opisan u komentaru iza drugog primera u odeljku 7.4).

12.6 *POMOĆNE NAREDBE ZA TESTIRANJE PROGRAMA*

Kao i naredbe za ispitivanje indikatora, to su naredbe CALL, kojima se pozivaju gotovi potprogrami SUBROUTINE.

Naredba:

CALL PDUMP (a_1,b_1,f_1, ..., a_n,b_n,f_n)

poziva potprogram PDUMP, koji upisuje u standardnu izlaznu datoteku sadržaj zona u memoriji između adresa a_1 i b_1, a_2 i b_2, ... a_n i b_n. Adrese mogu biti celobrojne promenljive ili konstante, dok su parametri f_1, f_2, ..., f_n isključivo celobrojne konstante, čije

vrednosti određuju oblik u kome će sadržaj zona biti upisan na spoljni nosilac i to:

F	Oblik za računar: IBM 360, veći	IBM 360, manji
0	heksadecimalni	heksadecimalni
1	LOGICAL 1	
2	LOGICAL 4	
3	INTEGER 2	
4	INTEGER 4	INTEGER
5	REAL 4	REAL
6	REAL 8	DOUBLE PRECISION
7	COMPLEX 8	
8	COMPLEX 16	
9	literal	

Posle izvršavanja potprograma PDUMP, nastavlja se na normalan način izvršavanje programa iz kojeg je potprogram pozvan.

Druga naredba ove vrste:

CALL DUMP (a_1,b_1,f_1, ..., a_n,b_n,f_n)

ima sve iste karakteristike kao i prethodna, samo se posle izvršavanja potprograma DUMP više ne može nastaviti rad programa koji ga je pozvao.

12.7 *REŠENJA ZADATAKA IZ »VEŽBI«*

Ovde su data rešenja za one zadatke koji su, u odeljcima »VEŽBE« na kraju svake glave, obeleženi zvezdicom. Prikazana rešenja često nisu jedina moguća; kod pojedinih je ukazano i na druge alternative. Očekuje se da će čitalac i sam nalaziti originalne, korektne odgovore.

Glava 2

2. 2,768.035.0 (nije dozvoljeno odvajanje dekadnih klasa pomoću tačke i zareza)
   +276 (nema decimalne tačke)
   1.8E86 (apsolutna vrednost karakteristike veća od dozvoljene)
   3E—7 (nema decimalne tačke)

3. Da

5. a. X + Y**3
   d. A + B/C
   f. A + B/(C + D)
   h. ((A + B)/(C + D))**2 + X**2
   j. 1.0 + X + X**2/2.0 + X**3/6.0
   k. (X/Y)**(G — 1.0)

6. b. (X + 2.0)/(Y + 4.0)
   ili
   (X + 2.)/(Y + 4.)
   e. (X+A+3.1416)**2/(2.0*Z)
   ili
   0.5*(X+A+3.1416)**2/Z
   g. (X/Y)**(R — 1.0)
   ili
   (X/Y)**(R — 1.)
   j. A + X*(B + X*(C +D*X))

7. c. kompleksni broj, oba dela realni brojevi dvostruke tačnosti
   e. kompleksni broj, oba dela realni brojevi obične tačnosti
   f. kompleksni broj, oba dela realni brojevi obične tačnosti
8. b. realni broj obične tačnosti
   d. realni broj obične tačnosti
   g. kompleksni broj, oba dela realni brojevi obične tačnosti

Glava 3

1. c. $P = 0.505606 \cdot 10^2$ — realni broj obične tačnosti
   d. K = 50 — ceo broj
   g. $Z = 0.0 \cdot 10^0$ — realni broj obične tačnosti
   m. J = 0 — ceo broj
2. a. $B = 0.13 \cdot 10^2$ — realni broj obične tačnosti
   b. $B = 0.0 \cdot 10^0$ — realni broj obične tačnosti
   e. L = 4 — ceo broj
   f. $B = 0.4 \cdot 10^1$ — realni broj obične tačnosti
   k. $B = 0.9999999 \cdot 10^0$ — realni broj obične tačnosti
   m. $B = 0.8 \cdot 10^1$ — realni broj obične tačnosti
   o. L = 5 — ceo broj
4. $A = 0.1325 \cdot 10^2$ — realni broj obične tačnosti
5. a. S = 2.0*P*R*SIN(3.14159/RO)
   e. U = —COS(X)**(P+1.0)/(P+1.0)
   f. EM2 = 0.5*ALOG((1.0+SIN(X))/(1.0—SIN(X)))
   i. E = X*ATAN(X/A) — 0.5*A*ALOG (A**2 + X**2)
   l. Q = (2.0/(3.14159*X))**0.5*SIN(X)
   ili
   Q = (2.0/3.14159/X)**(1.0/2.0)*SIN(X)
   ili
   Q = SQRT(2.0/(3.14159*X))*SIN(X)

Pošto je $(2/\pi)^{1/2}(1/x)^{1/2} = \sqrt{2/\pi} / \sqrt{x} = 0.7978846/\sqrt{x}$, ceo izraz se može napisati kompaktnije i sa manjim brojem operacija (brže izvršavanje naredbe u računaru!):

```
Q = 0.7978846/SQRT(X)*SIN(X)
```

Glava 4

```
2.      READ (5,16) A, B, C
   16 FORMAT (3F10.0)
      S = (A + B + C)/2.0
      P = SQRT(S*(S—A)*(S—B)*(S—C))
      WRITE (6,12) A, B, C, S, P
   12 FORMAT (5E20.7)
5.      READ (5,9) A, B, C
    9 FORMAT (3F10.0)
      X1 = (0.5/A)*(—B + SQRT(B*B — 4.0*A*C))
      X2 = (0.5/A)*(—B — SQRT(B*B — 4.0*A*C))
      WRITE (6,13) A, B, C, X1, X2
   13 FORMAT (5E20.7)
```

Da se ne bi dva puta izračunavala vrednost kvadratnog korena za isti argument, bolje je ako se napiše:

```
      READ (5,9) A, B, C
    9 FORMAT (3F10.0)
      DISKR = SQRT(B*B — 4.0*A*C)
      X1 = (0.5/A)*(—B + DISKR)
      X2 = (0.5/A)*(—B — DISKR)
      WRITE (6,13) A, B, C, X1, X2
   13 FORMAT (5E20.7)
9.      READ (5,36) A, X, S
   36 FORMAT (3F10.0)
      Y = SQRT(X**2 — A**2)
      Z = 0.5*X*S — 0.5*A**2*ALOG(ABS(X + S))
      WRITE (6,48) A, X, S, Y, Z
   48 FORMAT (5E20.7)
```

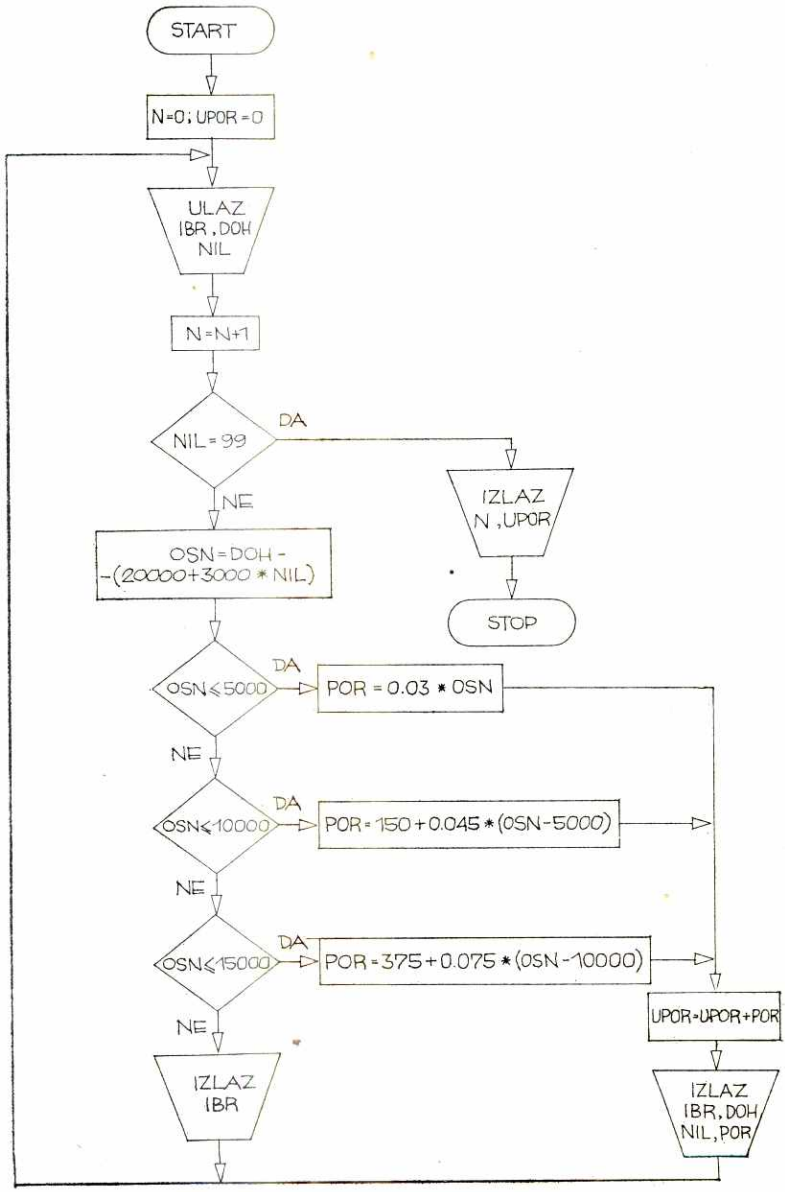
Glava 5

```
1.      IF (X — Y) 55, 55, 77
   55 AMAX = Y
      GO TO 8
```

```
      77 AMAX = X
       8 . . . . . .
 4.  101 IF (GAMA — 6.2832) 103, 102, 102
     102 GAMA = GAMA — 6.2832
         GO TO 101
     103 . . . . . .
 5.      IF (ABS(XREAL) — 1.0) 11, 22, 22
      11 IF (ABS(XIMAG) — 1.0) 21, 22, 22
8.a.    IF (X — 2.499) 52, 51, 66
      66 IF (X — 2.501) 51, 51, 52
  b.    IF (ABS(X — 2.5) — 0.001) 51, 51, 52
 9.     IF (M — 3) 654, 2, 654
       2 IF (R — S) 827, 583, 583
10.    GO TO (161,162,161,162,161,163,163,163,163), J
     333 . . . . . .
12.    X = 1.0
      33 Y = X*(24.6 + X*(16.7 — 8.32*X))
         WRITE (6,44) X, Y
      44 FORMAT (2E20.7)
         IF (X — 49.9) 55, 66, 66
      55 X = X + 0.1
         GO TO 33
      66 . . . . . .
```

Glava 6

```
1.4.  101 IF (GAMA.LT.6.2832) GO TO 103
          GAMA = GAMA — 6.2832
          GO TO 101
      103 . . . . . .
1.5.      IF ((ABS(XREAL).LT.1.0).AND.(ABS(XIMAG).LT.1.0)
          1GO TO 21
       22 . . . . . .
1.8. a.   IF (X.GE.2.499.AND.X.LE.2.501) GO TO 51
       52 . . . . . .
   b.     IF (ABS(X—2.5).LE.0.001) GO TO 51
       52 . . . . . .
3.a.      IF (0.999.LE.X.AND.X.LE.1.001) STOP
          GO TO 69
  b.      IF (ABS(X—1.0).LE.0.001) STOP
          GO TO 69
```



Slika 12.1 Dijagram progama za obračun poreza

4.
```
      IF (ABS(XREAL).LT.1.0.AND.ABS(XIMAG).LT.1.0)
     1X2 = 1.0
```

5.
```
      Y = (A.AND.(B.OR.C)).OR.(D.AND.E)
```

6. Vidi dijagram na slici 12.1

```
    1 FORMAT (I6, F8.2, I2, F6.2)
      N = 0
      UPOR = 0.0
    2 READ (5,1) IBR, DOH, NIL
      N = N + 1
      IF (NIL.EQ.99) GO TO 8
      OSN = DOH — (20000 + 3000*NIL)
      IF (OSN.GT.15000) GO TO 7
      IF (OSN.LE.5000) GO TO 3
      IF (OSN.LE.10000) GO TO 4
      IF (OSN.LE.15000) GO TO 5
    3 POR = 0.03*OSN
      GO TO 6
    4 POR = 150.0 + 0.045*(OSN—5000)
      GO TO 6
    5 POR = 375.0 + 0.075*(OSN—10000)
      GO TO 6
    6 UPOR = UPOR + POR
      WRITE (7,1) IBR, DOH, NIL, POR
      GO TO 2
    7 WRITE (7,1) IBR
      GO TO 2
    8 WRITE (7,9) N, UPOR
    9 FORMAT (I8, F16.2)
      STOP 111
      END
```

Glava 7

1.
```
      DIMENSION X(3)
      P = SQRT(X(1)**2 + X(2)**2 + X(3)**2)
```

3.
```
      DIMENSION (A(2,2), B(2,2), C(2,2)
      C(1,1) = A(1,1)*B(1,1) + A(1,2)*B(2,1)
      C(1,2) = A(1,1)*B(1,2) + A(1,2)*B(2,2)
      C(2,1) = A(2,1)*B(1,1) + A(2,2)*B(2,1)
      C(2,2) = A(2,1)*B(1,2) + A(2,2)*B(2,2)
```

```
5.      DOUBLE PRECISION A, B
        DIMENSION A(30), B(30)
        I = 1
        D = 0.0
   56   D = D + (A(I) — B(I))**2
        IF (I—30) 57, 58, 58
   57   I = I + 1
        GO TO 56
   58   D = SQRT(D)
6.      DIMENSION X(50), DX(49)
        I = 1
   13   DX(I) = X(I+1) — X(I)
        IF (I—49) 16, 18, 18
   16   I = I + 1
        GO TO 13
   18   . . . . . .
9.      DIMENSION Y(50)
        S = Y(I) + U*(Y(I+1) — Y(I—1))/2.0 +
        1U**2/2.0*(Y(I + 1) — 2.0*Y(I) + Y(I—1))
11.     DIMENSION A(7), B(7)
   21   FORMAT (7F10.0)
        READ (5,21) A
        READ (5,21) B
        I = 1
        C = 0.0
    4   C = C + A(I)*B(I)
        I = I + 1
        IF (I—7) 4, 4, 5
    5   C = SQRT(C)
        WRITE (6,22) C
   22   FORMAT (1PE20.7)
13.     COMPLEX C
        DIMENSION C(30)
        I = 1
        SUMABS = 0.0
   36   SUMABS = SUMABS + CABS(C(I))
        I = I + 1
        IF (I.LE.30) GO TO 36
15.     LOGICAL BUL
        DIMENSION BUL(40)
        INTEGER TRUE, FALSE
        I = 1
```

```
      TRUE = 0
      FALSE = 0
   33 IF (BUL(I)) GO TO 22
      FALSE = FALSE + 1
      GO TO 11
   22 TRUE = TRUE + 1
   11 I = I + 1
      IF (I.NE.41) GO TO 33
```

Glava 8

```
1.    DIMENSION AX4(50)
      SUMAX4 = 0.0
      DO 28 I = 1,50
   28 SUMAX4 = SUMAX4 + AX4(I)
4.    DIMENSION M(20)
      DO 48 I = 1,20
   48 M(I) = I*M(I)
5.    DIMENSION R(40), S(40), T(40)
      DO 65 I = 1,M
   65 T(I) = R(I) + S(I)
7.    DIMENSION F(50)
      M1 = M — 1
      DO 82 I = 2,M1
   82 F(I) = (F(I—1) + F(I) + F(I+1))/3.0
8.    DIMENSION B(50)
      BMAX = B(1)
      NBMAX = 1
      DO 628 I = 2,50
      IF (BMAX.GT.B(I)) GO TO 628
      BMAX = B(I)
      NBMAX = I
  628 CONTINUE
10.   DIMENSION A(15,15), X(15), B(15)
      DO 8 I = 1,15
      B(I) = 0.0
      DO 8 J = 1,15
    8 B(I) = B(I) + A(I,J)*X(J)
12.   DIMENSION AMTR(20,20)
      DPR = AMTR(1,1)
      DO 6 I = 2,20
    6 DPR = DPR*AMTR(I,I)
```

```
13.       DO 39 I = 100,300
          X = I
          X = X/100.0
       39 Y = 35.28*SQRT(1.0+X**2) + X**0.3333333 *EXP(X)
```

Glava 9

```
2.a.      I=bb15R=bb275.4S=bb—25.8T=bb134.5
3.        READ (5,12) ALFA, BETA, GAMA, DELTA
       12 FORMAT (4F10.0)
6.        READ (5,19) IKS, JOT, ARC, DET
       19 FORMAT (2I3, 2E14.6)
7.a.      FORMAT (2I7, 2F8.2)
  b.      FORMAT (2I7, 2F6.0)
  c.      FORMAT (I6, I8, 2E13.5)
  d.      FORMAT (I6, I8, 1P2E12.4)
  e.      FORMAT (3HbI=,I6,4HbbJ=,I6,4HbbX=,
         1          F6.1,4HbbY=,F6.1)
  f.      FORMAT (3HbI=,I6/3HbJ =,I6/
         1          3HbX=,F6.1/3HbY=,F6.1)
9.        READ (5,28) N, (TFE(I), I = 1,N)
       28 FORMAT (I2, 10F7.0)
11.       WRITE (6,33) P, Q, U, V
       33 FORMAT (2F12.4, 2E20.7)
14.       DIMENSION QSR(10,4)
          WRITE (6,25)
       25 FORMAT (12H1MATRICA QSR)
          DO 35 I = 1,10
       35 WRITE (6,45) (QSR(I,J), J = 1,4)
       45 FORMAT (4E20.7)
          ili bez naredbe DO:
          DIMENSION QSR(10,4)
          WRITE (6,25)
       25 FORMAT (12H1MATRICA QSR)
          WRITE (6,45) ((QSR(I,J), J=1,4),I=1,10)
       45 FORMAT (4E20.7)
15.       READ (5,15) (MESEC(I), I = 1,12)
       15 FORMAT (12A4)
          . . . . . .
          WRITE (6,16) MESEC(I)
       16 FORMAT (1Hb, A3)
```

16.a. (10F10.2)
b. (1P5E20.6) ili (1P5E20.6/1P5E20.6)
c. (F10.2//(1P3E20.6))

17.
```
      DEFINE FILE 8 (500, 100, L, ID1)
      DIMENSION M(10)
      . . . . . .
      . . . . . .
   25 FORMAT (5I20)
      WRITE (8'16,25) (M(K), K = 1,10)
```

Glava 10

1.
```
      F(X) = X**2 + SQRT(1.0 + 2.0*X + 3.0*X**2)
      A = (5.8 + Y)/F(Y)
      B = (3.2*Z + Z**4)/F(Z)
      C = SIN(Y)/F(Y**2)
      D = 1.0/F(SIN(Y))
```

4.
```
      FUNCTION Y(X)
      IF (X) 6, 7, 8
    6 Y = 1.0 + SQRT(1.0 + X*X)
      RETURN
    7 Y = 0.0
      RETURN
    8 Y = 1.0 — SQRT(1.0 + X*X)
      RETURN
      END
```

6.
```
      FUNCTION SUMABS(A,M,N)
      DIMENSION A(M,N)
      SUMABS = 0.0
      DO 53 I = 1,M
      DO 53 J = 1,N
   53 SUMABS = SUAMBS + A(I,J)
      RETURN
      END
```

7.
```
      SUBROUTINE PROSEK(A,N,SRED,NULA)
      DIMENSION A(50)
      SRED = 0.0
      NULA = 0
      DO 25 I = 1,N
      SRED = SRED + A(I)
      IF (A(I).EQ.0.0) NULA = NULA + 1
   25 CONTINUE
```

```
      EN = N
      SRED = SRED/EN
      RETURN
      END
9.    SUBROUTINE ASMAX(A,M,N,SUMA,REDN)
      DIMENSION A(M,N)
      SUMA = ZBIR(REDKOL,1,A,M,N)
      REDN = 1.0
      DO 48 I = 2,M
      AMAX = ZBIR(REDKOL,I,A,M,N)
      IF (AMAX.LE.SUMA) GO TO 48
      SUMA = AMAX
      REDN = I
48    CONTINUE
      RETURN
      END
      CALL ASMAX(PSI,15,29,PSIL,NRED)
```

LITERATURA

McCracken, D. D.,	A Guide to FORTRAN IV Programming, John Wiley & Sons, Inc., New York, 1965
Anderson, D. M.,	Computer Programming FORTRAN IV, Appleton-Century-Crofts, New York, 1966
Žokalj, M. M.,	FORTRAN-programiranje, Udruženje korisnika mašina za obradu podataka, Beograd, 1966
IBM System/360,	FORTRAN IV Language (C28-6515)
IBM System/360,	Basic FORTRAN IV Language (C28-6629-1)
Sperry Rand Corp.,	Fundamentals of FORTRAN (UP 7536)
Sperry Rand Corp.,	UNIVAC 9300 FORTRAN Supplementary Reference (UP 7549)
Honeywell, Inc.,	FORTRAN Compiler D (123.1305.001D.2-027)
IBM Student Text,	FORTRAN in the Physical Sciences (C20-1634-0)
IBM Student Text,	A FORTRAN Primer with Business Administration Exercises (C20-1605-0)

REGISTAR