HORIA DUMITRAȘCU

SĂ ÎNVĂȚĂM BASIC

EDITURA ALBATROS

Referent științific:
Dr. ing. DAN ROMAN

HORIA DUMITRAȘCU

SĂ ÎNVĂȚĂM BASIC

EDITURA ALBATROS • BUCUREȘTI
1987

1. LIMBAJUL BASIC ŞI CONTEXTUL SĂU INFORMATIC

Limbajul BASIC este limbajul de programare evoluat cel mai utilizat în microinformatică şi în informatica personală. Această carte, care se adresează începătorilor, prezintă cele mai importante aspecte ale programării în BASIC, sprijinindu-se în mod sistematic pe numeroase exemple şi aplicaţii.

Structurată în 5 capitole, cartea permite cititorului să se familiarizeze repede cu acest limbaj de programare, să asimileze conceptele şi instrucţiunile esenţiale. Se recomandă ca această carte să fie citită cu creionul în mînă sau, şi mai bine, în faţa unui terminal cuplat la un calculator, astfel încît exemplele prezentate să poată fi executate imediat, asigurînd astfel o mai bună înţelegere. Cititorul va fi probabil surprins să constate uşurinţa — pe care o va dobîndi curînd — de a manipula acest limbaj şi de a elabora secvenţe corecte de program.

În acest capitol introductiv este prezentat contextul informatic al limbajului BASIC; sînt explicate principalele elemente ale unui sistem de calcul, precum şi semnificaţia termenului de limbaj evoluat. Acest capitol va putea fi parcurs foarte repede de către cei ce posedă deja aceste noţiuni.

1.1 — ORIGINI ŞI PARTICULARITĂŢI

BASIC este limbajul cel mai utilizat în prezent în microinformatică, întrucît prezintă evidente avantaje şi este foarte răspîndit, fapt care asigură portabilitatea programelor scrise.

Limbajul BASIC s-a născut în jurul anului 1965. El a fost elaborat de profesorii John G. Nemeny şi Thomas E. Kurtz de la Dartmonth College, S.U.A. Numele său provine de la iniţialele definiţiei din limba engleză: „Beginner's All-purpose Symbolic Instruction Code", adică un cod de instrucţiuni simbolice utilizabil de toţi începătorii.

Într-adevăr, acest limbaj este reputat pentru ușurința cu care poate fi înțeles, învățat și utilizat. Cititorul se va convinge repede, încă de la primele capitole, că nu este cîtuși de puțin necesar să fie informatician ca formație, pentru a practica acest limbaj de programare. Practica acestui limbaj fa e ca informatica să devină o preocupare la îndemîna tuturor.

Limbaj evoluat Există, bineînțeles, o sumedenie de limbaje de programare evoluate cum sînt: COBOL, FORTRAN (cu care BASIC are cîteva asemănări), PASCAL (care s-a dezvoltat foarte mult în ultima perioadă), ADA (care a fost creat pentru a deveni un limbaj universal de programare, în acest adevărat turn Babel al limbajelor) etc. Toate aceste limbaje au, ca o caracteristică comună, faptul că provin din limba engleză, ceea ce poate părea un handicap cititorului care nu cunoaște această limbă; dar, așa cum se va constata repede, numărul cuvintelor ce trebuie cunoscute este atît de restrîns încît dificultatea dispare de la sine, după un minim de practică.

Limbaj interpretat. Limbaj compilat Spre deosebire de toate celelalte limbaje evoluate care sînt limbaje „compilate", BASIC este un limbaj „interpretat". Asta înseamnă că ordinele date în acest limbaj (instrucțiunile adresate calculatorului) sînt traduse de către mașină în limbajul care îi este propriu (codul binar utilizat de circuitele sale), unul după altul, un ordin tradus fiind imediat executat, înainte de traducerea următorului. În limbajele „compilate" (FORTRAN de exemplu) se face mai întîi traducerea tuturor ordinelor, numai după aceea putînd începe execuția.

Interpretor. Mod de lucru conversațional „Interpretorul" BASIC care efectuează traducerea instrucțiunilor în „limbaj mașină" (cod binar propriu circuitelor calculatorului) și comandă execuția lor, una cîte una, procură un confort considerabil: dacă detectează o eroare, o indică imediat printr-un mesaj afișat la terminal și nu după translatarea întregului program, ca în cazul limbajelor compilate. Programatorul își va corecta comanda introdusă înainte de a continua introducerea secvenței. Mai mult decît atît, o instrucțiune poate cere să se furnizeze programului date de tratat, înainte de a merge mai departe: programatorul interogat de program, prin intermediul terminalului, va introduce de la claviatură aceste date, după care va continua introducerea instrucțiunilor.

Acest mod interactiv sau „conversațional" de lucru este caracteristic limbajului BASIC și constituie unul dintre avantajele sale majore.

Limbaj de asamblare Să amintim că opus limbajelor de nivel înalt sau „evoluate" există limbajele de nivel scăzut sau „asambloarele" și bineînțeles limbajele numerice sau limbajele „mașină". Aceste limbaje de nivel scăzut sînt caracteristice unui anumit sistem; astfel, limbajul de asamblare al microprocesorului INTEL-8085 nu are nici o legătură cu asamblorul microprocesorului MOTOROLA-6800.

Limbaj de asamblare De fiecare dată cînd se va trece de la un microprocesor la altul, sau de la un calculator construit pe baza unui microprocesor (un ZILOG-Z80, de exemplu), la un calculator construit pe baza altui microprocesor (un 6502, de exemplu), va fi necesară învățarea asamblorului care îi este specific.

Portabilitate Limbajele evoluate beneficiază de un mare avantaj numit „portabilitate". Astfel, un program scris în BASIC pentru o anumită mașină va putea fi utilizat pe o altă mașină, utilizînd ca unitate centrală un alt microprocesor. Acest transfer al unui program de pe o mașină pe alta se numește „portabilitate".

Microsoft Din păcate, limbajele evoluate prezintă particularitatea existenței mai multor variante. Limbajul BASIC nu a scăpat nici el de la această regulă, în prezent existînd numeroase „dialecte" sau versiuni realizate în jurul aceluiași nucleu comun de bază, În informatica individuală versiunea de BASIC cel mai des utilizată este cea realizată de firma americană MICROSOFT din Washington.

BASIC extins Limbajul BASIC există în mai multe variante care diferă prin puterea lor. Astfel limbajul BASIC 16 K (al cărui interpretor ocupă 16 kilo-octeți de memorie) este mai puternic decît cel ce ocupă 8 sau 4 kilo-octeți. De aceea, această carte va prezenta setul de instrucțiuni al limbajului BASIC extins.

Cititorul va trebui să consulte manualul de BASIC care însoțește calculatorul său, pentru a descoperi particularitățile variantei de implementare. Astfel, variantele implementare pe calculatoarele TRS-80, APPLE-II, PET/CBM, MICRAL, HP-85, FELIX M-118, FELIX M-18, INDEPENDENT I-102 F sînt diferite, portabilitatea programelor necesitînd intervenția programatorului pentru modificarea lor.

1.2 — JARGONUL INFORMATIC

La fel ca în orice domeniu, în informatică există un jargon al specialiștilor în care se amestecă termeni cum sînt: ASCII, ROM, EPROM, CPU, CRT, bit, octet, serie-paralel etc.

A defini toți acești termeni ar însemna să facem un curs de microinformatică (și există astfel de cursuri foarte bune!). În scurta introducere tehnică prezentată în continuare acești termeni vor fi explicați doar atît cît să dea cititorului o imagine generală și să realizeze o primă familiarizare cu așa-zisul jargon informatic.

Calculator Calculatorul propriu-zis se prezintă sub forma uneia sau a mai multor plachete electronice implantate fie într-o carcasă specială, fie cel mai adesea în una dintre unitățile următoare: ecran, claviatură, unitate de disc floppy. Toate aceste plachete trebuie să fie „transparente" utilizatorului, așa cum este motorul unei bune mașini, a cărei capotă nu a fost încă ridicată.

Ecran Este un ecran catodic, tip ecran de televizor. Pe acest ecran se afișează informațiile utile, așa cum se va vedea. Servește deci, în mod exclusiv, ca mijloc de comunicare între mașină și dv.

Disc Floppy Un disc floppy este un suport magnetic capabil să stocheze programe sau date. Este deci o memorie externă calculatorului și, datorită faptului că poate stoca un volum apreciabil de date, este numită și memorie de masă.

Casetă magnetică La anumite calculatoare, unitățile de discuri floppy sînt înlocuite de unități de casetă magnetică. Calculatoarele personale pot utiliza casetofoane obișnuite.

Claviatură În cazul în care nu utilizați încă un calculator care să poată primi ordine vocale, aveți nevoie de o claviatură prin intermediul căreia să transmiteți informații mașinii. Claviatura este foarte asemănătoare celei de la mașinile de scris clasice.

Imprimanta Imprimanta este un echipament care, comandat de calculator, vă poate furniza anumite rezultate imprimate pe hîrtie, rezultate care altfel ar apărea pe ecranul catodic, și vor fi astfel fugitive.

Unitatea centrală este dispozitivul de comandă al unui microsistem, iar celelalte dispozitive se numesc periferice.

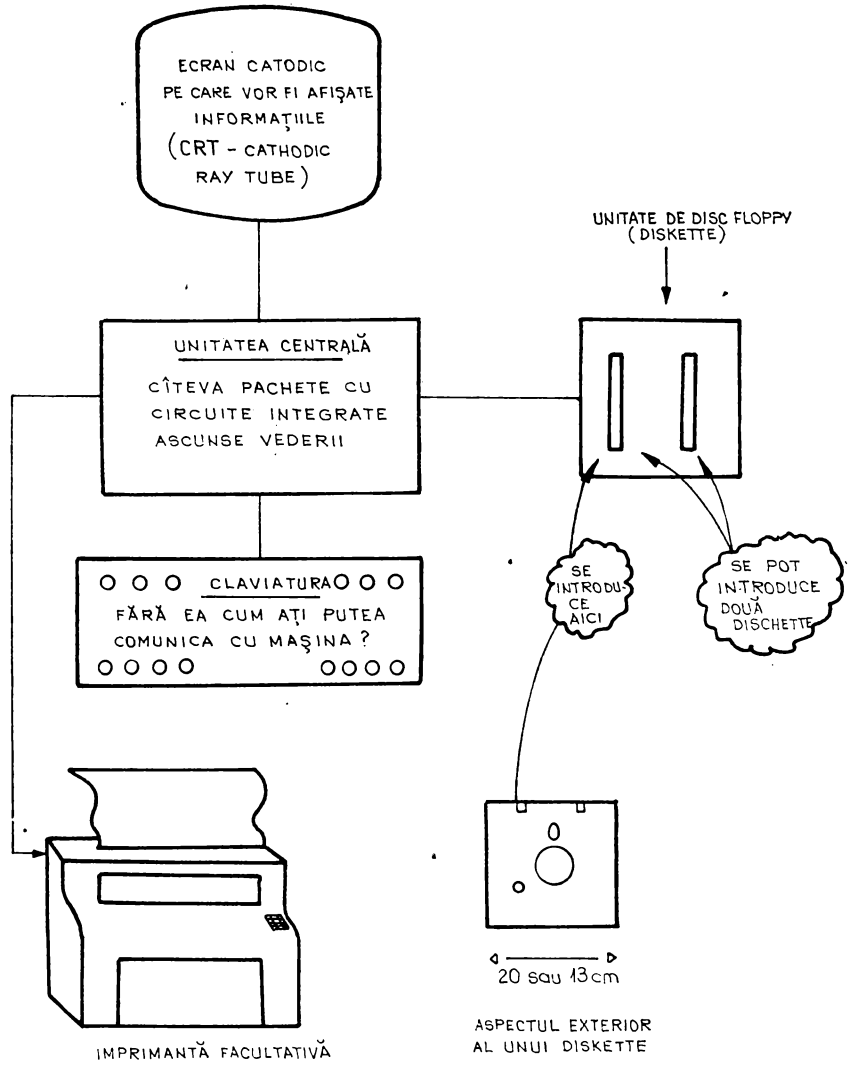

ECRAN CATODIC
PE CARE VOR FI AFIŞATE
INFORMAŢIILE
(CRT - CATHODIC
RAY TUBE)
UNITATEA CENTRALĂ
CÎTEVA PACHETE CU
CIRCUITE INTEGRATE
ASCUNSE VEDERII
CLAVIATURA
FĂRĂ EA CUM AŢI PUTEA
COMUNICA CU MAŞINA ?
UNITATE DE DISC FLOPPY
(DISKETTE)
SE
INTRODU-
CE
AICI
SE POT
INTRODUCE
DOUĂ
DISCHETTE
20 sau 13 cm
IMPRIMANTĂ FACULTATIVĂ
ASPECTUL EXTERIOR
AL UNUI DISKETTE

Fig. 1.2.1

Celulă de memorie. Adresă

Înainte de a ataca problema programării calculatorului, trebuie ca cititorul să aibă o idee precisă despre rolul memoriilor și tipul lor. Pentru a nu intra prea mult în detalii tehnice, propunem această analogie clasică: o memorie este o piesă de mobilă alcătuită din foarte multe sertare, numite „locații" sau „celule" de memorie. Fiecare dintre aceste locații este identificată printr-un număr de ordine (de exemplu: sertarul nr. 27) numit adresă. Informația existentă în locația respectivă este conținutul adresei.

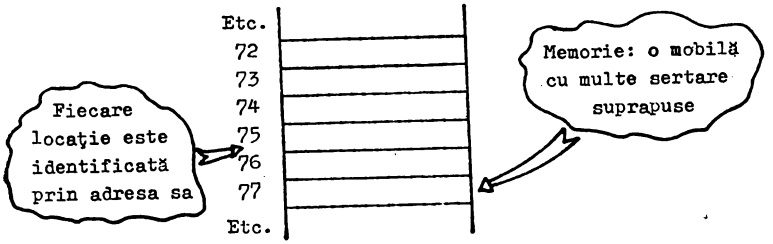


Fig. 1.2.2

De notat că: memoria nu este infinită; numărul locațiilor este limitat și trebuie cunoscut. În rest, nu trebuie să vă preocupați în legătură cu adresele acestor locații. BASIC preia această sarcină.

Memorie de masă

Există memorii interne și memorii externe. Memoriile externe sînt memorii de masă: discuri floppy, benzi sau casete magnetice. Acestea pot stoca un volum apreciabil de date.

Memorie centrală

Memoria internă sau centrală este aceea existentă pe plachete electronice și se prezintă sub formă de circuite integrate. Aceste memorii sînt foarte rapide și răspunsul este instantaneu la solicitările calculatorului, în timp ce un disc, de exemplu, este mult mai lent. Asta înseamnă că dacă programul dumneavoastră se află pe un suport magnetic extern (disc, bandă, casetă etc.), el va trebui să fie transferat în memoria centrală și numai după aceea executat.

RAM

Există anumite tipuri de memorie internă, numite RAM, care își pierd conținutul informațional atunci cînd opriți calculatorul, prin acționarea întrerupătorului sau scoatere din priză. Acest lucru vă va obliga să transferați pe un suport magnetic extern datele pe care doriți să le conservați pentru viitor. Numele acestor memorii provine de la inițialele denumirii din limba engleză: Random Access Memory.

ROM Există, de asemenea, și un alt tip de memorie internă al cărei conținut nu se pierde în momentul scoaterii de sub tensiune, dar care poate fi înscrisă o singură dată cu informații, ulterior nemaiputînd fi decît citită. Acest tip poartă numele de memorie ROM (Read Only Memory).

Memorii volatile și nevolatile Memorii vii și moarte Memoriile ROM se numesc în mod obișnuit, memorii nevolatile sau memorii moarte, în timp ce memoriile RAM se mai numesc și memorii vii sau volatile.

1.3 — AVANTAJE ȘI INCONVENIENTE ALE LIMBAJULUI BASIC

Limbajul BASIC are marele avantaj de a fi foarte ușor de învățat și manipulat și, în același timp, foarte răspîndit în mini și microinformatică sau în informatica personală. Se constată că majoritatea programelor publicate în diverse cărți și lucrări de informatică sînt scrise în BASIC; există biblioteci întregi de programe scrise în BASIC. Dacă ar fi să ne gîndim numai la industria jocurilor, la programele de gestiune sau de prelucrare a textelor, vom putea trage concluzia că BASIC este limbajul evoluat de programare cel mai răspîndit. Cu toate acestea BASIC are și dezavantaje. Iată cîteva dintre ele:

— din cauză că este interpretat și nu compilat, BASIC necesită un timp de execuție mai mare;

— fiind un limbaj de programare relativ bătrîn, BASIC nu se pretează la tehnicile programării structurate;

— execuția programului scris în BASIC necesită prezența în memorie a interpretorului, ceea ce nu e cazul limbajelor compilate. Aceasta duce la un volum mare de memorie ocupată.

În ciuda criticilor amintite, limbajul BASIC este atît de legat de informatică și este atît de utilizat încît va rămîne, probabil încă pentru mult timp, limbajul de bază al celui mai mare număr, de utilizatori.

1.4. EXERCIȚII:

Doriți să verificați dacă ați asimilat bine capitolul precedent? Răspundeți la întrebările următoare. Vă va lua foarte puțin timp și vă va ajuta să vă fixați esențialul.

1) Într-un limbaj „interpretat": A — o instrucțiune este tradusă și executată înainte de a trece la următoarea; B — execuția

nu poate începe decît după traducerea integrală a secvenței de instrucțiuni.

2) BASIC este un limbaj: A — interpretat; B — compilat.

3) Există memorii: A — numai interne; B — numai externe; C — și interne și externe.

4) Un ecran catodic este un periferic al calculatorului? A — Da; B — Nu.

5) O memorie volatilă este o memorie care: A — poate fi numai citită; B — poate fi și scrisă și citită.

6) O memorie ROM este o memorie: A — moartă; B — vie; C — nevolatilă.

Răspunsurile exercițiilor: 1 — A; 2 = A; 3 = C; 4 = A; 5 = B; 6 = A și C.

2. MODUL DE LUCRU IMEDIAT

Pentru o primă familiarizare cu BASIC se va prezenta modul „imediat". Acest mod de lucru permite utilizarea ordinatorului pe post de... calculator de buzunar pentru operații aritmetice. Acest mod mai este numit mod de lucru „delk" sau „bureau", în opoziție cu modul „program" ce va fi examinat în capitolele următoare.

2.1 — PORNIREA CALCULATORULUI

Dacă aveți acces la un calculator, utilizați-l pentru a învăța BASIC și pentru a înțelege exercițiile prezentate. Dacă nu, examinați cu multă atenție aceste exemple. Veți vedea că nu va fi dificil să simulați, pe hîrtie, cu creionul în mînă, execuția programelor prezentate.

Cei care dispun de un calculator personal (aceste cazuri sînt din ce în ce mai frecvente, avînd în vedere ieftinirea lor accelerată) vor putea să verifice programele prezentate. Dacă rezultatele obținute nu sînt cele scontate, înseamnă că ați comis o eroare. Amintiți-vă, în orice moment, că ordinatorul nu face altceva decît să vă execute ordinele fără a înțelege sensul lor: nu este sensibil decît la forma (sintaxa) ordinelor și nu la semantica lor. Astfel, el interpretează într-o manieră particulară un anumit caracter punct-și-virgulă, de exemplu. Omisiunea acestui caracter se poate traduce

printr-un alt rezultat decît cel așteptat. În ceea ce privește pornirea calculatorului, autorul nu vă poate sfătui să faceți nimic altceva decît să citiți cu atenție manualul său de utilizare livrat, împreună cu mașina, de firma constructoare.

La punerea sub tensiune nu uitați să verificați dacă priza este de 220 V. Citiți cu atenție în manualul de utilizare al calculatorului unde este comutatorul de pornire și oprire, care este programul rezident pe casetă magnetică sau diskette (disc floppy) pe care trebuie să-l încărcați în memoria centrală, pentru monitorizarea lucrului cu ordinatorul, și mai ales cum? Aceste chestiuni odată însușite, puteți trece la primele exerciții.

Dacă la primele programe vor fi chestiuni pe care nu le înțelegeți, nu vă chinuiți prea mult. Luați-le ca atare și vi le veți lămuri în continuare.

READY Să presupunem deci că ordinatorul dv. funcționează, că vă aflați în fața lui, cu manualul de utilizare alături împreună cu această carte și, eventual, o hîrtie și un creion. După încărcarea în memoria internă a interpretorului BASIC (acest lucru se face la microcalculatorul FELIX-M 118 prin simpla introducere a cuvîntului BASIC) mașina, pentru a vă asigura de bunăvoința sa, afișează pe ecran:

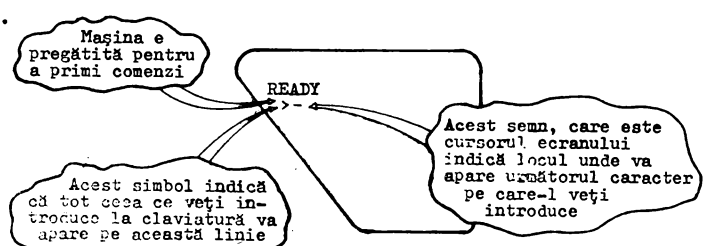


Fig. 2.1.1

Ecranul prezentat în desenul de mai sus este ecranul microcalculatorului românesc FELIX-M 118. Dacă veți lucra pe o altă mașină și imaginea care vă va apărea va fi diferită, să nu fiți surprinși. În locul cuvîntului READY alte mașini afișează OK sau HELLO sau BONJOUR.

PROMPTER Este un caracter oarecare (de exemplu „>") care indică linia curentă. Prompterul este diferit de la un interpretor la altul. BASIC-AMS folosește ca prompter caracterul „:".

CURSOR Indică locul de pe linia curentă unde va apărea următorul caracter introdus de la claviatură. Numele de cursor provine de la faptul că acesta aleargă pe întregul ecran. Apare fie sub forma unei linii subțiri, fie sub forma unui dreptunghi luminos.

2.2 — INSTRUCȚIILE MODULUI „IMEDIAT"

În modul de lucru „imediat" ordinatorul poate fi utilizat drept calculator de buzunar pentru operații aritmetice. Rezolvarea anumitor probleme se reduce la simpla calculare a unor expresii, pentru care nu este necesară scrierea unui program complet și apoi lansarea lui în execuție. De asemenea, modul „imediat" poate fi extrem de util în faza de punere la punct a programului.

Să presupunem că doriți să calculați valoarea expresiei 108×9. Pentru aceasta veți utiliza ordinatorul transmițîndu-i ordine de la claviatură. Comenzile pe care le veți introduce, la fel ca și răspunsurile mașinii vor apărea pe ecran. Prima operație pe care o veți face va fi să introduceți de la claviatură comanda:

```
PRINT 108*9
```

PRINT Comanda PRINT înseamnă imprimă sau afișează. Este utilizată atît pentru afișarea pe ecran cît și pentru imprimarea pe hîrtie, dacă microsistemul pe care-l folosiți dispune de o imprimantă. Se constată de asemenea că simbolul operației de înmulțire este asterixul, care elimină orice posibilitate de confuzie cu litera x. Ecranul va arăta:

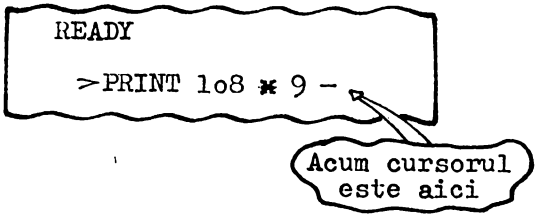


Fig. 2.1.2

CARRIAGE RETURN sau ENTER O comandă introdusă de la claviatură este citită de calculator numai după introducerea unui caracter terminator. Astfel de caractere sînt CARRIAGE RETURN (CR) sau ENTER.

Vom obține:

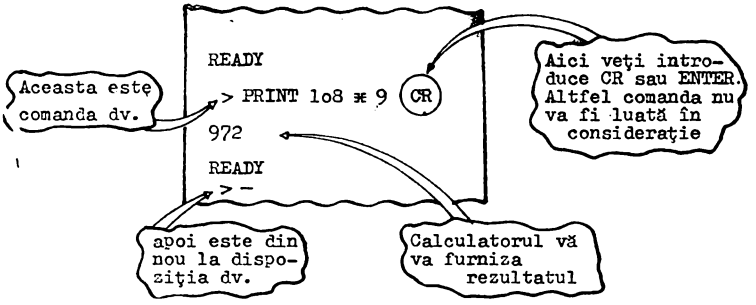


Fig. 2.1.3

Să considerăm un alt exemplu mai complicat (oh... numai cu puțin mai complicat!):

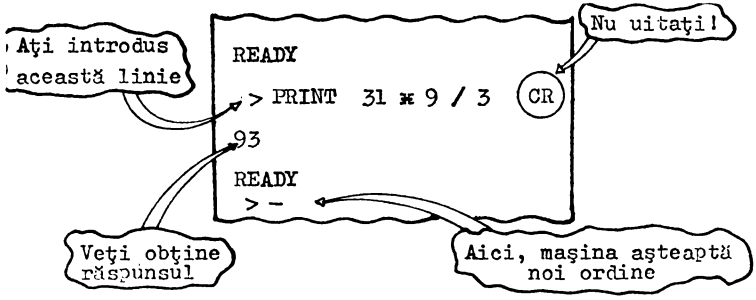


Fig. 2.1.4

Notați că bara oblică simbolizează operația de împărțire.

În limbajul BASIC ghilimelele au un rol special. Să luăm un exemplu:

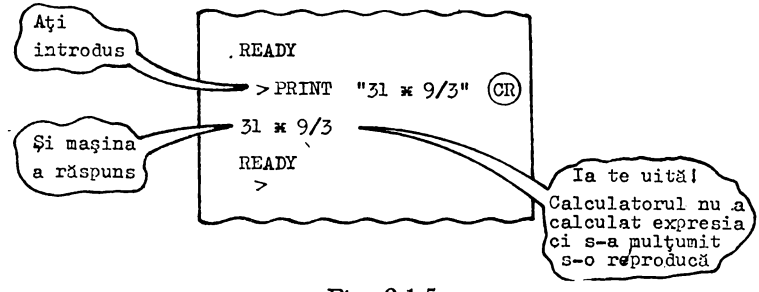


Fig. 2.1.5

Într-adevăr, atunci cînd întîlnește ghilimele într-un ordin PRINT, calculatorul va reproduce pe ecran textul încadrat.

> O comandă PRINT urmată de o expresie sau text între ghilimele se traduce prin afișarea sau tipărirea textului aflat între ghilimele.

Alte exemple ale instrucțiunii PRINT executate în mod imediat:

PRINT SIN (2*PI) — funcția sinus
PRINT SQR (64) — rădăcină pătrată
PRINT RND (*) — generator de numere pseudoaleatoare.

Toate aceste funcții vor fi definite în capitolele următoare.

În BASIC, la fel ca și în celelalte limbaje evoluate, se definesc constante și variabile. O variabilă este identificată printr-un număr de maximum 2 caractere, dintre care primul este o literă, iar al doilea este o cifră. O constantă este o valoare numerică ce nu se modifică în cursul execuției programului. O *instrucțiune de atribuire* este o instrucțiune prin care se atribuie unei variabile o anumită valoare, care este fie o constantă, fie valoarea unei alte variabile, fie valoarea unei expresii calculabile în care intervin constante și variabile. Din punct de vedere fizic această atribuire se traduce prin depunerea, la o anumită adresă de memorie, a unei valori numerice a unei constante sau a unei variabile. În cazul unei instrucțiuni de atribuire între două variabile, are loc un transfer al conținutului unei adrese la altă adresă.

LET Fie o variabilă K. Acestei variabile i se poate atribui o valoare numerică utilizînd instrucțiunea LET:

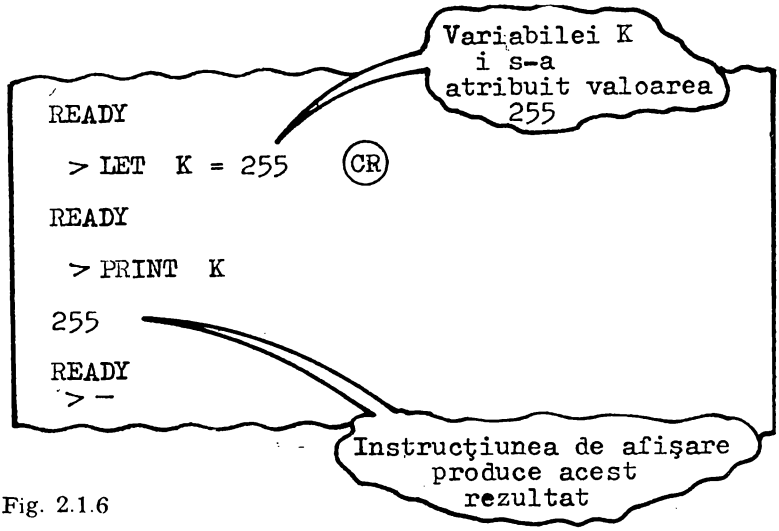


Fig. 2.1.6

În numeroase implementări ale limbajului BASIC ordinul LET poate fi omis, el fiind opțional. Așa stau lucrurile în cazul microcalculatoarelor românești FELIX-M.18, FELIX-M.118 și al minicalculatoarelor INDEPENDENT-I 100, I-102 F.

Exemplu: LET K = 255 echivalent cu K = 255.

În alte situații, cum este cazul microcalculatorului ZX-81, ordinul LET este obligatoriu.

Exemplu: K = 255 produce o eroare.

Modul imediat permite utilizarea ordinatorului ca mașină de calcul. În acest mod se pot realiza operații simple sau complexe, fără posibilitatea de memorare și reluare a secvenței de program. Setul de instrucțiuni al acestui mod de lucru cuprinde în afară de LET și PRINT următoarele instrucțiuni: INPUT, DIM, END, CALL, MAT Acestea vor fi prezentate mai tîrziu.

2.3 — EXERCIȚII

1) Ce efect va avea execuția instrucțiunilor următoare? A — PRINT „N"; B — PRINT 9*8; C — PRINT 512/16; D — PRINT „PRINT.

2) Ce va produce execuția secvenței următoare? LET K = = 512/16; PRINT „K = "; PRINT K.

3) Ce efect va produce execuția următoarelor secvențe? A — K = 3; PRINT K — K. B — K = 3; PRINT K — 6. C — PRINT K 1.

Răspunsurile exercițiilor: 1. A : N; B : 72; C : 32; D : PRINT. 2. K = 32 3. A : 0; B : —3.

C = nu există nicio variabilă K 1 definită anterior. Se va afișa valoarea 0, sau eroare.

3. COMENZILE INTERPRETORULUI BASIC

Pentru a vă familiariza cu operațiile principale efectuate asupra unui program BASIC, de la scrierea și pînă la execuția sa, se vor prezenta comenzile interpretorului. Aceste comenzi sînt extrem de necesare, fără ele cum ați putea face o serie de operații de manevră, ca, de exemplu: afișarea sau modificarea unui program, încărcarea unui program de pe casetă sau diskette în memorie sau salvarea sa pe suport magnetic etc.

Aceste comenzi pot fi privite din punct de vedere al tratării, ca instrucțiuni ale modului „imediat", datorită faptului că sînt executate imediat după introducere.

3.1 — COMENZI DE EDITARE

Comenzile de editare nu afectează execuția unui program și nu fac decît să pregătească zona de memorie a utilizatorului sau să realizeze punerea la punct a programului. În categoria acestor comenzi intră comenzi de listare sau ștergere a unor secvențe de instrucțiuni, eliminarea sau atribuirea unui nume programului aflat în memorie.

LIST Comanda servește la afișarea pe terminalul utilizatorului a programului sau a unei părți a programului aflat în memorie. Să considerăm următorul exemplu: în timpul punerii la punct a unui program apare necesitatea vizualizării sale, sau numai a unei secvențe.

Un program BASIC este o secvență de instrucțiuni numerotate în ordine crescătoare. Ca număr de linie sînt admise numai valori întregi pozitive. Valoarea maximă a unui număr de linie este 32767.

În continuare, în exemplele prezentate vom suprima caracterul CR. Presupunînd că ați înțeles deja mecanismul READY, simbolul „>" și rolul cursorului, nu le vom mai indica decît foarte rar

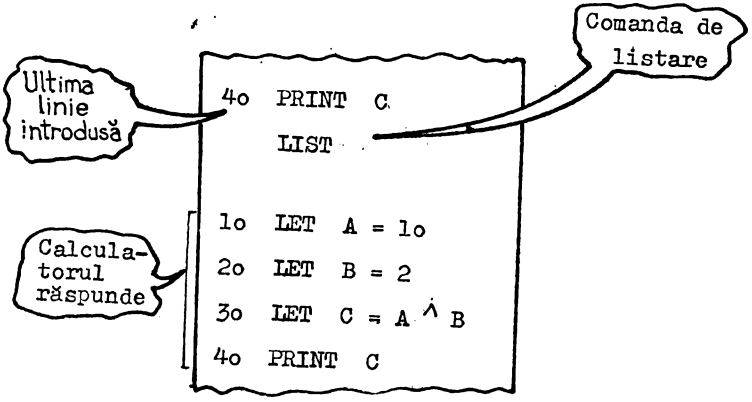


Fig. 3.1.1

Dacă s-ar fi introdus comanda LIST 20, ar fi fost vizualizate toate liniile cu număr mai mare sau egal cu 20 (veți vedea în capitolele următoare că în modul „program" fiecare instrucțiune BASIC trebuie să fie numerotată), iar dacă s-ar fi introdus LIST 30 s-ar fi vizualizat toate liniile cu nr. mai mic sau egal cu 30. De asemenea, comanda „LIST 20—40" ar fi afișat liniile cu nr. cuprins între 20 și 40.

DELETE Această comandă există numai la anumite variante ale limbajului BASIC, de exemplu la BASIC-AMS implementat pe minicalculatoarele INDEPENDENT și CORAL.

Ea servește la eliminarea dintr-un program a unei secvențe de instrucțiuni. Sintaxa comenzii este aceeași ca cea a comenzii LIST.

NEW Comanda NEW realizează eliminarea din memoria internă a programelor preexistente, punerea la zero a variabilelor și, la anumite calculatoare, ștergerea ecranului și poziționarea cursorului în poziția HOME (linia 1, coloana 1).

Această comandă funcționează atît în BASIC-AMS cît și în variantele de BASIC implementate pe microcalculatoarele TRS-8, PET/CBM, ZX-81. În BASIC-AMS comanda este însoțită, opțional, de numele atribuit noului program ce urmează a fi introdus.

SCR Comanda are același rol ca și NEW. Ea funcționează în variantele BASIC-18 și BASIC-118 implementate pe microcalculatoarele FELIX-M 18 și respectiv FELIX-M 118. Opțiunile sînt: P, dacă e vorba de ștergerea programului, sau D, dacă e vorba de ștergerea datelor. Absența opțiunii semnifică ștergerea deopotrivă a programului și a datelor.

RENAME Orice program încărcat în memoria internă dispune de un nume propriu. Comanda RENAME servește la schimbarea numelui programului curent dispus în memorie.

Modificarea unui program Există trei categorii de acțiuni prin care se poate interveni asupra unui program pentru a-l modifica: ștergerea uneia sau mai multor linii, înserarea uneia sau mai multor linii și modificarea uneia sau mai multor linii.

Ștergerea liniilor unui program se poate face utilizînd comanda DELETE prezentată mai sus.

O altă metodă este introducerea numărului de linie, urmat de caracterul CR.

Pentru inserarea de noi linii în program este necesar ca această eventualitate să fie prevăzută de la introducerea variantei inițiale, iar numerotarea liniilor să fie făcută cu rația 10, de exemplu.

Modificarea unei linii se poate face prin reintroducerea ei.

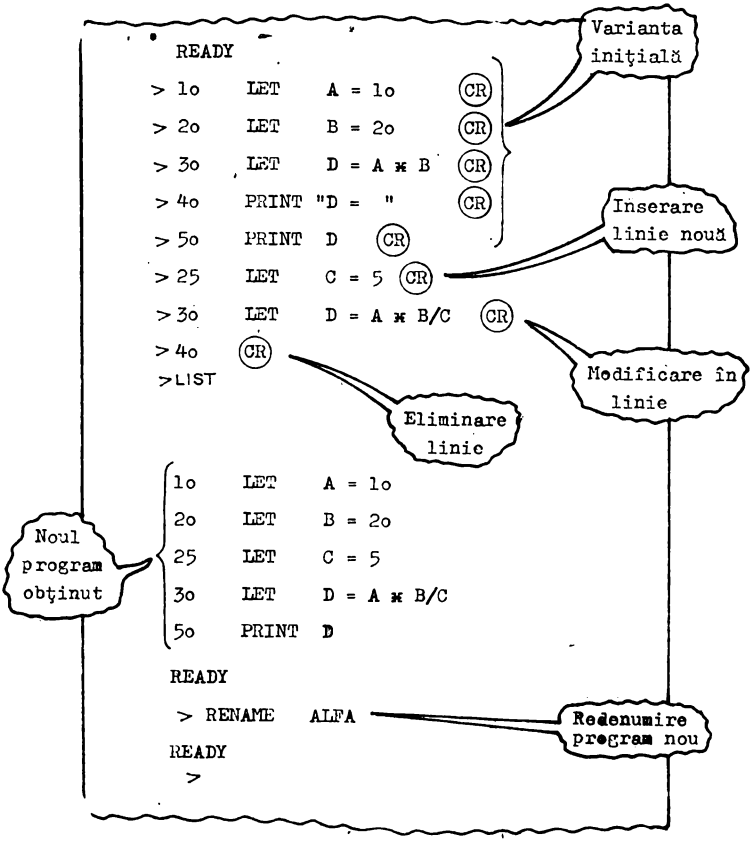


Fig. 3.1.2

EDIT Este posibilă modificarea unei linii de program fără a o reintroduce integral, dacă nu trebuie modificată în întregime. Pentru aceasta se va folosi **comanda** EDIT urmată de numărul liniei.

```
EDIT 50
50
```

Dacă veți tasta „L nr. linie" veți obține textul liniei menționate în comandă.

```
EDIT 50
50      PRINT „INTEGRARE PRIN METODA
                GAUSS"
50
```

Acționînd tasta de spațiu veți vizualiza caractere succesive.

Pentru a insera caractere noi veți vizualiza pînă la ultimul caracter care nu se schimbă:

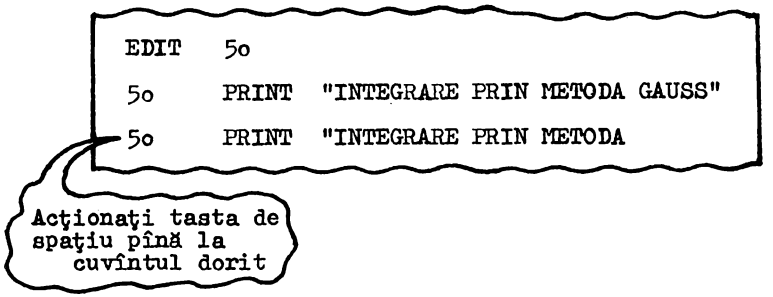


Fig. 3.1.3

Aici veti introduce I (inserare) urmat de caracterele dorite, de exemplu:

I — CUADRATURII GAUSS

ceea ce va provoca pe ecran următoarea imagine:

```
EDIT   50
50     PRINT „INTEGRARE PRIN METODA
       GAUSS"
50     PRINT „INTEGRARE PRIN METODA
       CUADRATURII GAUSS"
```

Dacă inserarea de caractere se face cu comanda I, suprimarea lor se face cu D. Modul de utilizare este același, cu diferența că se introduce numărul de caractere de suprimat, pornind numărătoarea de la caracterul curent, de exemplu:

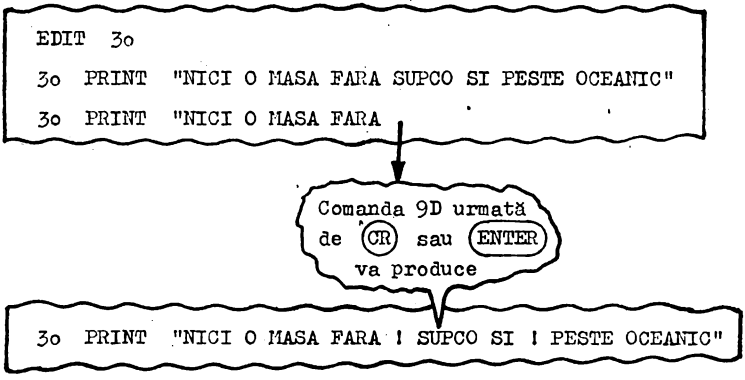


Fig. 3.1.4

Şi iată ce va produce o nouă afişare:

```
EDIT   30
30     PRINT „NICI O MASA FARA PESTE
       OCEANIC"
30
```

Anularea unei corecţii în curs se face cu comanda Q Programul de editare comportă şi alte ordine:

X — pentru adăugarea caracterelor la sfîrşitul unei linii;

H — pentru înlocuirea de caractere la sfîrşitul unei linii;

S — pentru căutarea unui anumit caracter: de exemplu SM

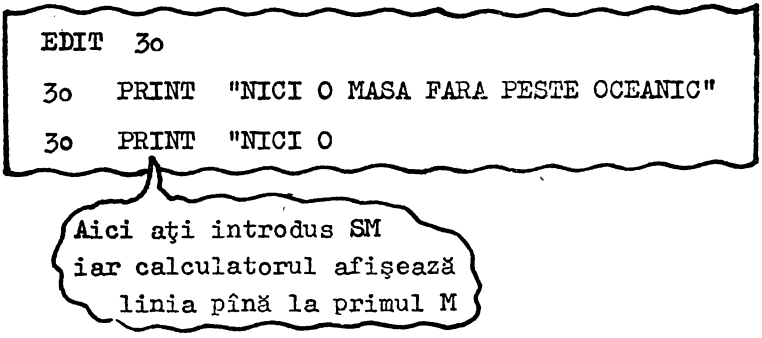


Fig. 3.1.4 bis

Puteți să indicați și despre a cîta apariție a caracterului specificat este vorba:

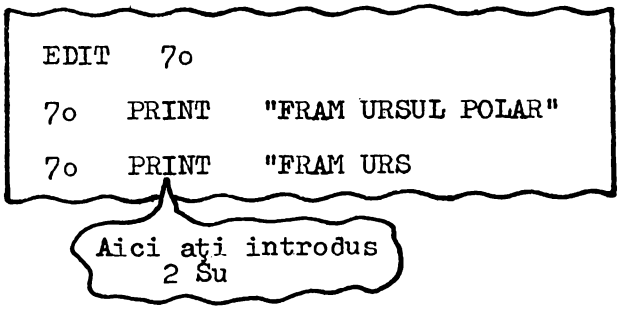


Fig. 3.1.5

Un K va suprima toate caracterele precedente în timp ce un C va determina modificarea caracterelor următoare.

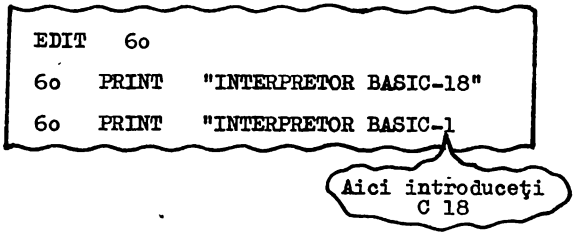


Fig. 3.1.6

După introducerea unui L veți obține:

```
60 PRINT „INTERPRETOR BASIC-118"
```

3.2 — COMENZI DE EXECUȚIE

În această categorie intră comenzile de lansare a programului, de pregătire a relansării și continuare a unui program întrerupt.

RUN Un program BASIC odată introdus nu poate fi lansat în execuție decît printr-o comandă RUN. Această comandă lansează în execuție programul existent în memoria centrală. În BASIC-18/118 (versiuni implementate pe

microcalculatoarele FELIX-M 18/M 118) comanda admite drept parametru numărul instrucțiunii cu care va începe execuția. În BASIC-AMS comanda admite drept parametru numele programului dorit a fi executat, care va fi, în prealabil, încărcat de pe suport extern în memoria centrală. Numele programului este un specificator de fișier în sens AMS.

CONT Această comandă permite reluarea unui program întrerupt printr-o instrucțiune STOP.

HELP Introducerea unei linii de program eronate determină interpretorul să afișeze un mesaj de eroare. De obicei acest mesaj este afișat în clar, dar uneori, așa cum este cazul interpretorului BASIC-AMS, se afișează numai un cod de eroare. În situația în care utilizatorul nu are la dispoziție o listă a codurilor de eroare, prin comanda HELP poate obține, din partea calculatorului, informații exacte cu privire la cauza erorii.

3.3 — COMENZI PENTRU LUCRUL CU PERIFERICELE

Așa cum am menționat în paginile anterioare, pentru ca un program să poată fi executat, este necesar ca el să se afle în memoria internă. În situația în care el se află înscris într-o memorie nevolatilă (memorie ROM sau memorie moartă) deconectarea de la rețea nu alterează cu nimic imaginea din memorie a programului. Dacă însă, execuția programului urmează să se facă într-o memorie volatilă, atunci este necesară încărcarea programului în această memorie după fiecare punere sub tensiune a ordinatorului. Încărcarea în memorie a programului presupune ca acesta să existe pe disc floppy sau casetă magnetică. Toate aceste operații de manipulare și transfer al programelor între memoria internă și cea externă sînt realizate prin intermediul unor comenzi ce vor fi prezentate în continuare.

PRINTERIS
READERIS
PUNCHIS Aceste instrucțiuni sînt folosite în BASIC-18 și BASIC-118 pentru a stabili perifericul de intrare pentru comanda LOAD (READERIS), perifericul de listare pentru instrucțiunea PRINT (PRINTERIS) și perifericul de ieșire pentru comanda SAVE (PUNCHIS).

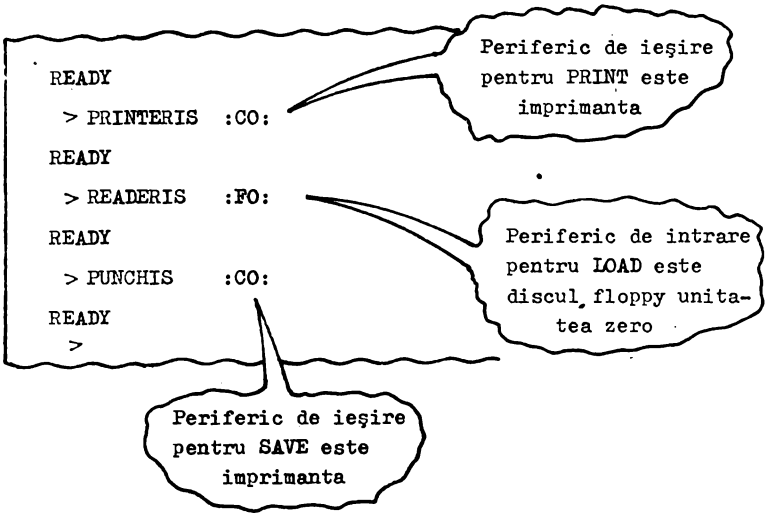


Fig. 3.3.1

SAVE Această comandă permite obţinerea unei copii a programului aflat în memorie, pe un anumit periferic. Sintaxa comenzii este diferită de la o variantă de implementare la alta. De exemplu, în BASIC-118 sintaxa este următoarea:

SAVE [N 1, N 2]

unde N 1 şi N 2 sînt numerele de linie ce delimitează secvenţa care urmează a fi copiată. Perifericul către care se va direcţiona operaţia de copiere va fi cel menţionat în ultima comandă PUNCHIS. Spre deosebire de BASIC-118, BASIC-AMS conţine în comandă şi numele perifericului. Astfel comanda:

SAVE DX : TEST.BAC

va determina înregistrarea pe disc floppy (DX:) a programului din memorie. Copia se va face într-un fişier cu numele TEST.BAC.

LOAD Această comandă, complementară lui SAVE, este utilizată pentru a transfera în memoria internă un program înregistrat pe un dispozitiv periferic extern. Fără această operaţie prealabilă nu este posibilă execuţia nici unui program.

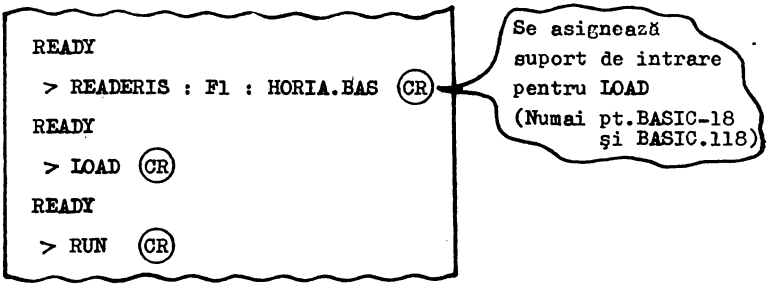


Fig. 3.3.2

Cu această secvență de comenzi se face lansarea în execuție a programului HORIA.BAS stocat pe discul floppy montat pe unitatea nr. 1 (F 1).

CSAVE Cu această comandă se realizează copierea pe casetă magnetică a programului aflat în memorie. Comanda trebuie să conțină numele sub care va fi înregistrat programul, sub forma unei singure litere, sau a unui cuvînt, din care nu se va memora decît prima literă.

CSAVE „X"

Înainte de a introduce comanda verificați dacă ați pornit casetofonul și dacă funcționează în regim de înregistrare.

CLOAD Este operația inversă lui CSAVE și se supune acelorași reguli ca și aceasta.

3.4 — EXERCIȚII

1. Scrieți comanda prin care să eliminați programul existent în memorie și punerea la zero a variabilelor (eventual ștergerea ecranului și poziționarea cursorului în linia 1 coloana 1).

2. În memoria internă se află încărcat programul cu numele SIMPSON. Scrieți comanda prin care să modificați numele acestui program în TRAPEZ.

3. Fiind dată următoarea secvență:

```
10 LET X = 2*PI↑ 2
20 LET Y = SIN(X)
30 LET Z = X/Y
```

să se scrie comenzile prin care să se modifice secvența astfel încît să devină:

```
10 LET X = (2*PI ↑ 2)/4
20 PRINT X
25 LET Y = SIN (X)
30 LET W = X/Y*2
```

4. Care dintre următoarele secvențe este corectă pentru încărcarea și execuția pe un FELIX M-118 a programului POISS.BAS, de pe discul floppy, unitatea 0.

a)
```
READY
> READERIS : C: (CR)
READY
> LOAD (CR)
READY
> RUN
```

b)
```
READY
> READERIS : C1 : (CR)
READY
> LOAD (CR)
READY
> RUN
```

c)
```
READY
> READERIS :F1: POISS.
                BAS (CR)
READY
> LOAD (CR)
READY
> RUN (CR)
```

d)
```
READY
> READERIS :F1: (CR)
READY
> LOAD (CR)
READY
> RUN (CR)
```

RĂSPUNSURILE EXERCIȚIILOR

1) NEW; 2) RENAME TRAPEZ;

3) EDIT 10
10

În continuare se acționează tasta spațiu pînă cînd linia va arăta astfel:

```
10 LET X =
```

aici vom introduce: I_(CR) ceea ce va produce:

```
10 LET X = (2*PI ↑ 2
```

apoi:

EDIT 10

10

vom acționa tasta spațiu pînă cînd linia va arăta astfel:

10 LET X = (2*PI ↑2

vom introduce: I_) (CR) ceea ce va produce:

10 LET X = (2*PI ↑ 2)

apoi după comanda:

EDIT 10

10

și H_/4, linia de program va arăta astfel:

10 LET X = (2*PI ↑ 2)/4

Pentru a modifica linia 20 vom introduce:

20 PRINT X

Pentru a insera linia 25 vom introduce:

25 LET Y = SIN (X)

Pentru a modifica linia 30 vom proceda astfel:

EDIT 30

30

aici vom introduce S _Z ceea ce va produce

30 LET

apoi C _W, H_ *2, L ceea ce va produce:

30 LET W = X/Y*2

4) c)

4. MODUL PROGRAM

Pînă acum am prezentat modul de lucru imediat și comenzile sau modul comandă al interpretorului BASIC standard. În realitate, modul program este cel care va fi utilizat pentru scrierea

programelor, modurile imediat și comandă fiind utile numai pentru listarea și respectiv punerea la punct a programelor, după ce acestea au fost scrise.

4.1 — REGULI DE SINTAXĂ

Pentru a sesiza bine diferența dintre modul program, modul imediat și modul comandă, introduceți următoarea secvență:

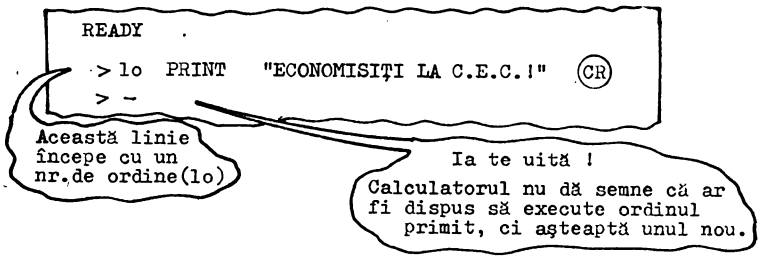


Fig. 4.1.1

Simplul fapt că linia de comandă este precedată de un număr de ordine, este pentru calculator o indicație că se va intra cu modul program.

Numerotarea liniilor În modul program se introduc secvențe de instrucțiuni. Fiecare instrucțiune trebuie să fie numerotată crescător. Sînt admise numai numere întregi, pozitive de la 1 la 32767. Liniile unui program pot fi numerotate în ordinea următoare: 1, 2, 3, 4, 5, 6, ... etc. În practică însă este preferabil să se adopte numerotarea cu pasul 10, aceasta pentru ca o eventuală inserare de instrucție să se poată face fără renumerotarea întregului program.

Pentru ca un program introdus să fie executat se folosește comanda RUN prezentată în capitolul anterior.

Punct și virgulă Adeseori ignorat în scrierea curentă, caracterul „;„ are în BASIC un rol foarte important. Pentru a demonstra acest lucru, să considerăm un exemplu:

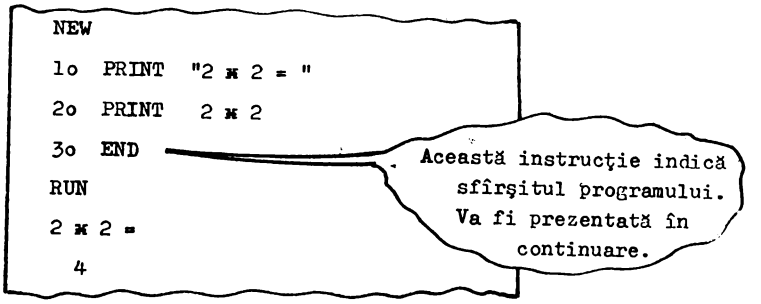


Fig. 4.1.2

Pentru a ameliora forma sub care se prezintă acest rezultat este preferabil ca operația 2*2 = 4 să apară pe o singură linie. Acest lucru se poate realiza utilizînd caracterul „;".

```
10 PRINT „2*2 = "; 2*2
20 END
RUN
2*2 = 4
READY
```

Punctul și virgula (;) semnifică continuarea afișării pe aceeași linie fără a efectua (CR).

Instrucția END În exemplele de mai sus am efectuat lansarea în execuție a cîtorva secvențe de program. Sfîrșitul oricărui program se marchează prin instrucția END care nu are decît acest rol. Este necesară utilizarea ei pentru ca interpretorul BASIC să poată determina sfîrșitul programului.

Instrucțiile LET și PRINT Aceste instrucții au fost prezentate cu prilejul modului imediat. În modul program semnificația lor este absolut identică.

Instrucția INPUT Nu toate datele pe care le utilizează un program, în cadrul prelucrărilor pe care le face, pot fi cunoscute în faza de introducere a sa. Apare uneori necesitatea de a furniza date programului în mod interactiv, în timpul execuției sale, prin dialog cu acesta. Pentru a realiza comunicarea cu autorul său, programul afișează diverse mesaje pe ecran, cu ajutorul instrucției PRINT și citește răspunsurile la aceste mesaje, cu ajutorul instrucției INPUT.

Următorul exemplu este edificator în ce privește modul de utilizare al instrucției INPUT.

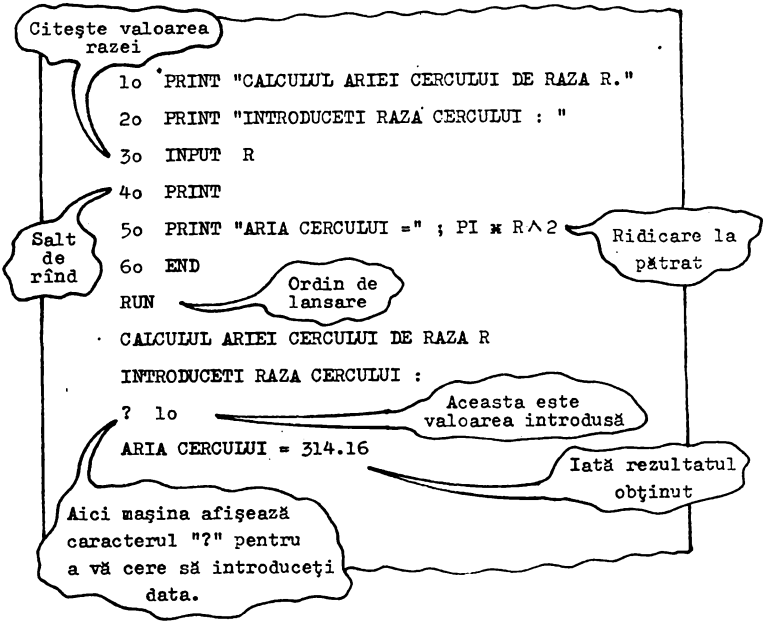


Fig. 4.1.3

Ceea ce se remarcă în acest program este, în primul rînd, instrucţia INPUT din linia 30. Este, după cum am arătat, o instrucţie de citire care se va traduce, după lansarea în execuţie a programului, în apariţia caracterului „?", după care se va aştepta introducerea, de la claviatură, a datei solicitate, urmate de (CR). Instrucţia din linia 40 nu are alt rol decît acela de a provoca avansul cursorului ecranului la o linie nouă.

De subliniat că, la fel ca în orice limbaj de programare de origine americană, în BASIC marca zecimală nu este virgula ci punctul Deci 314.16 trebuie citit 314,16

Exerciţii:

1) Rolul caracterului „;" este următorul: A — de a separa instrucţiile între ele; B — de a provoca un salt la rîndul următor; C — de a provoca o afişare fără avans la linie nouă.

2) Instrucţia: 10 PRINT „2 * 4 = "; 128/16 va provoca următoarea afişare: A — 8 = 128/16; B — 8 = 8; C — 2 * 4 = 128/16; D — 2 * 4 = 8.

3) Instrucția INPUT are rolul de a: A — afișa un șir de date; B — cere de la claviatură date noi.

4) Semnul „?„ afișat pe prima poziție din stînga unei linii, în timpul execuției unui program, semnifică faptul că mașina așteaptă: A — date noi; B — un ordin RUN.

Răspunsuri: 1) = C; 2) = D; 3) = B; 4) = A.

4.2. — ORGANIGRAME

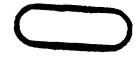
Se spune adeseori că un desen bine executat este mult mai lizibil și mai ușor descifrabil decît un text explicativ. Că așa stau lucrurile, o dovedesc foarte bine organigramele. Organigramele sau schemele logice, sau logigramele, nu sînt altceva decît niște diagrame care reflectă într-o manieră grafică, sugestivă, un proces de prelucrare care are loc în calculator. Un proces de prelucrare automată este îndeobște cunoscut sub numele de algoritm. Reprezentarea grafică a unui algoritm se realizează utilizînd simboluri convenționale a căror formă indică tipul acțiunilor ce vor fi executate și al căror conținut precizează semantica acestora. Simbolurile unei organigrame se clasifică în 6 grupe:

* *blocuri de calcul:*
aceste blocuri indică prelucrarea unor date și atribuirea unor valori

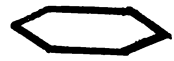
* *blocuri de decizie:*
aceste blocuri indică anumite condiții ce trebuie verificate și determină ordinea efectuării operațiilor în funcție de valoarea acestor condiții

* *blocuri de intrare-ieșire:*
aceste blocuri reprezintă citirea și scrierea datelor manipulate de program

* *blocuri de etichetă:*
aceste blocuri permit atribuirea unei etichete unui grup de prelucrări



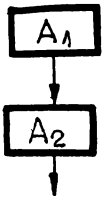
* *blocuri de stop și start:*
aceste blocuri permit marcarea punctelor de început și sfîrșit al programelor



* *blocuri de apel al subprogramelor sau procedurilor:*
indică transferul controlului către un anumit subprogram sau procedură specificată.

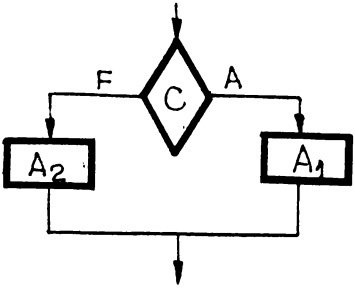
Aceste grupe de simboluri pot fi combinate în așa fel încît să facă posibilă descrierea tuturor categoriilor de prelucrări ce au loc în cadrul unui algoritm. Prelucrările sînt reprezentate, în principal, prin intermediul a 3 structuri fundamentale:

a) structuri secvențiale:



indică efectuarea operației A2 după operația A1

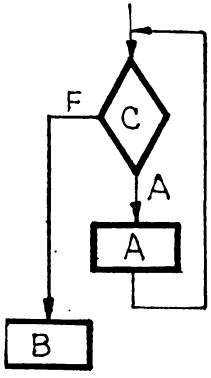
b) structuri alternative:



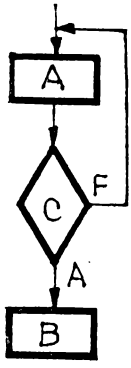
evaluează condiția C și execută acțiunea A1 dacă aceasta este adevărată, A 2 dacă este falsă

c) structuri repetitive sau de ciclare:

repetă execuția acțiunii A atît timp cît condiția C este adevărată, apoi execută acțiunea B



sau



repetă execuția acțiunii A pînă cînd condiția C devine adevărată, apoi execută acțiunea B

Vom prezenta în continuare cîteva exemple de organigrame. Ne propunem să scriem un program care să afișeze pe ecran, în poziția stînga sus, caracterele: „***". Dorim ca această afișare să aibă loc după aproximativ 1 secundă de la lansarea în execuție a programului, deci de la comanda RUN. Iată reprezentarea grafică a algoritmului:

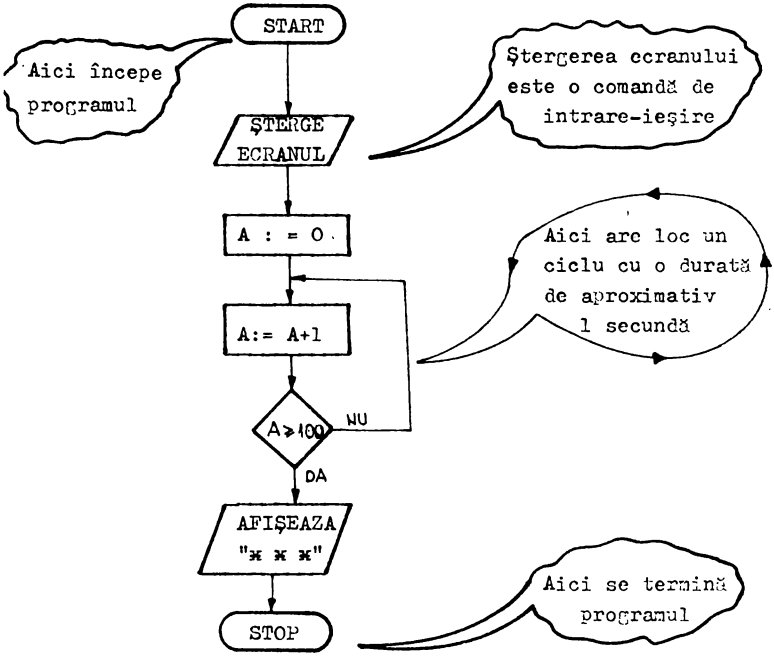


Fig. 4.2.1

De remarcat că temporizarea de aproximativ 1 secundă poate fi simulată printr-o contorizare de la 1 la 100. Calculatorul este rapid dar nu instantaneu și fiecare operație are o anumită durată. În BASIC se pot efectua aproximativ 100 de operații simple pe secundă.

În continuare vom complica încă puțin problema. Vom afișa, după aproximativ 1 s, caracterele „***" în stînga sus a ecranului, apoi după o nouă temporizare de aproximativ 1 s, le vom șterge și vom relua ciclul. Iată noua organigramă:

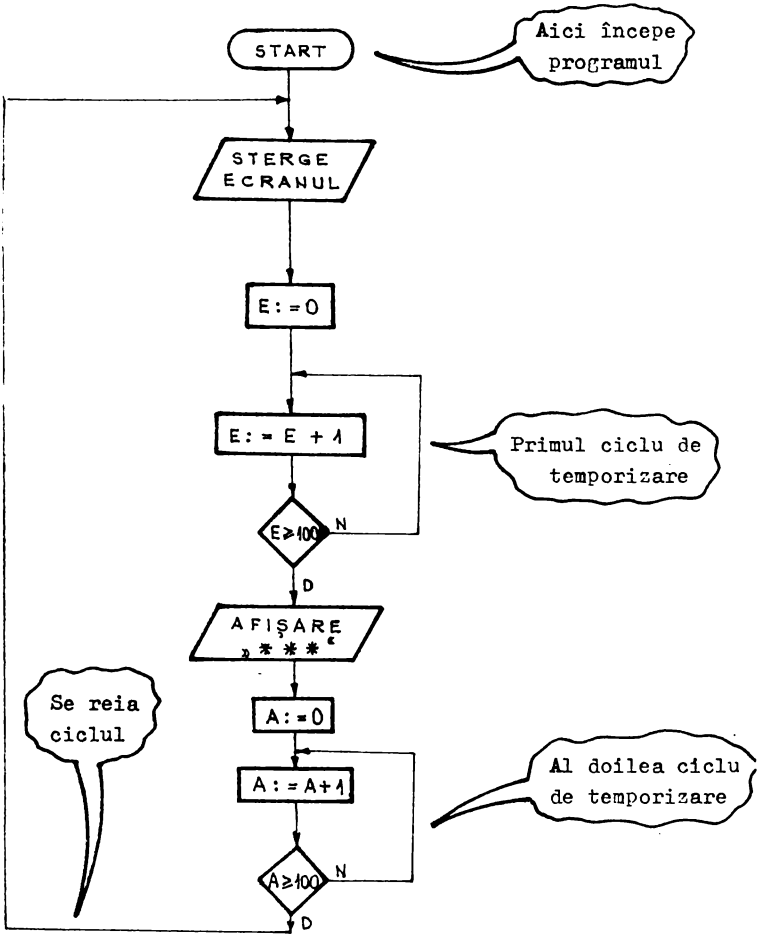


Fig. 4.2.2

După cum se poate vedea, schema de prelucrare a acestui program este liniară, ceea ce reprezintă un avantaj în ceea ce privește claritatea și lizibilitatea programului. Există însă și situații cînd criteriile lizibilității și clarității programului au mai puțină prioritate, în comparație cu cel al economiei de memorie utilizată. Acesta din urmă stă la baza rescrierii organigramei în forma următoare:

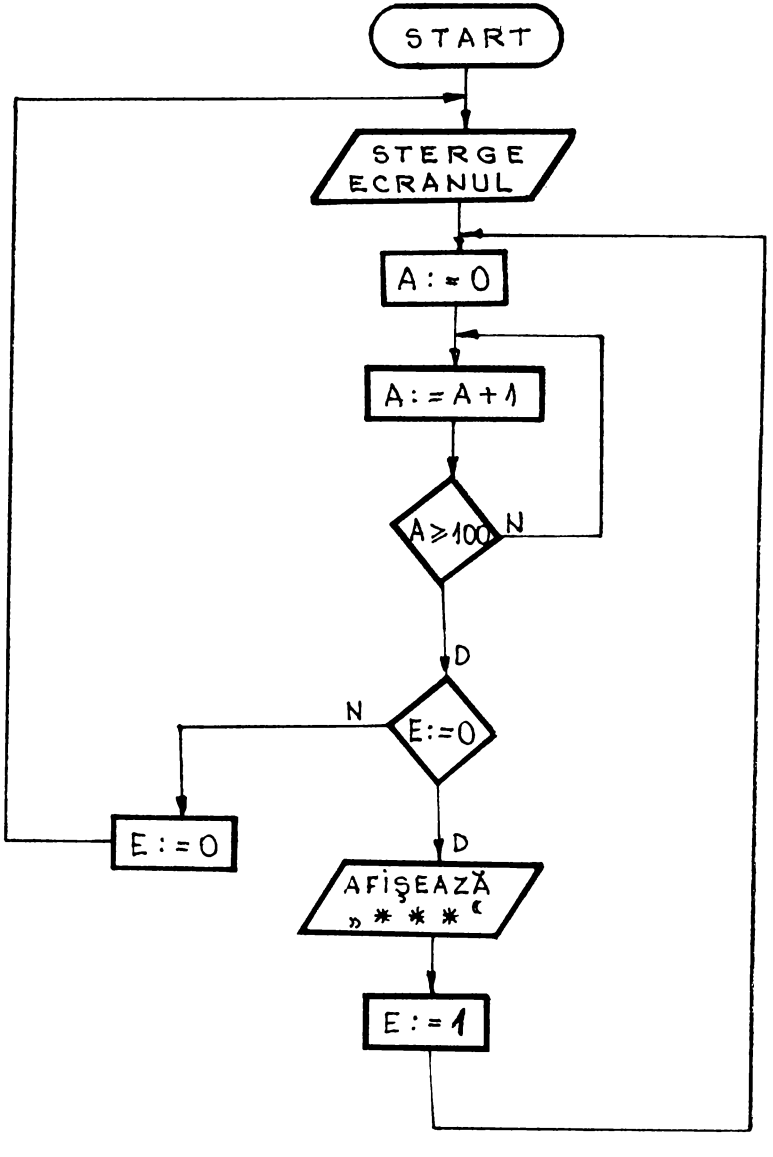


Fig. 4.2.3

Se observă că cele două secvențe de temporizare, din logigrama anterioară, au fost compactate într-una singură, lucru care duce la economie de memorie. Deși soluția prezentată pierde în claritate, ea este mai puțin costisitoare și deci mai eficientă. Eficiența și fiabilitatea unui program este de asemenea, direct proporțională cu modularitatea sa.

Modularitatea este un deziderat conform căruia programul este împărțit în secvențe autonome, numite module, care realizează funcții de sine stătătoare. Avantajul constă în faptul că la scrierea acestor module pot lucra simultan programatori diferiți, ceea ce permite o mai mare productivitate o muncii de programare. Prezentăm în continuare aceeași logigramă scrisă modular, cu un subprogram apelat în ciclu din programul principal. Acest subprogram este un program subordonat programului principal, către care se transferă controlul atunci cînd programul principal o cere. Alte detalii în legătură cu subprogramele și modul lor de apel se vor prezenta în paragraful 4.10.

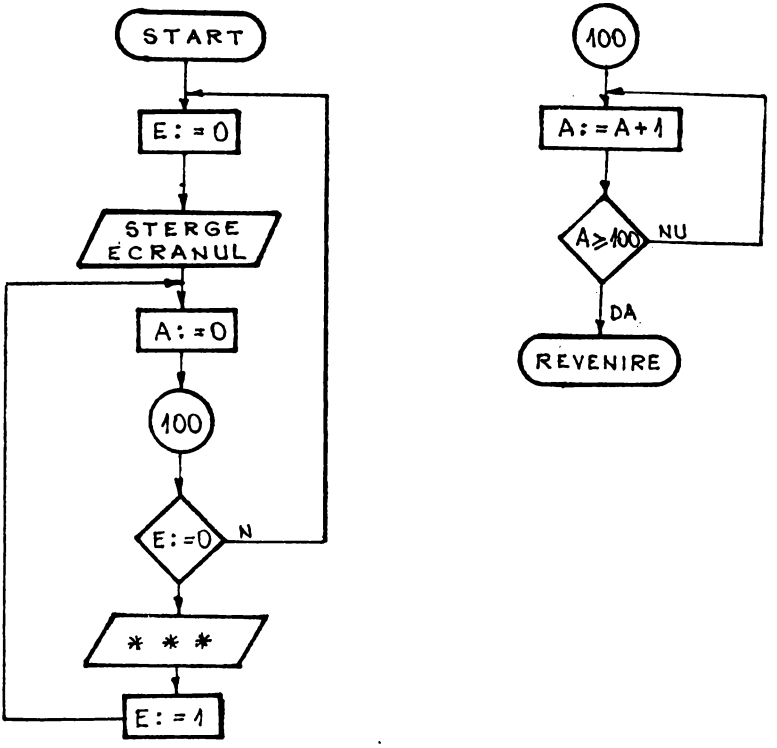


Fig. 4.2.4

Exerciții

1) Pentru a indica un salt condiționat se utilizează: A — un dreptunghi; B — un romb; C — un trapez.

2) Un paralelogram indică: A — o operație de intrare-ieșire; B — un bloc de start sau stop.

Răspunsuri: 1) = B; 2) = A.

4.3. — RAMIFICAȚII ȘI CICLURI

Problema pe care o vom aborda în acest capitol este transpunerea sub formă de instrucții ale limbajului BASIC, a algoritmilor descriși prin metoda organigramei. Vom examina, pentru început, instrucțiile de ramificație și ciclare GOTO și FOR și posibilitățile de stopare a unui program prin intervenție operator (BREAK), apoi de a continua execuția programului stopat (CONT).

4.3.1 — Instrucția de salt necondiționat GOTO

Există numeroase cazuri cînd este necesară reluarea repetată a execuției unui program, pentru diverse valori ale uneia sau mai multor variabile. Să considerăm următorul exemplu:

```
10 PRINT „SA SE CALCULEZE 2 LA PUTEREA"
20 INPUT K
30 PRINT „REZULTATUL ESTE"; 2↑K
40 GOTO 10
50 END
```

GOTO În linia 40 a fost introdusă instrucția GOTO care indică transferul controlului programului la linia 10. Se reia deci execuția programului începînd cu linia 10.

Salt necondiționat În felul acesta, ori de cîte ori execuția programului va ajunge în linia 40, se va efectua, în mod invariabil un așa-zis salt necondiționat care va face ca următoarea instrucție de executat să fie cea din linia 10. Iată care va fi efectul execuției acestui program:

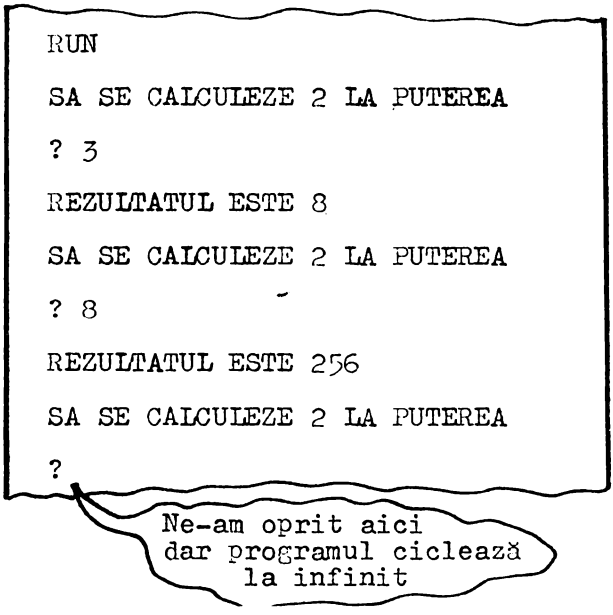


Fig. 4.3.1.1

GOTO indică un salt la o instrucție al cărei număr este precizat. GOTO se scrie într-un singur cuvînt. Există însă și interpretoare de BASIC care îl acceptă și în două cuvinte: GO TO

4.3.2. — Posibilități de ieșire dintr-un ciclu infinit

Se pune întrebarea cînd anume se termină programul din exemplul de mai sus. Evident niciodată. El va pretinde în mod obsedant să i se ofere diferite valori pentru a-și putea face treaba, dar nu va mai putea fi folosit și pentru alte prelucrări. De aceea s-au prevăzut posibilități de stopare a acestui ciclu infernal. În ceea ce privește comenzile care realizează această funcție, ele diferă de la o mașină la alta.

BREAK
PAUSE
ARRET

Acestea sînt trei dintre cele mai frecvente taste funcționale a căror acționare determină suspendarea execuției programului în curs. La alte calculatoare acest lucru este realizat prin apăsarea simultană a două taste CRTL și C.

CRTL-C Această acțiune întrerupe programul în curs și readuce sistemul în modul comandă. La unele calculatoare această comandă provoacă și afișarea numărului de linie unde programul a fost stopat. În acest fel se poate bloca un program a cărui execuție se derulează „ad vitam aeternum".

4.3.3 — Continuați, vă rog

CONT După stoparea unui program prin BREAK nu sîntem obligați să reluăm de la început activitatea sa, prin RUN. Dacă vă răzgîndiți și doriți să reluați și doriți să reluați programul din punctul întrerupt, folosiți comanda CONT.

> Reluarea unui program stopat prin BREAK se face, din punctul de unde a fost întrerupt, prin CONT

4.3.4 — Instrucția FOR...TO

Să revenim asupra exemplului prezentat în 4.2, punctul b, pentru a-l analiza cu mai multă atenție. Ciclul utilizat acolo pentru provoca o temporizare (întîrziere) nu mai este un ciclu infinit, ca cel din exemplul precedent, ci este un ciclu cu un număr cunoscut de pași. Iată cum vom codifica această secvență de temporizare.

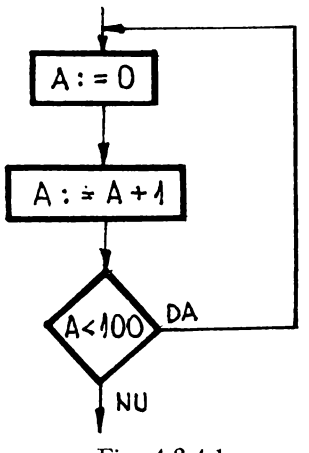


Fig. 4.3.4.1

```
10 FOR A = 1 TO 100
20 NEXT A
30
```

FOR...TO Linia 10 indică li mitele numărătorului A cu instrucția „FOR A = 1 TO 100" care înseamnă „EXECUTA PENTRU A CU VALORI DE LA 1 LA 100". În mod automat, mașina va începe execuția secvenței atribuindu-i lui A limita inferioară (1) și va incrementa această valoare, cu 1, pînă cînd va deveni egală cu 100.

NEXT Incrementarea variabilei A, despre care vorbeam mai sus, este indicată prin instrucția „NEXT A", adică, „URMĂTOAREA VALOARE A LUI A". Calculatorul va determina noua valoare a lui A și va reveni la linia 10. Ciclul se va derula pînă cînd A va atinge valoarea 100 inclusiv. În acest moment va avea loc așa-zisa „ieșire din ciclu" iar controlul va fi transferat la instrucția 30.

Ciclurile FOR...TO pot utiliza un pas mai mare decît 1. Acest pas poate fi 2, 3, 4 etc. Dacă pasul nu este precizat, valoarea sa implicită este 1.

4.3.5 — Cicluri FOR...TO cu pas diferit de 1

Ciclurile FOR...TO pot utiliza un pas diferit de 1. Reamintim că pasul unui ciclu FOR...TO este valoarea cu care se incrementează variabila de control, la sfîrșitul fiecărui ciclu.

STEP Pentru a programa un ciclu cu pas diferit de 1 este suficientă utilizarea ordinului complet FOR...TO.. ...STEP, unde STEP înseamnă „pas". Să considerăm următorul exemplu:

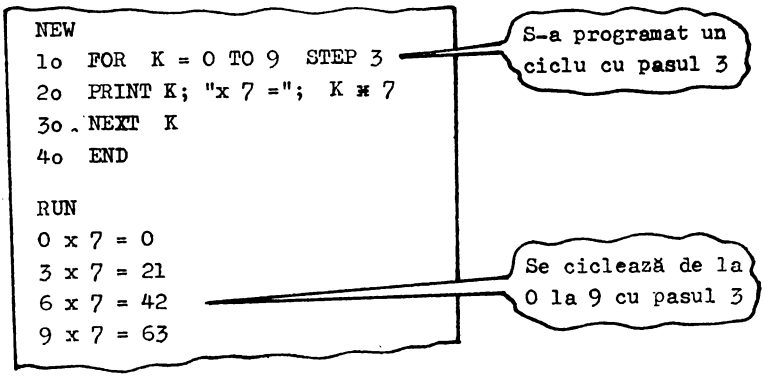


Fig. 4.3.5.1

În următorul exemplu vom prezenta un ciclu cu pas negativ:

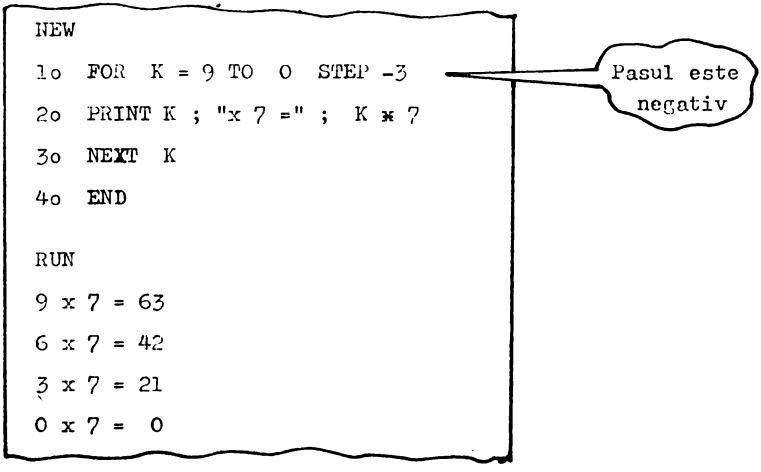


Fig. 4.3.5.2

Aveți grijă să nu vă alterați programele din memorie. Dacă nu veți începe secvența cu NEW care șterge vechile programe, noile programe riscă că păstreze reminiscențe străine. De asemenea, absența lui END poate determina ramificarea către aceste programe ale căror resturi subzistă încă în memorie. Deci, prudență.

Exerciții

1) Iată un program care conține 5 erori. Detectați-le:

```
23 INPUT „C"
38 PRINT, C, C * C
43 LET C = 5
51 PRINT: C IMPARTIT LA 2 FAC
680 PRINT C/2
800 END
870 GOTO 23
```

2) GOTO este o instrucție de: a) salt; b) atribuire.

3) În următoarea instrucție: 20 FOR K = 3 TO 11 prima valoare atribuită lui K este: a) 0; b) 3; c) 11

4) Această instrucție este corectă: FOR K = 0 TO K = 9?

5) Ce va produce acest program:

```
10 PRINT „CITE STELE DORITI?"
20 INPUT N
30 FOR K = 1 TO 2 * N
40 PRINT „*";
50 NEXT K
60 GOTO 20
70 END
```

a)

```
CITE STELE DORITI?
? 3
* * * * * *
? 2
* * * *
?
   etc.
```

b)

```
CITE STELE DORITI ?? 3
* *
* *
* *
CITE STELE DORITI ?? 2
*
*
```

6) Aceste două instrucții sînt echivalente: 10 FOR R = 3 TO 21; 20 FOR R = 3 TO 21 STEP 1 ?

7) Putem stopa un program aflat în execuție, acționînd asupra unei taste?

4.4 — INSTRUCȚII DE SALT CONDIȚIONAT

Instrucția condiționată IF...THEN se folosește împreună cu instrucția de salt GOTO pentru a realiza ramificații în funcție de valoarea unei condiții specifice.

IF...THEN Utilizarea instrucției GOTO provoacă execuția unui salt necondiționat la adresa indicată. Instrucția IF...THEN determină un salt condiționat, așa cum reiese din următorul exemplu:

```
1o  PRINT  "TABLA INMULTIRII CU 3"
2o  A = 0
3o  PRINT  "3 ORI "; A; "=" ; 3 * A
4o  A = A + 1
5o  IF  A < = 9  THEN 3o
6o  PRINT  "PROGRAM TERMINAT".
7o  END
```

Bizar

Fig. 4.4.1

Să examinăm linia 40 care propune o egalitate cel puțin ciudată, din punct de vedere matematic. În programare această expresie are sens și semnifică faptul că variabilei A i se atribuie o valoare egală cu vechea sa valoare plus 1. Astfel, dacă inițial variabila A avea valoarea 0, după execuția instrucției valoarea ei va fi 1.

Linia 50 este însă aceea care ne interesează în mai mare măsură. Ea spune că: dacă (IF) A este mai mic sau egal cu 9, atunci (THEN) controlul este transferat liniei 30. Asta înseamnă că, în situația în care valoarea lui A depășește cifra 9, controlul este dat liniei 60. Este deci vorba de un salt condiționat de valoarea variabilei A. Iată ce se va întîmpla dacă vom lansa programul în execuție.

```
RUN
TABLA INMULTIRII CU 3
3 ORI 0 = 0
3 ORI 1 = 3
3 ORI 2 = 6
3 ORI 3 = 9
3 ORI 4 = 12
3 ORI 5 = 15
3 ORI 6 = 18
3 ORI 7 = 21
3 ORI 8 = 24
3 ORI 9 = 27
PROGRAM TERMINAT
```

Dacă nu știați pînă acum tabla înmulțirii cu 3 o veți ști poate, la sfîrșitul acestui capitol

Fig. 4.4.2

Următorul exemplu de utilizare al instrucției IF...THEN ne va conduce la o situație extrem de frecventă și anume la situația prelucrării parolei. Să ne imaginăm următoarea situație: avem la dispoziție un program cu ajutorul căruia putem exploata un volum însemnat de informații, dar accesul la aceste informații este permis numai acelor utilizatori care cunosc parole de acces. Secvența de program prezentată în continuare, oferă o modalitate de verificare a parolei de acces.

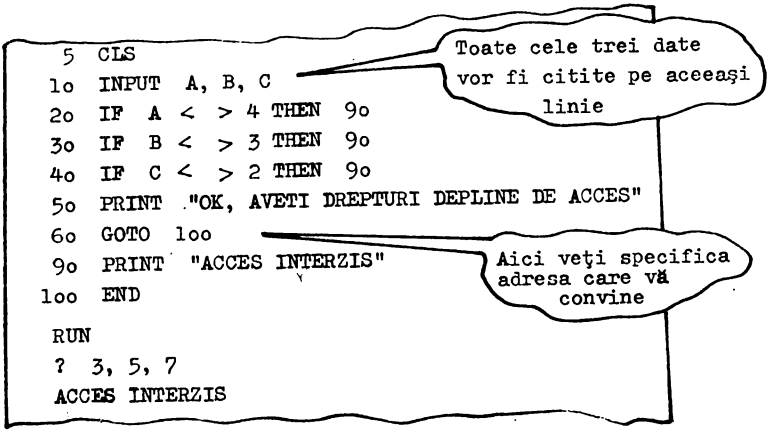


Fig. 4.4.3

În exemplul de mai sus, în linia 10 se face citirea a trei valori: A, B şi C. Ori de cîte ori vom dori să citim sau să scriem mai multe valori cu un singur INPUT sau cu un singur PRINT, vom folosi ca delimitator între valori caracterul virgulă

Exerciţii

1) IF...THEN determină un salt: A — condiţionat; B — necondiţionat.

2) Condiţia este specificată după: A — cuvîntul THEN; B — cuvîntul IF.

3) Care este rolul instrucţiei CLS: A — ştergerea unei linii; B — ştergerea întregului ecran.

4) O virgulă plasată într-o instrucţie INPUT determină: A — salt de linie nouă după citirea fiecărei variabile; B — separarea variabilelor.

5) O virgulă plasată într-o instrucţie PRINT determină: A — scrierea variabilelor una după alta, fără spaţiu între ele; B — scrierea variabilelor una după alta, separate prin spaţii; C — scrierea fiecărei variabile pe o linie nouă.

6) Examinaţi următoarele programe:

```
10 LET K = 1
20 IF K > 4 THEN 60
```

```
30 PRINT K; K * K; K * K * K
40 LET K = K + 1
50 GOTO 20
60 END
```

```
10 FOR K = 1 TO 4
20 PRINT K; K * K; K * K * K
30 NEXT K
40 END
```

Furnizează rezultate identice? A — Da; B — Nu.

Răspunsuri: 1) A; 2) B; 3) B; 4) B; 5) B; 6) A.

4.5 — DECLARAREA ŞI CITIREA DATELOR

Aşa cum s-a arătat în capitolele precedente, un program este o secvenţă de instrucţii capabilă să realizeze o serie de operaţii de prelucrare asupra unor date. Acest capitol face o prezentare a instrucţiilor de declarare şi citire a datelor. De asemenea, se va sublinia importanţa introducerii de comentarii în structura programului.

DATA Putem introduce date în cadrul unui program declarîndu-le cu ajutorul instrucţiei DAŢA.

De exemplu instrucţia:

```
10 DATA 1, 3, 5, 7
```

introduce o listă de valori care sînt primele patru cifre impare.

De remarcat prezenţa virgulei care serveşte, în acest caz, drept separator

READ Vă întrebaţi, probabil, cum vă putea dispune programul de datele introduse prin DATA pentru a le manipula şi prelucra. Foarte simplu! Citindu-le cu ajutorul instrucţiei READ.

Aceste două instrucţii, DATA şi READ, sînt complementare. Una fără cealaltă nu are sens.

Prezentăm în continuare un exemplu de utilizare al instrucţiilor DATA şi READ, care efectuează afişarea unor date interne:

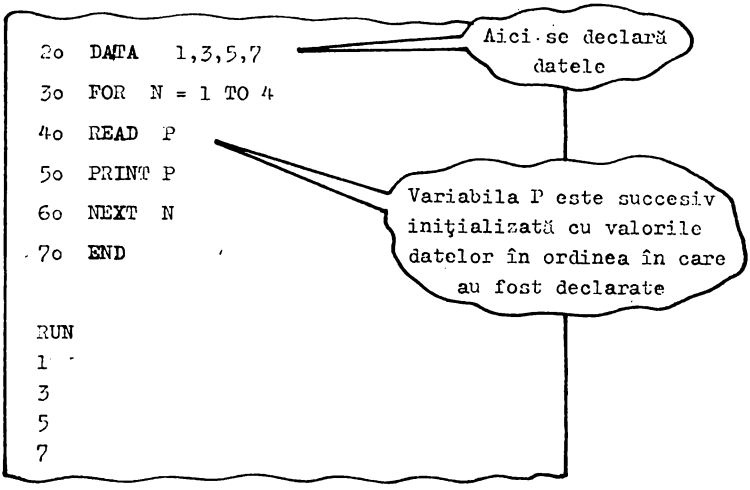


Fig. 4.5.1

În linia 20 se declară patru constante numerice care sînt în ordine, primele cifre impare. În linia 30 se deschide un ciclu cu patru pași. În linia 40 se atribuie unei variabile, în mod arbitrar denumite P, prima constantă declarată. Această constantă este afișată în linia 50. Linia 60 face o trecere la următorul pas și ciclul se reia începînd cu linia 40. După ce toate cele patru constante au fost afișate programul se termină.

Utilizarea instrucțiilor DATA și READ comportă următoarele trei reguli:

1. La întîlnirea instrucției DATA ordinatorul nu face nimic altceva decît să memoreze valorile specificate. Numai după execuția instrucției READ, va prelua pe rînd, una cîte una, aceste valori.

2. În consecință, instrucția DATA poate apare înainte sau după READ, efectul ei fiind același.

3. Instrucția de citire respectă aceeași ordine a instrucției de introducere, datele fiind preluate de la stînga la dreapta.

Dacă lista de date manipulate de un program este mai lungă, aceasta poate continua pe două sau mai multe linii. Programul le va citi ca și cum s-ar succeda pe o singură linie. Iată un exemplu:

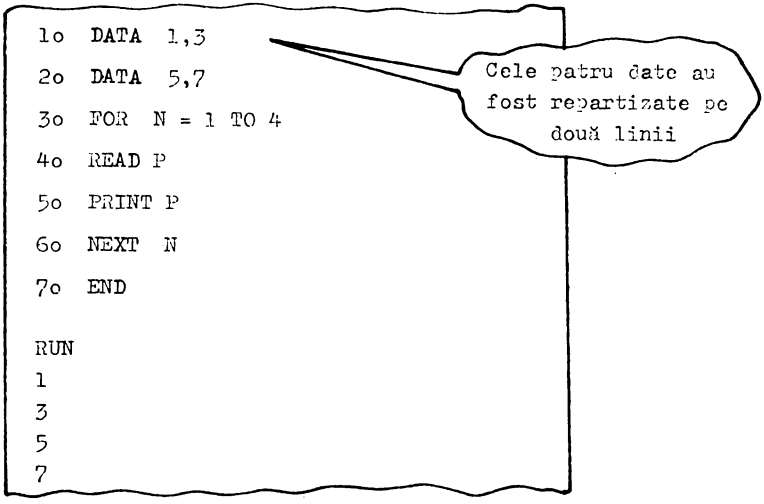


Fig. 4.5.2

Odată cu lansarea prin ordinul RUN a unui program care conține instrucții DATA și READ, ordinatorul va inițializa un indicator cu ajutorul căruia va determina care este următoarea valoare din lista de valori generată prin DATA, pe care o va preleva prin READ.

Dacă doriți să puneți la zero acest indicator, pentru a relua explorarea listei de valori de la început, fără a stopa și relansa programul, puteți face acest lucru folosind comanda RESTORE.

RESTORE Să presupunem că, în cadrul unei aplicații de gestiune a stocurilor, dorim să obținem două liste, utilizînd același set de date. Vom considera exemplu din fig.4.5.3;

REM Este indicat ca programele să conțină comentarii explicative care să le facă mai ușor lizibile și mai clare. Aceste linii de comentariu, cu caracter pur explicativ, sînt introduse de instrucția REM.

> O instrucție REM este o notă pe care programatorul o plasează în interiorul programului pentru a furniza explicații unui eventual cititor.

REM este urmată de un text care poate conține orice caractere De exemplu:

```
40 REM * * TRANSFORMATA FOURRIER * *
```

```
10  PRINT  "NR CURENT" , "CANTITATE" , "PRET"
20  FOR  N = 1 TO 4
30  READ  R, Q, P
40  PRINT R, Q, P
50  NEXT  N
60  RESTORE
70  PRINT
80  PRINT  "NR CRT" , "CANT" , "PRET" , "VALOARE"
90  FOR  N = 1 TO 4
100 READ R, Q, P
110 PRINT R, Q, P,  Q * P
120 NEXT  N
130 END
140 DATA 1,63,12,2,30,6
150 DATA 3,42,20,4,10,35

RUN
NR.CURENT     CANTITATE      PRET
    1 .          63           12
    2            30            6
    3            42           20
    4            10           35
NR.CRT        CANT           PRET        VALOARE
    1            63           12           756
    2            30            6           180
    3            42           20           840
    4            10           35           350
```

Se reia lista de la început

Acestea sînt datele prelucrate

Primul tabel

Al doilea tabel

Fig. 4.5.3

În BASIC extins instrucția REM poate fi înlocuită prin utilizarea ghilimelelor sau apostrofului. Astfel, următoarele linii de program sînt echivalente:

```
10 "SECVENTA DE TEST"
10 'SECVENTA DE TEST'
10 REM SECVENTA DE TEST
```

Exerciții :

1) Fie următoarele două secvențe:

```
10 DATA 1, 2, 3, 4, 5
20 DATA 6, 7, 8, 9, 10
```

```
10 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Execuția lor conduce la obținerea aceluiași rezultat? A — da; B — nu.

2) Putem plasa o instrucție de declarare de date în orice loc în program? A — da; B — da, dar nu într-o buclă FOR...NEXT; C — da, dar nu după END; D — da.

Fie următorul program:

```
10 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
20 READ A, B, C
30 READ X, Y, Z
40 RESTORE
50 READ K, L
60 PRINT A; B; C; K; L; X; Y; Z
70 END
```

3) Care sînt valorile ce vor fi citite în linia 20? A — 1, 2, 3; B — 7, 8, 9.

4) Care sînt valorile ce vor fi citite în linia 30? A — 1, 2, 3; B — 4, 5, 6; C — 7, 8, 9.

5) Care sînt valorile ce vor fi citite în linia 50? A — 1, 2, 3; B — 7, 8; C — 1, 2.

6) Care secvență va fi afișată? A — 1, 2, 3, 1, 2, 4, 5, 6; B — 1, 2, 3, 4, 5, 6, 7, 8; C — 1, 2, 3, 7, 8, 9, 1, 2.

7) Ce se va afișa cu instrucția următoare:

40 REM PRINT 4 * 3
50 END

A — PRINT 4 * 3
B — 12
C — nimic

Răspunsuri:

1) A; 2) D; 3) A; 4) B; 5) C; 6) A; 7) C.

4.6 — OPERAȚII PE ȘIRURI DE CARACTERE

În afară de valorile numerice pe care le prelucrează, interpretorul BASIC poate manipula șiruri de caractere. Aceste șiruri de caractere constituie obiectul prezentului capitol. Vor fi prezentate aici instrucțiile LEN, LEFT$, MID$, RIGHT$, INSTR.

4.6.1 — Definiția unui șir de caractere

ȘIR DE CARACTERE Un șir de caractere, sau pe scurt un șir, nu este altceva decît o secvență de simboluri alfanumerice. Orice ordinator poate prelucra atît numere, cît și cuvinte. Dar noțiunea de cuvînt a limbajului nu are nici o semnificație pentru mașină, de aceea s-a introdus accepția de șir de caractere.

Iată cîteva șiruri de caractere perfect corecte:

„CONSUMAȚI APA MINERALĂ HEBE!"

„RAZA UNUI NUCLEU ATOMIC ESTE DE 10^{-13} CENTIMETRI"

„XOO7 = 375 F 50"

sau chiar:

„! ! ! = ? ? AH, AH = = "

sau și mai și:

„ „ (șir de caractere vid).

> Un șir de caractere este delimitat de ghilimele. Acesta este unicul simbol pe care un șir de caractere nu-l poate conține.

CONSTANTA DE TIP ȘIR Un șir de caractere este întotdeauna o constantă.

VARIABILA DE TIP ȘIR Identificatorul unui șir de caractere este întotdeauna o variabilă.

Să ilustrăm cele de mai sus printr-un exemplu:

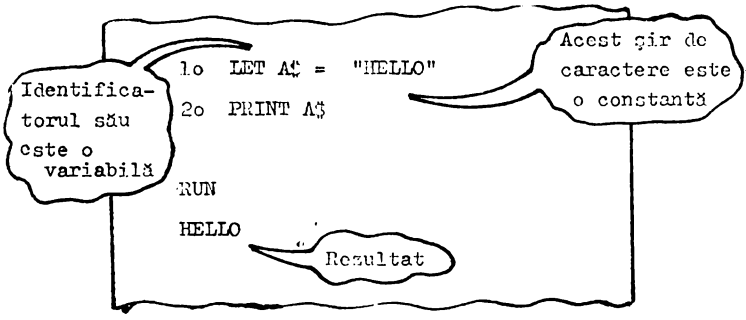


Fig. 4.6.1.1

O variabilă de tip șir (identificatorul unui șir) este reprezentată la o literă a alfabetului urmată de simbolul „dolar" ($)

La ce ar putea servi un șir vid? De exemplu:

```
10 PRINT   "SALT LA LINIA URMATOARE"
20 LET  Z$ = "  "
30 PRINT  Z$
40 PRINT "SFIRSIT"
50 END

RUN
SALT LA LINIA URMATOARE

SFIRSIT
```

Șir vid

Rezultatul

Fig. 4.6.1.2

4.6.2 — Adunarea șirurilor de caractere

Adunarea șirurilor înseamnă, de fapt, concatenarea sau juxtapunerea lor.

Iată un exemplu:

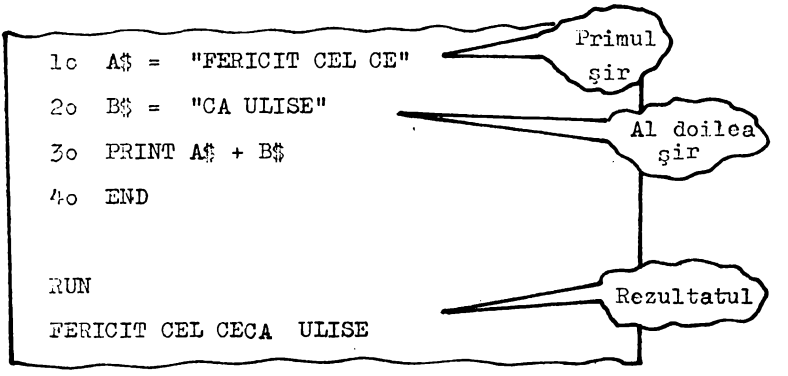


Fig. 4.6.2.1

Cele două şiruri de caractere au fost astfel concatenate încît nu sînt separate de un spaţiu.

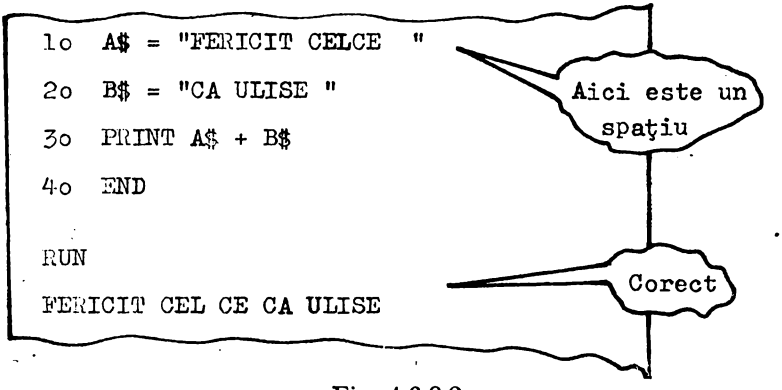


Fig. 4.6.2.2

4.6.3 — Introducerea şirurilor

Aşa cum putem cere introducerea numerelor, în mod interactiv, în timpul execuţiei unui program, tot astfel putem proceda şi atunci cînd este vorba despre şiruri de caractere.

```
10 PRINT „CUM VA NUMITI?"
20 INPUT N $
30 PRINT „NUMELE DUMNEAVOASTRA ESTE"; N$
40 END
RUN
CUM VA NUMITI?
? HORIA
NUMELE DUMNEAVOASTRĂ ESTE HORIA
```

4.6.4 — Lungimea unui şir

LEN Cum se poate determina prin program lungimea unui şir de caractere? Cu ajutorul instrucţiei LEN.

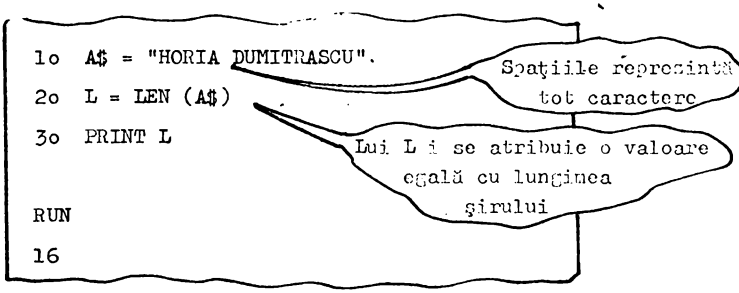


Fig. 4.6.4.1

4.6.5 — Extragerea subşirurilor

Să presupunem acum că, fiind dat şirul de caractere A$ = „FERICIT CEL CE CA ULISE", dorim să reaşezăm caracterele pe trei rînduri.

**LEFT$** Selecţia primelor 12 caractere se va face cu instrucţia LEFT$ (A$, 12).

**MID$** Partea centrală va fi selecţionată prin ordinul MID$ (A$, i, j) unde i reprezintă punctul de plecare în cadrul şirului, adică 13, iar j numărul de caractere de extras, adică 6.

**RIGHT$** În fine, partea rămasă o vom extrage cu ajutorul instrucţiei RIGHT$ (A$, 5). Iată programul:

```
10 A$ — „FERICIT CEL CE CA ULISE"
20 B$ = LEFT$ (A$, 12)
30 C$ = MID$ (A$, 13, 6)
40 D$ = RIGHT$ (A$ 5)
50 PRINT B$
60 PRINT C$
70 PRINT D$
80 END
RUN
FERICIT CEL
CE CA
ULISE
```

4.6.6 — Căutarea unui caracter sau subșir în cadrul unui șir

Să considerăm exemplul prezentat la punctul anterior și să presupunem că dorim să cunoaștem poziția primului T în cadrul șirului A$. Pentru aceasta vom scrie:

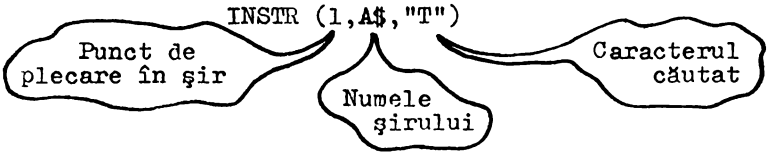


Fig. 4.6.6.1

Următorul exemplu folosește instrucțiile prezentate:

```
10 A$ = „FERICIT CEL CE CA ULISE"
20 B = LEN (A$)
30 PRINT B
40 C = INSTR (1, A$, „T")
50 PRINT C
60 D = INSTR (3, A$, „E")
70 PRINT D
80 END
RUN
23
7
8
```

Exerciții

1) Următoarele șiruri de caractere sînt valide? A — „BUNA ZIUA"; B — „? OK!"; C — „SCRIETI: „1+1=2".

4.7 — REALIZAREA PAGINĂRILOR

Acest capitol constituie o introducere în tehnica formatării datelor pe ecranul terminalului, cu ajutorul ordinului PRINT TAB. Vom vecdea cum putem trasa curbe cu ajutorul acestui ordin. Vom examina, de asemenea, comanda PRINT USING și cîteva dintre principalele sale moduri de utilizare.

4.7.1 — Organizarea ecranului

În modul cel mai general, ecranul display-ului are un număr de 24 de linii, fiecare a cîte 80 de coloane, deci, în total, 1 920 de caractere.

Pentru a afla numărul caracterelor afișabile pe ecranul pe care lucrați, este suficient să introduceți, de la tastatură, diferite caractere, pînă la umplerea totală a unei linii, apoi să le numărați. Veți constata că o dată ce o linie se umple, cursorul ecranului se va poziționa automat la începutul liniei următoare.

Pentru a număra liniile, este suficient să introduceți, la începutul fiecărei linii, un caracter oarecare urmat de un CARRIAGE-RETURN sau un ENTER. După ce veți ajunge la ultima linie, introducerea liniei următoare va provoca decalajul global al paginii de text în sus.

Deci, considerînd ecranul ca avînd formatul standard de 80×24, putem spune că o punere în pagină, sau o formatare, înseamnă a afișa o informație ținînd cont de poziția sa pe ecran, adică de numărul liniei și al coloanei.

PUNCTUL ȘI VIRGULA O formă elementară de formatare a datelor afișate constă în utilizarea caracterului „;"

Comparați următoarele două secvențe:

```
1o  PRINT  "A"
2o  GOTO   1o
```

Primul program

```
1o  PRINT "A" ;
2o  GOTO  1o
```

Al doilea program

Fig. 4.7.7.1

În faza de execuție, primul program va afișa caractere A pe o singură coloană, în stînga ecranului. Această coloană va umple cele 24 de linii, după care va defila în continuare, pînă cînd veți apăsa BREAK sau CTRL-C.

Cel de-al doilea program va umple întregul ecran cu caractere A. Singura diferență între cele două programe este apariția caracterului „;" la sfîrșitul liniei 10, în cel de-al doilea program. Acest caracter produce continuarea afișării, fără a sări la o linie nouă. Numai atunci cînd linia curentă este plină, se trece automat la linia următoare.

Iată un alt exemplu:

```
10 CLS
20 PRINT „ASTAZI SINTEM IN"
30 INPUT A$
40 PRINT „BUCURESTI,"
50 PRINT A$
60 END
RUN
ASTAZI SINTEM IN
? 1 IANUARIE ANUL 2004
BUCURESTI
1 IANUARIE ANUL 2004
```

Urm ătoarea versiune diferă de prima numai prin două caractere „;"

```
10 CLS
20 PRINT „ASTAZI SINTEM IN";
30 INPUT A$
40 PRINT „BUCURESTI,"
50 PRINT A$
60 END
RUN
ASTAZI SINTEM IN? 1 IANUARIE ANUL 2004
BUCURESTI, 1 IANUARIE ANUL 2004
```

Utilizînd caracterul „:" drept operator de concatenare, putem grupa două sau mai multe instrucții pe aceeași linie. Același program de mai sus poate fi scris în maniera următoare:

```
10 CLS
20 PRINT „ASTAZI SINTEM IN"; : INPUT A$
30 PRINT „BUCURESTI,"; A$
80 END
RUN
ASTAZI SINTEM IN? 1 IANUARIE ANUL 2004
```

Dacă vom introduce o nouă linie cu numărul 25 care să conțină o instrucție CLS, vom obține:

Fig. 4.7.1.2

Atenție: nu toate interpretoarele BASIC acceptă o astfel de compactare

4.7.2 — Ordinul PRINT TAB

Să trecem la un alt tip de program:

```
1o  PRINT  "X          X/X          X-X "     ,
2o  PRINT  ". . . . . . . . . . . . . . . . ."
3o  FOR  X = 4  TO 2oo STEP=5o
4o  PRINT X; "      " ; X/X ;"          " ;     X-X
5o  NEXT X
6o  END

RUN
X            X/X              X-X
. . . . . . . . . . . . . . . . . . . . .
4             1                0
54            1                  0
1o4           1                    0
154           1                    0
```

Aliniamentul vertical nu este prea grozav

Fig. 4.7.2.1

În linia 10 se afișează un cap de tabel, cu o spațiere fixă între rubrici. În 20 se afișează o linie ce va sublinia acest cap de tabel pentru a ameliora prezentarea. Linia 30 comandă calculul a 4 valori care sînt: 4, 54, 104, 154. În 40 se afișează valorile calculate, separate prin spații.

Din cauză că X variază ca lungime și numărul de spații separatoare rămîne constant, coloanele de 1 și 0 capătă o alură dezordonată. Cum putem oare remedia acest defect? Cea mai simplă metodă

este de a utiliza tabularea prin intermediul caracterului „ , " pe care BASIC îl traduce într-un ordin de afișare spațiat pe 15 caractere, indiferent de lungimea valorilor afișate. Rezultatul va fi mult mai atrăgător:

```
10  PRINT  "X" , "X/X" , "X-X"
20  PRINT  "_____________________________"
30  FOR  X = 40  TO  200  STEP = 50
40  PRINT X, X/X, X-X
50  NEXT  X
60  END

RUN
X                X/X                X-X
_____________________________________________
4                1                  0
54               1                  0
104              1                  0
154              1                  0
```

15 caractere 15 caractere

Fig. 4.7.2.2

Ordinul PRINT TAB Rezultatul este satisfăcător, dar dacă dorim o spațiere de 8 caractere, nu de 15, cum procedăm? În această situație, este suficient să comandăm o afișare tabulată pe 8 caractere; aceasta se traduce prin ordinul PRINT TAB (8). Va rezulta următorul program, unde numai liniile 10 și 40 au fost modificate față de programul precedent.

```
1o  PRINT  "X" ; TAB(8) "X/X" ; TAB(8) "X-X"
2o  PRINT  "_____"
3o  FOR  X=4  TO  2oo  STEP 5o
4o  PRINT X ; TAB(8) X/X ; X-X
5o  NEXT  X
6o  END

RUN
X       X/X       X-X
_____________________
4         1         0
54        1         0
1o4       1         0
154       1         0
```

8 spaţii 8 spaţii

Fig. 4.7.2.3

Astfel putem realiza, într-o manieră extrem de simplă orice spaţiere dorim, indiferent de lungimea valorilor de afişat.

Atenţie la sintaxă! Verificaţi ca valoarea de tabulare să fie corect scrisă, între paranteze. Nu trebuie să lăsaţi spaţiu între TAB şi valoarea de tabulare.

Iată un alt exemplu în care se prezintă o afişare formatată pe patru coloane cu o aliniere corectă.

```
10 PRINT „X";
20 PRINT TAB(14) „X * * 2";
30 PRINT TAB(29) „X * * 3";
40 PRINT TAB(44) „X * * 4"
50 FOR T=0 TO 63
```

```
 60 PRINT „—";
 70 NEXT T
 80 FOR X = 1 TO 5
 90 PRINT X;
100 PRINT TAB (15) X*X
110 PRINT TAB(15) X * X * X
120 PRINT TAB(15) X * X * X * X
130 NEXT X
140 END
```

Executați acest program și verificați corectitudinea afișării.

Nu toate interpretoarele BASIC recunosc comanda PRINT TAB.

4.7.3 — Trasarea curbelor

Pe același principiu de mai sus se pot trasa curbe. Iată un exemplu simplu:

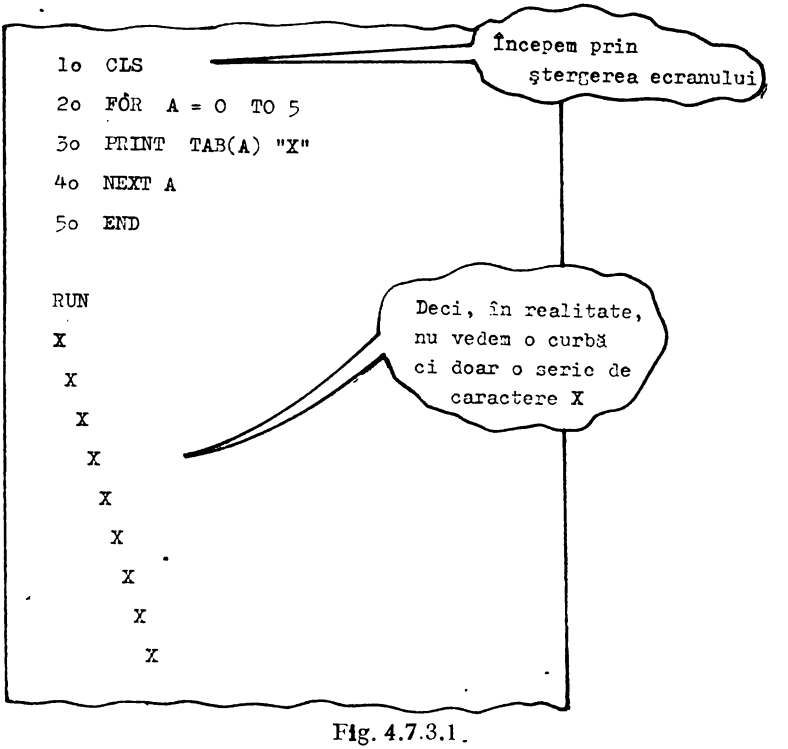


Fig. 4.7.3.1.

Aici variabila *A* este incrementată cu 1 la fiecare pas; în consecință caracterele X vor fi decalate cu o poziție, de la o linie la alta.

Dacă valoarea variabilei *A* nu ar urmări o progresie liniară, atunci am putea trasa curbe. În acest caz ordinul TAB este urmat de o expresie aritmetică:

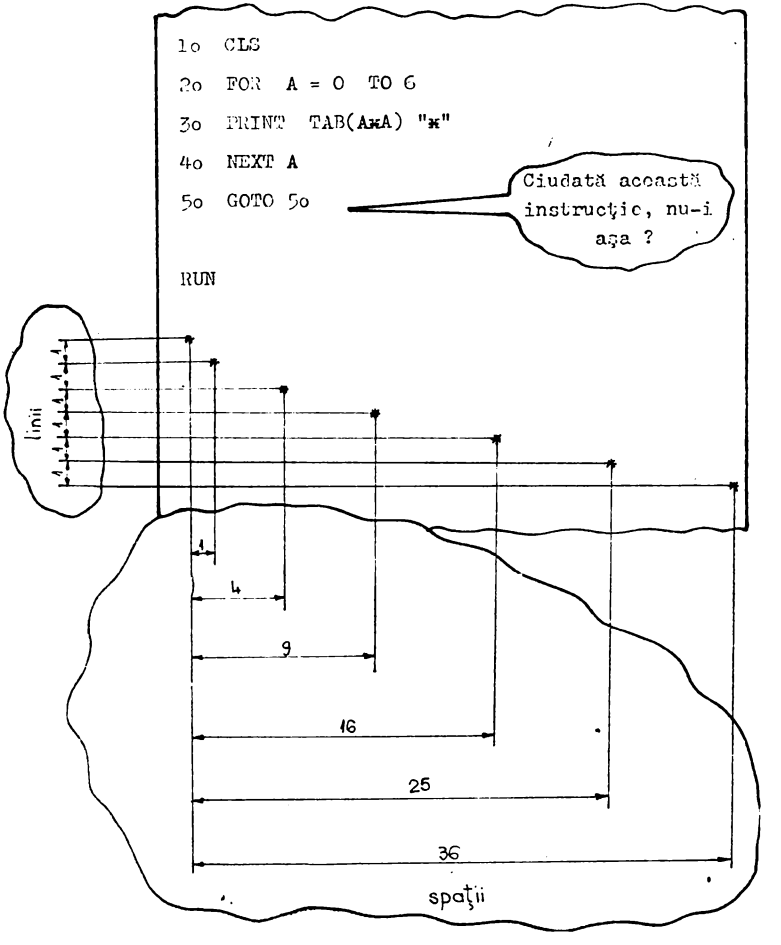


Fig. 4.7.3.2

În acest program, în linia 50, intervine un ordin bizar, la prima vedere. Această instrucție execută un salt la ea însăși. Programul va cicla în acest punct fără să poată ieși din ciclu și fără să poată executa altă operație. Acest lucru se va întîmpla pînă în momentul în care veți apăsa tasta BREAK. Aceasta este o metodă de a pune ordinatorul în așteptare, fără terminarea programului în curs de execuție.

> Procedînd ca mai sus, ordinatorul nu va afișa pe ecran prompterul READY pentru a vă indica că a terminat programul și așteaptă noi ordine. Imaginea va rămîne deci nealterată. Este o metodă des folosită în grafică sau jocuri.

Dacă doriți să măriți numărul de puncte afișate pentru aceeași curbă, este suficient să divizați tabularea, de exemplu cu valoarea 10.

```
10 CLS
20 FOR X=1 TO 25
30 PRINT TAB(X * X/10) „*“
40 NEXT X
50 END
RUN
*
 *
  *
    *
      *
        *
          *
            *
              *
                 *
                   *
                     *
                         *
                            *
                               *
                                  *
                                     *
                                         *
                                             *
                                                 *
                                                      *     *     *     *     *
```

Ordinul PRINT TAB(...) nu ia în considerare decît partea întreagă a valorii calculate.

4.7.4 — Comanda PRINT USING

Comanda PRINT USING este folosită pentru a determina diverse formate de afişaj sau tipărire. Această comandă se aplică pentru tipărirea numerelor, sau a şirurilor de caractere.

În cazul numerelor, comanda PRINT USING stabileşte numărul de cifre posibile înaintea virgulei, poziţia virgulei şi numărul de zecimale. Comanda este urmată de un „format" în care cifrele sînt reprezentate de simboluri „#" (diez).

FORMATARE De exemplu, un format anunţat de un ordin PRINT USING şi urmat de „###.##" defineşte un număr zecimal format din trei cifre înaintea virgulei şi două după.

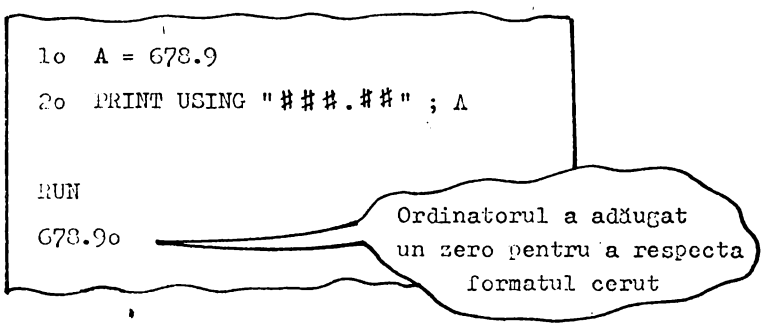


Fig. 4.7.4.1

În continuare, să examinăm un exemplu mai complex de utilizare al comenzii PRINT USING. Vom introduce date cu ajutorul instrucţiei DATA şi le vom citi printr-un READ. Aceste date vor fi afişate mai întîi ca atare, apoi vor fi formatate prin PRINT USING, şi vom constata diferenţa (vezi fig. 4.7.4.2):

Ce se întîmplă, însă, în situaţia în care numărul pe care dorim să-l formatăm depăşeşte partea întreagă a formatului? Iată ce (fig. 4,7.4.3):

```
1o  READ A
2o  PRINT A
3o  IF A < > 1oo  THEN 1o
4o  PRINT
5o  RESTORE
6o  READ A
7o  PRINT USING  "### .##" ; A
8o  IF A < > 1oo  THEN 6o
9o  DATA 1,3.14,642.3,3.721,4567.1,1oo
```

Primul afişaj fără PRINT USING

Lista DATA se reia de la început

Al doilea afişaj cu PRINT USING

```
RUN
 1
 3.14
 642.3
 7.721
 4567.1
 1oo
```

Fără PRINT USING această listă de valori are un aspect dezordonat

```
  1.oo
  3.14
 642.3o
  3.72
4567.1o
 1oo.oo
```

Folosirea lui PRINT USING ameliorează aspectul

Fig. 4.7.4.2

```
1o  A = 9876
2o  PRINT USING  "###.#" ; A

RUN
% 9876.o
```

Numărul este afişat integral.

Semnul "%" indică depăşire de capacitate

Fig. 4.7.4.3

Comanda PRINT USING realizează și o rotunjire a părții fracționare a numărului. De exemplu, dacă dorim să afișăm valoarea 23.876 cu formatul „###.##" vom obține pe ecran valoarea 23.88.

Comanda PRINT USING permite, de asemenea, rezervarea de spații în cadrul formatului.

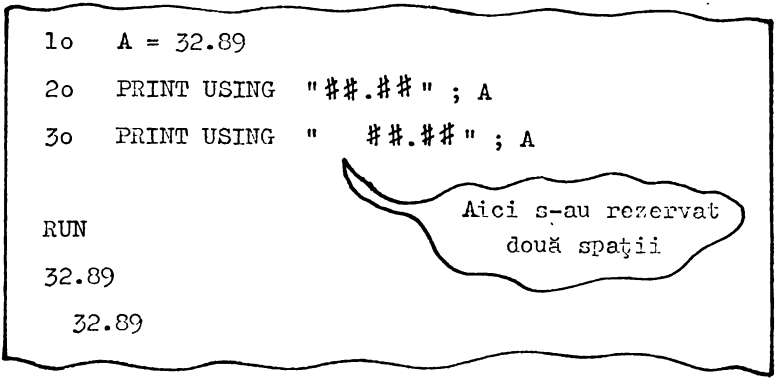


Fig. 4.7.4.4

O altă posibilitate a instrucției PRINT USING este de a separa grupurile de cîte trei cifre printr-o virgulă. Pentru aceasta este suficient să plasăm o virgulă înaintea punctului zecimal.

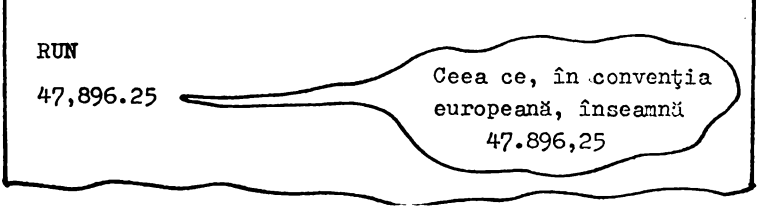


Fig. 4.7.4.5

Două caractere asterisc plasate înaintea formatului, vor avea ca efect umplerea cu asteriscuri a spațiilor rezervate la început. La fel, două simboluri „$$" înaintea formatului vor produce afișarea unui singur simbol „ $" înaintea primei cifre.

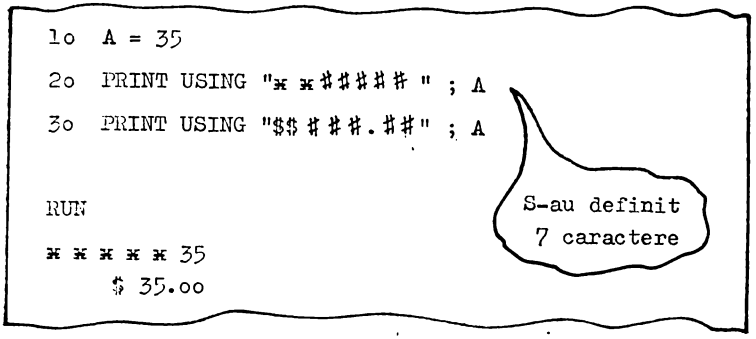


Fig. 4.7.4.6

Un singur simbol „ $" înaintea formatului va determina afișarea numărului dorit, precedat de un simbol „ $", care își păstrează poziția în care a fost definit în format:

Format definit	Număr	Rezultat
$ ###.##	5678	$ 5678.00
$ ###.##	35	$ 35.00

Patru semne de exclamare așezate la sfîrșitul formatului vor avea ca efect afișarea numărului în formă științifică, exprimat prin intermediul puterilor lui 10.

Format definit	Număr	Rezultat
#.# !!!!	35	3.5 E + 01
#.# !!!!	−35	−3.5 E + 01
.### !!!!	35	.350 E + 02
+.## !!!!	35	+.35 E + 02

Valoarea 3.5 E+01 are următoarea semnificație:

$$3{,}5 \times 10^1$$

iar +.35 E+02:$0{,}35 \times 10^2$.

Următorul exemplu ilustrează o altă modalitate de utilizare a comenzii PRINT USING:

```
10 A$ = „###·##"
20 PRINT USING A$; 35
RUN
35.00
```

Rezervarea de spații pentru afișarea unui șir de caractere se face cu ajutorul a două bare oblice. Numărul de spații rezervate este egal cu numărul de spații dintre bare plus 2. Cînd se dorește rezervarea unui singur spațiu se plasează un semn de exclamare.

Format definit	Șir de caractere	Rezultat
/ /	EFGH	EFGH
/ /	EFGHI	EFGH
/ /	ABC	ABC
!	T	T

Iată un ultim exemplu:

```
1o  READ A$ , A
2o  PRINT USING  "/      /"  SINT DE ####KG"; A$ , A
3o  IF A$ = "SF" THEN END
4o  GOTO 1o
5o  DATA  STOCURILE , 32o , IESIRILE, 35
6o  DATA  COMENZILE , 178 , SF

RUN
STOCURILE SINT DE 32o KG
IESIRILE SINT DE 35  KG
COMENZILE SINT DE 178 KG
```

Remarcați acest IF THEN

Fig. 4.7.4.7

Utilizînd comanda PRINT USING vom constata că este extrem de utilă pentru editarea tabelelor, facturilor, comenzilor, formularelor sau a tuturor documentelor care necesită o prezentare standardizată.

Exerciții

1) Comanda PRINT TAB comandă o tabulare: A — pornind de la marginea ecranului; B — pornind de la coloana precedentă.

2) Următoarea secvență de program este corectă: A — Da; B — Nu.

```
10 LET Y = A * B
20 PRINT TAB Y
```

3) Ce va produce următoarea secvență:

```
10 FOR A = 0 TO 7
20 PRINT TAB (A) „*";
30 NEXT A
40 END
- RUN
```

A — * * * * * * * *

B — *

```
        *
          *
            *
              *
                *
                  *
```

4) Cum va apărea numărul 35.1 dacă va fi afișat cu un PRINT USING „####·##": A — 035.10; B — 35.10; C — 35.1.

5) Care va fi rezultatul instrucției: 10 PRINT USING „###·##"; 32.348? A — 32.34; B — 032.34; C — 32.35.

Răspunsuri: 1 = A; 2 — B (nu); ar fi fost da, dacă nu s-ar fi omis parantezele: PRINT TAB(Y) ! 3 = A (s-a folosit caracterul „;" în linia 20); 4 = B; 5 = C.

4.8 — NUMERE ALEATOARE ÎNTREGI ȘI SALTURI MULTIPLE

Acest capitol studiază instrucția ON X GOTO... urmată de multe adrese. Această instrucție poartă numele de „salt multiplu", sau, folosind accepțiunea FORTRAN a aceluiași concept (FORTRAN este un alt limbaj de programare, în general mai cunoscut decît BASIC), de „GOTO calculat". Vom examina, de asemenea, în cadrul prezentului capitol, funcția caracterului „;" în sintaxa BASIC. În sfîrșit, vom vedea cum putem genera un număr aleator

(RND), cum putem transforma un număr fracționar într-unul întreg (INT), sau cum putem reduce numărul de zecimale ale unui număr fracționar. Vom analiza, de asemenea, rolul ordinului RANDOM.

4.8.1 — Salturi multiple

În cursul derulării unui program, sînt dese acele situații cînd este necesară aefectuarea unui salt, la o adresă care depinde de valoarea calculată a unei variabile. Iată cel mai simplu exemplu:

```
1o  PRINT  "INTRODUCETI UN NUMAR ÎNTRE 1 ŞI 3"
2o  INPUT  N
3o  IF N = 1 THEN 8o
4o  IF N = 2 THEN 1oo
5o  IF N = 3 THEN 12o
6o  PRINT "EROARE ! INTRODUCETI DIN NOU NUMARUL"
7o  GOTO 2o
8o  PRINT "ATI INTRODUS 1"
9o  GOTO 13o
1oo PRINT "ATI INTRODUS 2"
11o GOTO 13o
12o PRINT "ATI INTRODUS 3"
13o END

RUN
INTRODUCETI UN NUMAR INTRE 1 ŞI 3
? 5
EROARE! INTRODUCETI DIN NOU NUMARUL
? 2
ATI INTRODUS 2
```

Iată cele trei ramificații posibile

Acestea sînt salturi necondiționate

Fig. 4.8.1.1

Încercați să înțelegeți singuri funcționarea acestui program. Pentru a vă ajuta, iată cîteva explicații: în 20 se solicită introducerea unui număr; dacă acest număr este 1, linia 30 vă ramifică în 80; dacă este 2 linia 40 vă ramifică în 100; dacă este 3 linia 50 vă ramifică în 120. Dacă numărul introdus este diferit de 1, 2 sau 3, atunci vi se arată că ați comis o eroare și sînteți invitați să repetați operația. În liniile 80, 100 și 120 se afișează mesaje diferite, în funcție de valoarea citită.

ON... GOTO Putem simplifica considerabil acest program concatenînd cele trei ordine de ramificație într-unul singur, un salt multiplu, executat printr-un ON...GOTO.

```
10 PRINT „INTRODUCETI UN NUMAR INTRE 1 și 3"
20 INPUT N
30 ON N GOTO 60, 80, 100
40 PRINT „EROARE! INTRODUCETI DIN NOU NU-
   MARUL"
50 GOTO 20
60 PRINT „ATI INTRODUS 1"
70 GOTO 110
80 PRINT „ATI INTRODUS 2"
90 GOTO 100
100 PRINT „ATI INTRODUS 3"
110 END
```

În linia 30, se poate vedea saltul multiplu care înlocuiește liniile 30, 40 și 50 ale programului precedent. În rest, programul nu a mai suferit modificări. Ordinatorul evaluează valoarea lui N și execută un salt la prima adresă pentru N=1, la cea de-a doua pentru N=2, sau la cea de-a treia pentru N=3.

Instrucția ON N GOTO adresă, adresă, ...adresă permite execuția unui număr de salturi egal cu numărul de adrese specificate. Saltul se execută la cea de a i-a adresă, unde i este valoarea lui N.

Se impune o singură condiție, și anume ca N să fie un număr întreg. Evident, un salt la cea de-a 2,5-a adresă, nu ar avea nici un sens. La cea mai mare parte a calculatoarelor, dacă calculul variabilei N conduce la o valoare fracționară, este luată în considerare, pentru efectuarea saltului, numai partea întreagă. De exemplu, dacă N rezultă din calculul 10/3 = 3,333 calculatorul va reține doar valoarea 3 pentru efectuarea saltului.

4.8.2 — Utilizarea operatorului de concatenare

Operatorul de concatenare este caracterul „:". Utilizarea lui într-un program BASIC permite scrierea a două instrucții diferite pe aceeași linie.

De exemplu, programul anterior, ar putea fi scris, dacă utilizăm operatorul de concatenare, astfel:

```
1o  PRINT "INTRODUCETI UN NUMAR INTRE 1 ŞI 3"
2o  INPUT N
3o  ON N GOTO 5o, 6o, 7o
4o  PRINT "EROARE ! INTRODUCETI DIN NOU NUMARUL" : GOTO 1o
5o  PRINT "ATI INTRODUS 1" : GOTO 8o
6o  PRINT "ATI INTRODUS 2" : GOTO 8o
7o  PRINT "ATI INTRODUS 3"
8o  END
```

Utilizarea operatorului de concatenare

Fig. 4.8.2.1

4.8.3 — Generarea unui număr aleator

Problema generării unui număr aleator se rezumă la a cere ordinatorului să furnizeze un număr oarecare, la întîmplare. Acest număr ales la întîmplare, se numește număr aleator („random number" în engleză).

RND Comanda RND permite obținerea unui număr aleator. Să luăm cîteva exemple. Dacă dorim să obținem un număr aleator cuprins între 0 și 1, vom scrie: RND(0).

```
1o  LET  N = RND(0)
2o  PRINT N
3o  END

RUN
 .47381
```

Am ales aici un număr la întîmplare

Fig. 4.8.3.1

În practică, liniile 10 și 20 pot fi condensate într-un singură:

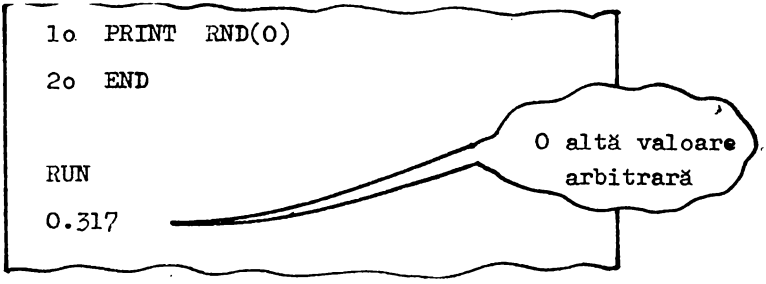


Fig. 4.8.3.2

Numărul aleator obținut nu este un veritabil număr aleator, pentru că rezultă din calculul unei funcții care poate genera un număr foarte mare de valori. Spunem că avem de-a face cu un număr pseudo-aleator, ceea ce este mult mai aproape de adevăr.

Atenție: anumite interpretoare BASIC nu admit comanda RND(0), ci RND(1), pentru a obține același rezultat. Pentru alte interpretoare, comanda RND este echivalentă cu RND(0). Unele interpretoare, ca de pildă BASIC-AMS, BASIC-18 și BASIC-118 furnizează valori aleatoare cuprinse între 0 și 1. De exemplu:

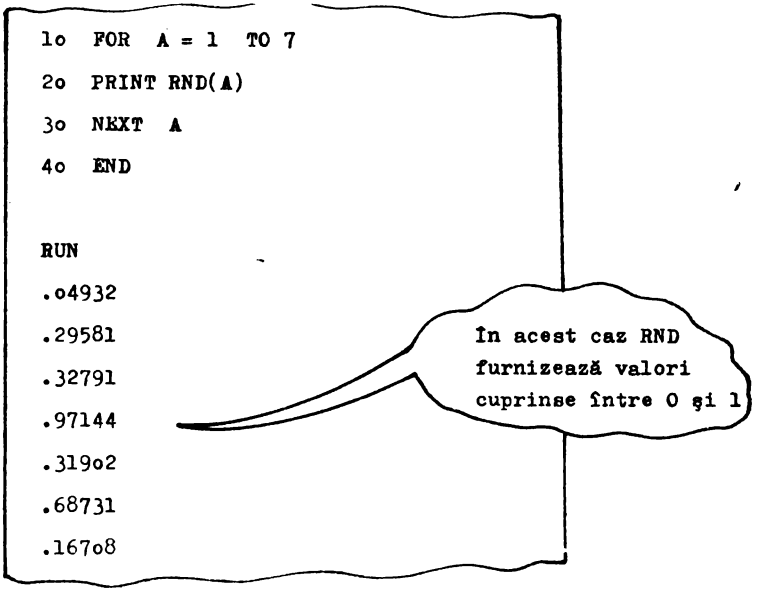


Fig. 4.8.3.3

Există interpretoare BASIC care furnizează valori aleatoare întregi. Astfel, ordinul RND(N) va determina obținerea unei valori aleatoare întregi, cuprinse între 0 și N. De exemplu:

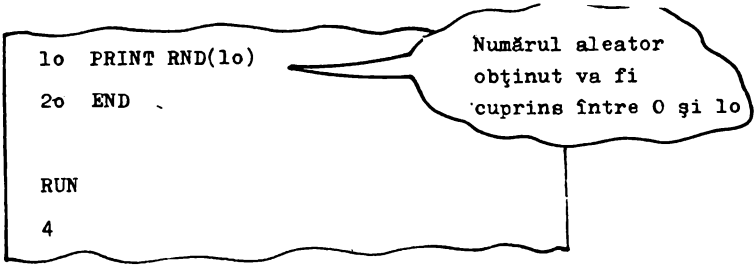


Fig. 4.8.3.4

Să presupunem acum că dorim să obținem un număr aleator, nu neapărat, întreg, cuprins între 2 și 10. Utilizînd comanda RND(10) vom obține un număr cuprins între și 10, ceea ce nu este convenabil. Pentru a rezolva această problemă vom utiliza comanda RND(0), care ne va furniza un număr între 0 și 1, iar valoarea obținută o vom ajusta, după cum urmează:

```
10 PRINT (RND(0) * 8 + 2
20 END
```

Această formulă complexă va produce următoarele operații:
1. RND(0) va produce un număr arbitrar 0.17421
2. Acest număr înmulțit cu 8 va rezulta 1.39368
3. După ce adăugăm 2 obținem 3.39368

Deci, introducînd o comandă RUN:

```
RUN
3.39368
```

Să notăm că această soluție nu este perfectă, lucru ce poate fi verificat prin faptul că limitele intervalului, numerele 2 și 10, nu vor fi obținute niciodată. Acest lucru împietează asupra caracterului aleator al numărului obținut.

Interpretoarele BASIC au defectul de a diferi între ele. Veți putea găsi următoarele variante ale instrucției RND(X):
1. Dacă $X > 0$, RND(X) furnizează valoarea aleatoare în cadrul seriei;
2. Dacă $X = 0$, RND(X) furnizează ultima valoare aleatoare deja obținută.
3. Dacă $X < 0$, începe o nouă secvență aleatoare.

4.8.4 — Revenirea la numere întregi

INT Ordinul RND furnizează valori fracționare, cuprinse între 0 și 1. Uneori avem nevoie de numere întregi. Pentru aceasta este necesar să suprimăm tot ceea ce se găsește în dreapta virgulei. Acest lucru se poate obține folosind ordinul INT.

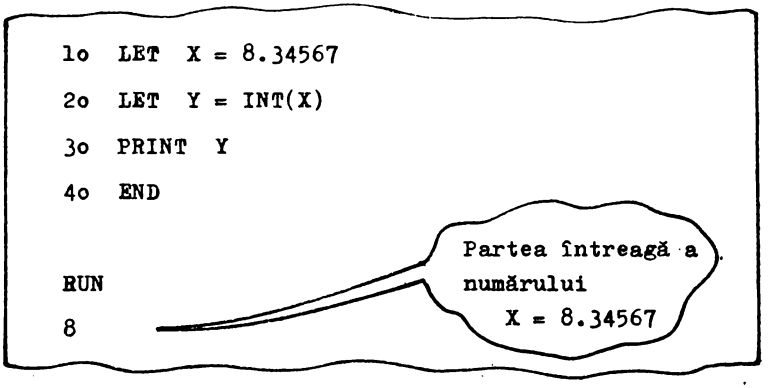


Fig. 4.8.4.1

Pentru mai multă simplitate am fi putut scrie:

```
10 PRINT INT (8.34567)
20 END
RUN
8
```

4.8.5 — Reducerea numărului de zecimale

Pentru că ne aflăm în acest punct unde vorbim despre numere zecimale (fracționare) și numere întregi, iată cum putem reduce numărul de zecimale al unei valori date. Să presupunem că dorim să transformăm valoarea 6.10345 lei (care a reieșit dintr-un calcul, pentru că altfel nu are nici un sens) în prețul 6.10 lei. Pentru a obține acest rezultat este suficient să utilizăm ordinul INT în cadrul următoarei secvențe:

1. vom înmulți 6.10345 cu 100 și vom obține 610.345;
2. vom determina partea întreagă din 610.345 și va rezulta 610;
3. în sfîrșit, vom împărți cu 100 și vom obține 6.10;

Iată programul:

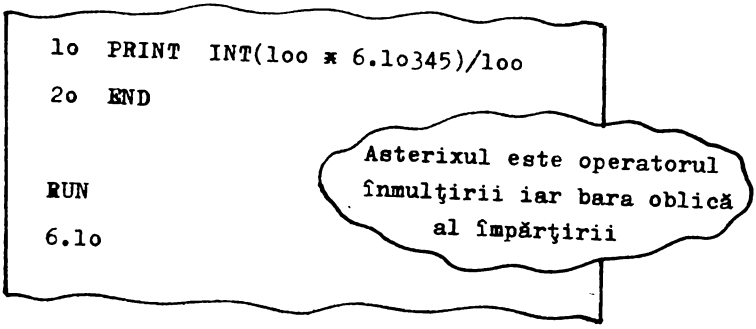


Fig. 4.8.5.1

Pentru a reduce la N numărul zecimalelor ale unui număr X se calculează:

INT((10 * * N) * X)/(10 * * N)

În anumite situaţii este preferabilă reducerea numărului de zecimale prin rotunjire, mai degrabă decît prin trunchiere. Rotunjirea este o operaţie în urma căreia, numărul 3.643 de exemplu, va fi transformat în 3.64, în timp ce numărul 3.647 va deveni 3.65. Acest lucru poate fi obţinut adăugînd 5 la ultima zecimală.

```
10 A = 3.647
20 A = A + .005
30 PRINT INT(100 * A)/100
40 END
RUN
3.65
```

4.8.6 — Reiniţializarea seriilor aleatoare

Înainte de a încheia discuţia despre numere aleatoare, este necesar să indicăm o problemă ce uneori poate genera neplăceri. Aşa cum am arătat, valorile generate nu sînt decît pseudo-aleatoare. Ele sînt determinate pornind de la un mic program inclus în interpretorul BASIC, program al cărui algoritm foloseşte pentru generarea seriilor aleatoare o valoare fixă. Asta înseamnă că, ori de cîte ori vom folosi aceeaşi valoare iniţială, vom obţine aceeaşi serie pseudo-aleatoare. Dacă dorim să evităm acest lucru (fără stingerea ordinatorului care provoacă modificarea aleatoare a valorii iniţiale) o putem face utilizînd ordinul RANDOM care la anumite interpretoare BASIC se scrie RANDOMIZE. Iată un exemplu:

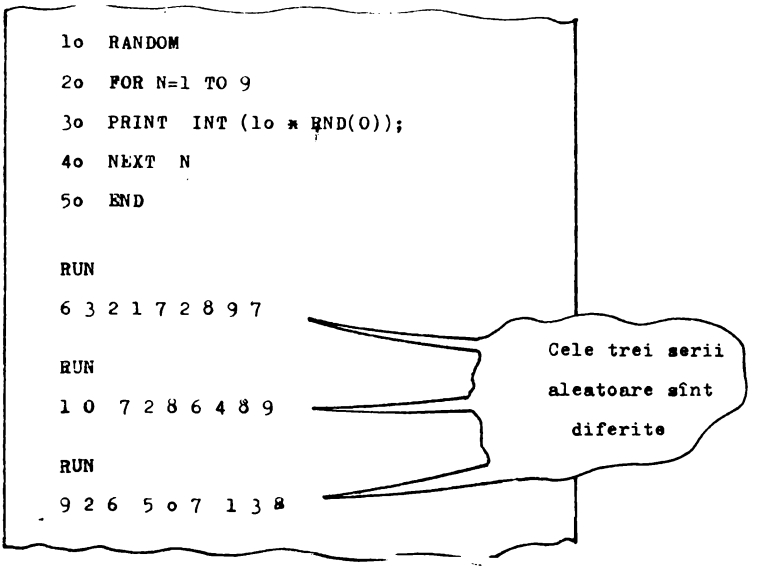


Fig. 4.8.5.2

Exerciții

1) Instrucția următoare este corectă?

10 IF N GOTO 60; 80; 100

A — Nu; B — Da.

2) Instrucția următoare este corectă?

50 ON N GOTO 80, 30, 100

A — Da; B — Nu.

3) În exemplul de mai sus, N poate fi expresie fracționară?

A — Da; B — Nu.

4) Ordinul RND provine din: A — ROUND (pentru rotunjire); B — RANDOM (aleator).

5) INT (543.2109) va produce: A = 543; B = 543.21.

6) INT (−27.89) va produce: A = −27; B = −28.

Răspunsuri: 1 — A; 2 — A; 3 — A (în cea mai mare parte din cazuri) 4 — B; 5 — A; 6 — B.

4.9 — ARITMETICĂ ELEMENTARĂ

A sosit acum timpul să ne oprim puțin asupra operațiilor aritmetice pe care le autorizează limbajul BASIC: adunare, scădere, înmulțire, împărțire, ridicare la putere, extragerea rădăcinii pătrate. Vom defini, de asemenea, noțiunea de variabilă numerică, vom preciza lungimea acestor variabile și vom prezenta notația științifică.

4.9.1 — Operații aritmetice și priorități

Limbajul BASIC permite execuția următoarelor operații aritmetice:

- adunarea cu semnul „+"
- scăderea cu semnul „—"
- înmulțirea cu semnul „*"
- împărțirea cu semnul „/"
- ridicarea la putere cu unul dintre semnele „**", „↑".

Datorită faptului că în aceeași expresie pot interveni mai multe operații, este necesară stabilirea unei priorități care să indice ordinea executării diferitelor operații din acea expresie. De exemplu, în expresia $3 \times 5 + 2$, dacă se execută mai întîi operația 3×5, rezultatul va fi 17, dacă se execută mai întîi operația $5 + 2$, rezultatul va fi 21. Este deci necesară existența unei priorități de execuție a operațiilor aritmetice.

În BASIC, ridicarea la putere, înmulțirea și împărțirea, au prioritate față de adunare și scădere.

Dacă apar situații cînd, dintr-un motiv sau altul, doriți să executați operații aritmetice într-o altă ordine decît cea stabilită mai sus, nu ezitați să folosiți parantezele. Astfel:

$$8 + 14/2 = 15, \text{ dar } (8 + 14)/2 = 11.$$

4.9.2 — Extragerea rădăcinii pătrate

Pentru a rezolva problema extragerii rădăcinii de indiferent ce ordin dintr-un număr, este necesar să ne amintim cîteva reguli elementare de matematică. Astfel, rădăcina cubică din A se poate scrie:

$$\sqrt[3]{A} = A^{1/3}$$

Astfel, pentru a extrage rădăcină cubică din $A + B$ vom scrie:

$$(A + B) * * (1/3)$$

O foarte frecventă sursă de erori este, în acest caz, omiterea parantezelor. De exemplu, fie $A = 9$, $B = 18$; BASIC va calcula:

1. dacă expresia este corect scrisă, $(9 + 18) * *1/3$, vom avea mai întîi $9 + 18$, deci 17 și apoi $27 * * 1/3$ adică 0.333...3.

Dacă omiteți parantezele, atunci formula devine (9 + 18) * * 1/3 și vom avea 9 + 18 = 27, 27 * * 1 = 27, 27/3 = 9. După cum observați există o diferență foarte mare între cele două rezultate.

SQR Această metodă este aplicabilă pentru extragerea rădăcinii de orice ordin, dar pentru extragerea rădăcinii pătrate BASIC dispune de un operator special SQR. De exemplu:

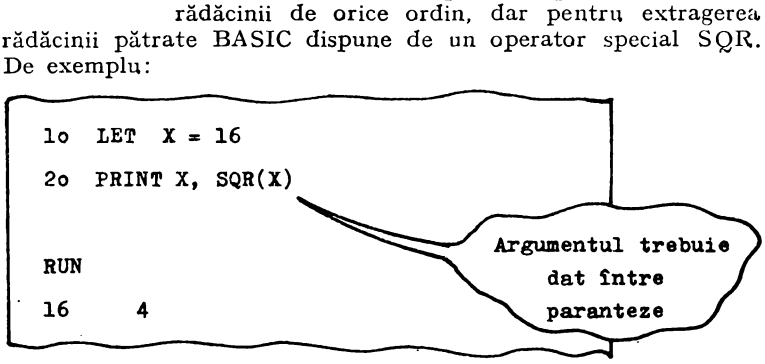


Fig. 4.9.2.1

Prin urmare dispunem de două metode pentru calculul unei rădăcini pătrate. Le vom ilustra cu ajutorul următorului program care calculează rădăcinile unei ecuații de gradul 2.

Evident, vă mai amintiți relația de calcul a rădăcinilor unei ecuații de gradul al doilea.

$$X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Există două rădăcini, dar cu condiția ca discriminantul (expresia $B^2 - 4AC$) să fie pozitiv. Iată programul:

```
 10 INPUT A, B, C
 20 LET D = B↑ 2 − 4 * A * C
 30 IF D < 0 THEN 90
 40 LET R = D↑ .5
 50 LET X1 = (−B + R)/(2 * A)
 60 LET X 2 = (−B − R)/(2 * A)
 70 PRINT „RADACINILE ECUATIEI SINT“;X1;“ și “;X2
 80 STOP
 90 PRINT „NU EXISTA RADACINI REALE“ ;
100 END
RUN
? 3, −5, 2
RADACINILE ECUATIEI SINT 2 și − .33334
RUN
? 4, 5, 6
NU EXISTA RADACINI REALE
```

4.9.3 — Cîteva noțiuni complementare

CONSTANTĂ NUMERICĂ În BASIC, orice valoare numerică, ca de exemplu 63, —28 etc., se numesc constante numerice, pentru că valoarea lor este fixată și nu se mai schimbă.

VARIABILĂ NUMERICĂ IDENTIFICATOR Valorile numerice ale unui program BASIC care se schimbă se numesc variabile numerice. Numele unei variabile numerice este un identificator. De exemplu, în instrucția:

```
10 LET P = 63
```

P este o variabilă numerică sau un identificator.

O variabilă numerică este reprezentată printr-o literă, urmată eventual de o cifră. Astfel, B, C3, X1 sînt variabile numerice. Dimpotrivă: 3C sau BB nu sînt recunoscute ca variabile.

PUNCTUAȚIA Să ne amintim că americanii utilizează punctul, drept marcă zecimală, în locul virgulei. Astfel, dacă doriți să scrieți 3,14 în BASIC, veți scrie 3.14. De asemenea un zero înaintea unui punct care nu mai este urmat de nici o altă cifră, este superfluu. Deci, .35 este echivalent cu 0,35. Tot astfel, zerourile nesemnificative de după punctul zecimal sînt inutile și vor fi suprimate. Vom scrie mai curînd 5.2 dec.t 5.20.

LUNGIMEA UNUI NUMĂR Lungimea unui număr (numărul său de cifre) este, de asemenea, limitată și depinde de interpretorul BASIC utilizat. Dacă interpretorul cu care lucrați nu acceptă decît numere de maximum 8 cifre, atunci veți avea dreptul să folosiți valori cuprinse între — 12345678 și 12345678. Dacă încălcați această regulă, veți primi pe ecran un mesaj de eroare.

VIRGULA FLOTANTĂ Există interpretoare de BASIC foarte elementare de tip „TINY BASIC" care nu suportă zecimalele. Dacă efectuați operația 10/3 veți obține rezultatul 3. Aceste interpretoare lucrează numai cu numere întregi. Există însă și interpretoare puternice de tipul celor implementate pe echipamentele românești de calcul, care furnizează rezultatele însoțite de zecimalele corespunzătoare. De pildă, rezultatul operației 10/3 va fi 3.3333. Despre aceste interpretoare vom spune că utilizează aritmetica în virgulă flotantă. Constantele în virgulă flotantă se mai numesc și reale.

NOTAȚIA ȘTIINȚIFICĂ Notația științifică este utilizată pentru a putea lucra mai ușor cu valori mari. De exemplu valoarea 3.000.000.000 (trei miliarde) se va scrie, în notație științifică, 3.10^9, sau folosind sintaxa BASIC 3*10**9. Din cauză că 10 este în mod implicit baza ridicată la o putere dată, nu vom mai scrie 10 ci E, suprimînd și simbolul înmulțirii.

Iată cîteva exemple:

$3.10^9 = 3\ E + 9$
$5.10^{-4} = 3\ E - 4$
$.123456\ E - 4 = 0.0000123456$
$.123456\ E + 7 = 1234560$

Exerciții

1) Expresia următoare este corectă?

10 PRINT 3(4+2)/5

A — da; B — nu.

2) Această expresie este corectă?

20 PRINT (4+3)(3+2)+3)/5

A — da; B — nu.

3) Următoarele instrucții sînt echivalente?

PRINT B*B și PRINT B↑2

A — da; B — nu.

4) Aceste ordine sînt echivalente?

PRINT N↑(1/2) și PRINT SQR(N)

A — da; B — nu.

5) În programul care calculează rădăcinile ecuației de gradul doi putem înlocui linia 20 cu:

20 LET D = B*B−4*A*C

A — da; B — nu.

Răspunsuri: 1=B, 2=B, 3=A, 4=A, 5=A.

4.10 — SUBPROGRAME

În acest capitol vom examina rolul subprogramelor și instrucțiile GOSUB și RETURN. De asemenea vom prezenta modalități de apel condiționat al subprogramelor utilizînd ON...GOSUB și IF...THEN GOSUB. În sfîrșit, vom vedea cum poate fi înlocuită utilizarea subprogramelor cu utilizarea funcțiilor definite (DEF FN).

Vom profita, de asemenea, pentru a marca cîteva diferențe limbajul BASIC standard și BASIC extins.

4.10.1 — Instrucțiile GOSUB și RETURN

Un subprogram este un program distinct, de sine stătător, la care putem face apel ori de cîte ori este necesar. În acest caz, apelul se face prin numărul primei sale linii. Pentru a le diferența, programul care apelează un subprogram poartă numele de „program principal". Pentru a înțelege modul de funcționare al programului principal și subprogramului, vom recurge la următorul exemplu simplu: avem două programe, un program principal și un subpro-

gram; programul principal solicită introducerea a două numere; odată introduse aceste două numere, programul principal apelează subprogramul care efectuează suma celor două numere. Apoi, subprogramul efectuează un RETURN prin care informează programul principal în legătură cu terminarea misiunii sale. Acesta din urmă va afișa rezultatul obținut. Să notăm deci două amănunte:

1. Apelul subprogramului se face prin ordinul GOSUB.
2. Revenirea din subprogram se face prin RETURN. Iată programul:

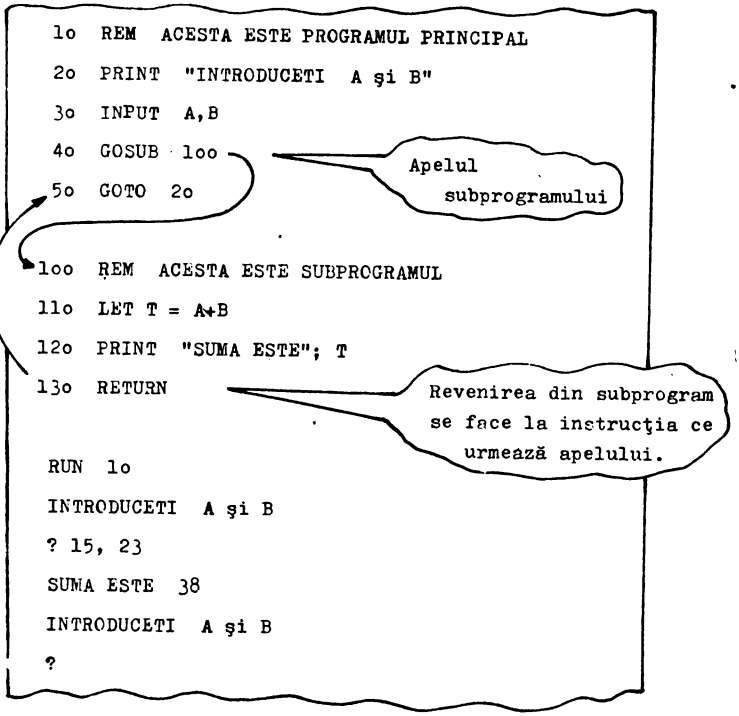


Fig. 4.10.1.1

Reguli:

- Apelul unui subprogram se face prin GOSUB urmat de numărul instrucției cu care începe subprogramul.
- Revenirea din subprogram, RETURN, nu menționează nici un număr de instrucție, ea făcîndu-se întotdeauna la instrucția care urmează apelului.

Modul de funcționare al tandemului program principal-subprogram, poate fi ilustrat prin următoarea modalitate grafică:

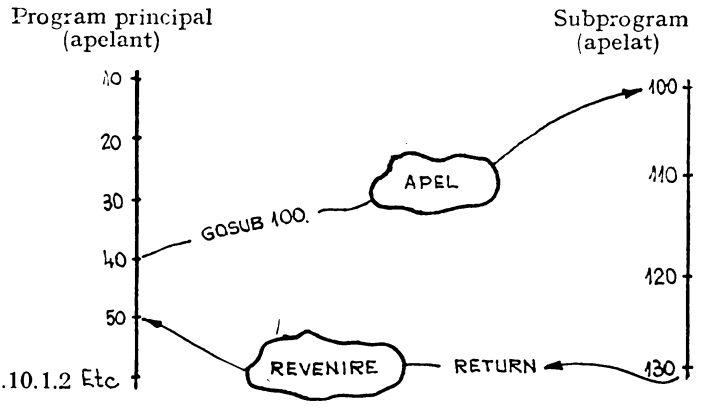


Fig. 4.10.1.2 Etc

4.10.2 — Calculul celui mai mare numitor comun

Un program principal poate apela de mai multe ori același subprogram. De fiecare dată programul principal va indica adresa subprogramului prin numărul primei sale linii. Revenirea din subprogram se va face la instrucția imediat următoare apelului. Acest proces poate fi ilustrat de următoarea diagramă:

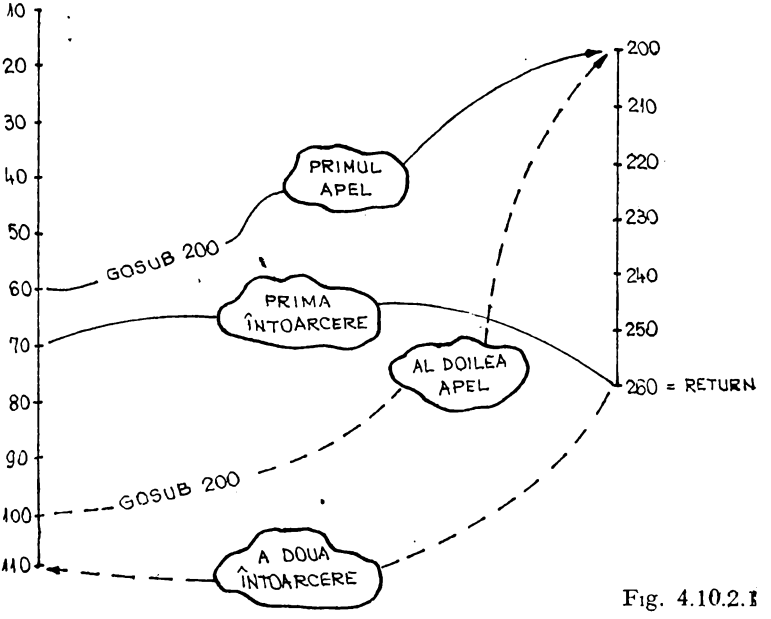


Fig. 4.10.2.1

În cele ce urmează prezentăm o ilustrare a principiului de mai sus, utilizînd exemplul faimosului „cel mai mare numitor comun", care procură atîta bătaie de cap elevilor cursului elementar. Programul următor determină cel mai mare numitor comun a trei numere, *A*, *B* și *C*.

```
 1o  REM  PROGRAM PRINCIPAL
 2o  PRINT "INTRODUCETI A, B şi C"
 3o  INPUT  A,B,C
 4o  LET  X = A
 5o  LET  Y = B
 6o  GOSUB  2oo
 7o  LET  X = G
 8o  LET  Y = C
 9o  GOSUB  2oo
1oo  PRINT "CMMDC =" ; G
11o  GOTO 2o

2oo  REM  SUBPROGRAM
21o  LET  Q = INT(X/Y)
22o  LET  R = X - Q * Y
23o  IF  R = 0  THEN 3oo
24o  LET X = Y
25o  LET Y = R
26o  GOTO 2oo
3oo  LET G = Y
31o  RETURN
32o  END

RUN
INTRODUCETI  A,B şi C.
? 6o, 9o, 12o
CMMDC = 3o
INTRODUCETI  A,B şi C
? 38456, 64872, 98765
CMMDC = 1
?
```

Primul apel

Primul punct de întoarcere

Al doilea apel

Al doilea punct de întoarcere

Fig. 4.10.2.2

4.10.3 — Subprograme în cascadă

Atunci cînd problemele pe care le avem de rezolvat cu ordinatorul se dovedesc a fi complexe, putem utiliza mai multe subprograme diferite. Este posibil ca un subprogram să apeleze, la rîndul său, un alt subprogram care să apeleze și el un subprogram etc. Această tehnică de programare este îndeobște cunoscută sub numele de „apel de subprograme în cascadă". Iată o schemă de principiu a acestei metode, schemă în cadrul căreia ne-am limitat la un număr de două subprograme controlate de un program principal.

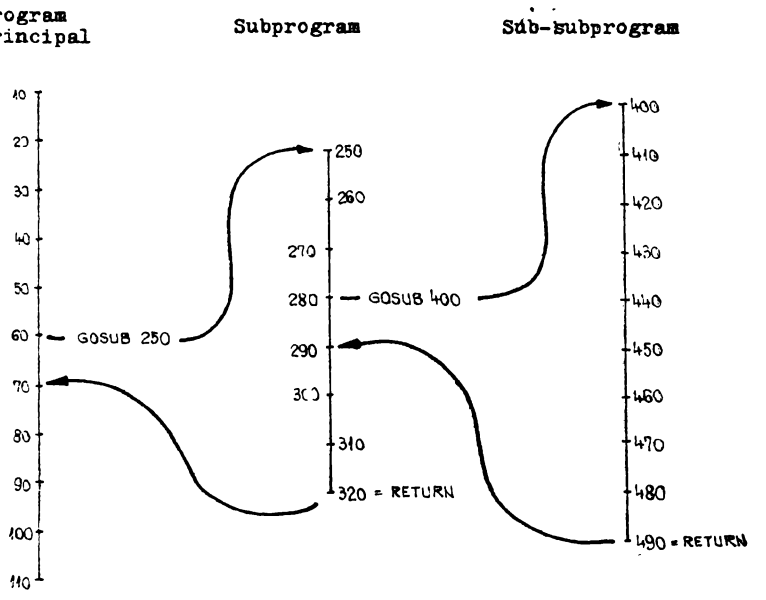


Fig. 4.10.3.1

Interpretorul BASIC este cel care realizează gestiunea adreselor de revenire din subprogram.

Aceasta înseamnă că nu putem reveni dintr-un așa-zis subsubprogram în programul principal, decît prin intermediul subprogramului care joacă rolul de apelant pentru sub-subprogram. Acest lucru se face numai prin RETURN.

O remarcă importantă:

Subprogramele trebuie să fie juxtapuse. Ele nu pot fi imbricate unele în altele căci acest lucru poate cauza erori grave.

4.10.4 — Apelul condiționat al subprogramelor

Utilizarea subprogramelor este o metodă ce prezintă avantaje majore. Realizînd o distribuire a funcțiilor, această metodă permite scrierea unor programe mai bune, sub aspectul clarității și al lizibilității. Din acest motiv, au fost prevăzute și instrucții de apel condiționat al subprogramelor, instrucții care derivă din GOTO și IF.

În paragrafele anterioare, am putut vedea cum se organizează ramificațiile multiple:

```
70 ON X GOTO 200, 300, 400
```

Să ne amintim că, după cum X este evaluat la una dintre valorile: 1, 2 sau 3, se va efectua o ramificație către una dintre instrucțiile 200, 300 sau 400. Putem organiza în așa fel lucrurile, încît instrucțiile cu numerele 200, 300, 400, să reprezinte tot atîtea începuturi de subprograme, fiecare dintre ele încheiate prin RETURN Vom scrie atunci:

```
70 ON X GOSUB 200, 300, 400
```

Am înlocuit, pur și simplu, ordinul GOTO cu GOSUB; revenirea din subprogram se va face obligatoriu la instrucția următoare instrucției nr. 70.

Putem proceda în aceeași manieră cu ordinul IF...THEN.

```
100 IF A = B THEN GOSUB 200
```

Putem, de asemenea, folosi ordinul mai puternic IF...THEN... ...ELSE evident, cu condiția ca interpretorul BASIC implementat pe mașina dv. să o permită):

```
80 IF A=B THEN GOSUB 500 ELSE GOSUB 600
```

Putem combina, fără îndoială, toate aceste forme de apel al subprogramelor, creînd numeroase structuri și variante.

4.10.5 — Definirea funcțiilor

De multe ori se întîmplă ca în mai multe puncte ale unui program să trebuiască să folosim una și aceeași secvență de instrucțiuni, secvență mult prea succintă pentru ca scrierea unui subprogram să fie justificată. Pentru aceste situații, limbajul BASIC prevede așa-zise funcții utilizator, apelabile în orice punct al programului. Să presupunem, de exemplu, că avem de calculat adesea suprafața unui cerc de rază R cunoscută. Formula este πR^2. Bineînțeles că nu vom scrie un subprogram al cărui obiect să constituie efectuarea acestui calcul. Vom prefera să definim o funcție pe care să o apelăm ori de cîte ori dorim. Ordinul utilizat pentru definirea unei funcții este DEF FN, dar FN este imediat urmat de o literă atribuită funcției și care va fi numele acesteia.

DEF FN Iată, de exemplu, definiția funcției S (de la suprafață):

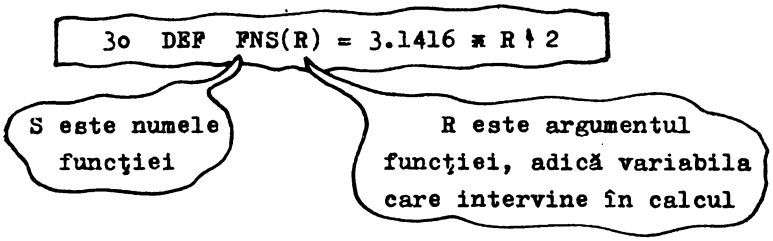


Fig. 4.10.5.1

Iată cum arată un program care utilizează funcții definite:

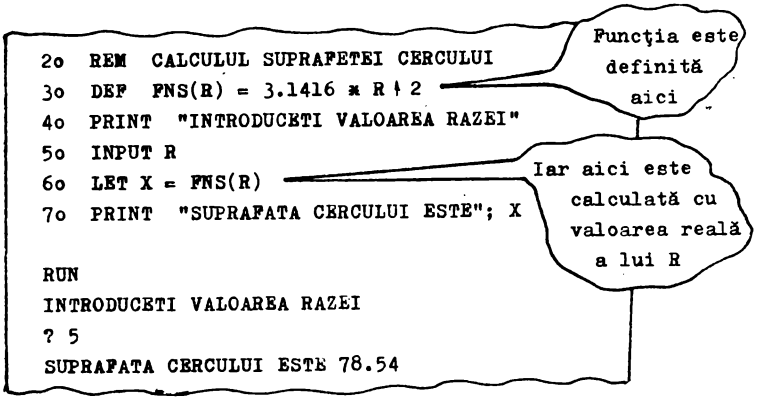


Fig. 4.10.5.2

Iată un al doilea exemplu de utilitate practică. Este vorba despre un program care efectuează conversia din grade Celsius în grade Fahrenheit:

```
10 REM CONVERSIE CELSIUS-FAHRENHEIT
20 DEF FNT(X) = (9/5) * X + 32
30 PRINT „DORITI O CONVERSIE?";
35 INPUT A$
40 IF A$ = „NU" THEN 90
50 PRINT „IN CELSIUS,"
55 INPUT C
60 PRINT „IN FAHRENHEIT"; FNT(C)
80 GOTO 30
90 END
RUN
DORITI O CONVERSIE? DA
IN CELSIUS
? 0
IN FAHRENHEIT 32
DORITI O CONVERSIE? NU
```

Atunci cînd cunoaștem valoarea variabilei, este suficient să o nominalizăm între paranteze :

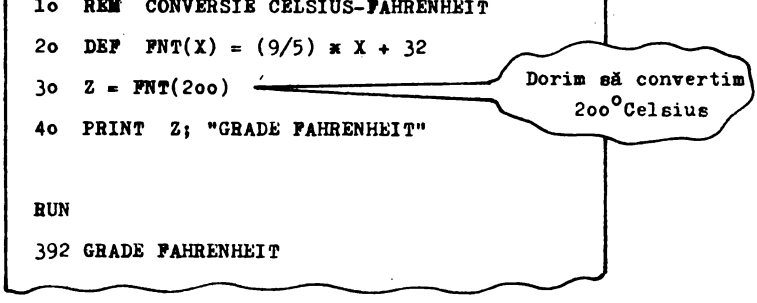


Fig. 4.10.5.3

4.10.6 — BASIC standard, BASIC extins

Limbajul BASIC există în diverse dialecte, versiuni și variante de implementare, în funcție de tipul ordinatorului gazdă. Limbajul de bază se numește BASIC standard, față de care există și un BASIC extins, cunoscut și sub numele de BASIC-PLUS.

INPUT și PRINT combinate

Dacă vom rescrie în BASIC extins programul de conversie Celsius-Fahrenheit vom obține următoarea versiune:

```
1o  REM  * CONVERSIE CELSIUS-FAHRENHEIT *
2o  DEF FNT(X) = (9/5) * X + 32
3o  INPUT  "DORITI O CONVERSIE"; A$
4o  IF  A$ = "NU"  THEN 9o
5o  INPUT  "IN CELSIUS"; C
6o  PRINT  "IN FAHRENHEIT"; FNT(C) : GOTO 3o
9o  END
```

De remarcat acest PRINT + INPUT

Fig. 4.10.6.1

În BASIC extins:

- Putem grupa ordine PRINT și INPUT într-un singur INPUT, partea care corespunde lui PRINT terminîndu-se întotdeauna cu caracterul „;".
- Două instrucții separate prin „:" pot apărea pe aceeași linie.
- Ordinul de atribuire LET este opțional.
- Etc. ...dar este suficient pînă aici.

Exerciții

1) Următoarea instrucție este corectă?

200 GO SUB 50

A — nu; B — da.

2) Următoarea instrucție este corectă?

80 RETURN 200

A — da; B — nu.

3) Putem apela de mai multe ori același program? A — da; B — nu.

4) Putem apela condiționat un subprogram? A — da; B — nu.

5) Putem reveni (RETURN) condiționat dintr-un subprogram? A — da; B — nu.

6) Un subprogram poate apela un alt subprogram? A — da; B — nu.

7) Putem reveni dintr-un al doilea subprogram (dintr-un sub-subprogram) direct în programul principal? A — da; B — nu.

8) Care dintre instrucțiile următoare este corectă?

A — DEFN(X)... B — DEF FN 3... C — DEF FNX...
D — DEF FNA 3...

9) Următoarea instrucție este corectă?

```
50 LET X = FNA(17)
```

A — da; B — nu.

10) Putem scrie:

```
120 INPUT „DA SAU NU?"; A$
```

A — da; B — nu.

Răspunsuri: 1 — A; 2 = B; 3 = A; 4 = A; 5 = B; 6 = A; 7 = B; 8 = C; 9 = A; 10 = A.

4.11 — LISTE ȘI TABLOURI

Pentru a putea manipula mai facil un mai mare volum de informații, se utilizează listele și tablourile de date. Acestea sînt dimensionate și „declarate" cu ajutorul ordinului DIM. Așa cum vom vedea în continuare, elementele acestor structuri de date sînt „calificate" utilizînd așa-zisele „variabile indexate".

4.11.1 — Variabile indexate; declarația DIM

Pînă acum, în exemplele furnizate, am recurs la utilizarea variabilelor numerice și a celor de tip șir de caractere. Acestea erau referite fie printr-o literă (A, B, X, ...) eventual însoțite și de o cifră (A3, Z2, ...), în cazul variabilelor numerice, fie de o literă urmată de simbolul „$" (A$, A$, ...), în cazul variabilelor de tip șir. Toate aceste variabile erau unice, distincte.

Există însă situații cînd este dificil de știut dinainte, cîte variabile vom folosi, și care vor fi cele efectiv prelucrate și prezentate. Iată în continuare un astfel de program în care dorim să aflăm vîrsta medie a unui grup de persoane și abaterea individuală de la această vîrstă medie. Din cauză că avem de calculat o abatere, este necesar să memorăm fiecare vîrstă individuală, prin urmare, să rezervăm un număr de variabile egal cu numărul de persoane care intră în alcătuirea grupului.

Pentru a simplifica problema, să presupunem pentru început că avem de-a face cu un grup de patru persoane. Le vom atribui variabilele *A1*, *A2*, *A3*, și *A4*. Notînd cu *S* suma anilor, iar cu *N* numărul vîrstelor introduse, vom obține următorul program:

```
 5  REM * IATĂ UN PROGRAM DETESTABIL *
10  LET  S=0
15  LET  N=0
20  PRINT  "INTRODUCETI VIRSTA DUPA FIECARE ?"
30  PRINT  "INTRODUCETI 0 PENTRU TERMINAREA PROGRAMULUI"
40  INPUT  A1
50  IF  A1= 0  THEN 200
60  N = N+1
70  S = S+A1
80  INPUT A2
90  IF A2 = 0  THEN 200
100 N = N+1
110 S = S+A2
120 INPUT A3
130 IF A3 = 0  THEN 200
140 N = N+1
150 S = S+A3
160 INPUT A4
170 IF A4 = 0  THEN 200
180 N = N+1
190 S = S+A4
200 LET M = S/N
210 PRINT  "VIRSTA MEDIE ESTE"; M ; "ANI"
220 PRINT  "ABATERILE INDIVIDUALE SINT;"
230 PRINT  A1-M, A2-M, A3-M, A4-M
240 END
```

După fiecare intrare se actualizează contorul vîrstelor și suma lor.

Fig. 4.11.1.1

Dacă lansăm în execuție programul de mai sus, iată ce va produce el:

```
RUN
INTRODUCETI VIRSTA DUPA FIECARE?
INTRODUCETI 0 PENTRU TERMINAREA
    PROGRAMULUI
? 16
? 28
? 55
? 0
VIRSTA MEDIE ESTE 33 ANI
ABATERILE INDIVIDUALE SINT:
-17       -5        22        -33
```

VARIABILE INDEXATE Această manieră de programare este destul de primitivă. În afară de aceasta, mai există o alta fundamentată pe utilizarea variabilelor indexate. Ideea este următoarea: variabilele pot fi numeroase; dacă li se atribuie, în mod global, constanta P, de exemplu, atunci, pentru a le diferenția, va fi suficient să scriem P urmat de un indice: P_1, P_2, ..., P_{73}, ..., P_{138}... Din păcate, utilizarea indicilor, în cazul mașinii de scris, este imposibilă. De aceea, indicii vor fi înlocuiți de numere de ordine incluse între paranteze rotunde: $P(1)$, $P(2)$, ..., $P(73)$, ..., $P(138)$... Puteți merge atît de departe cît vă permite dimensiunea memoriei.

LISTE În acest fel putem crea liste de valori. O listă de valori este deci o colecție de date referite prin numele lor. Iată un exemplu:

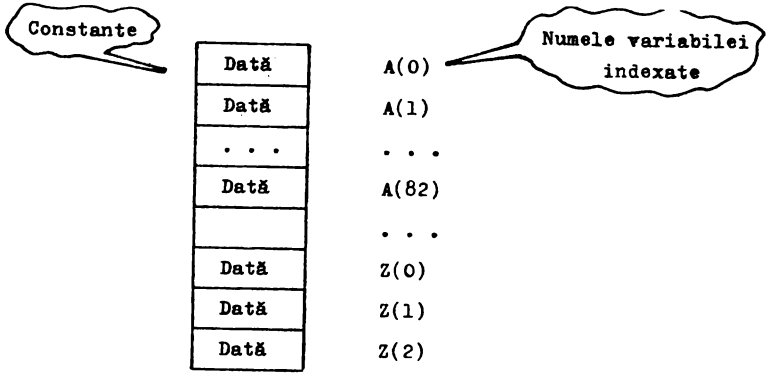


Fig. 4.11.1.2

Fiecare linie din tabelul de mai sus, conține cîte o constantă ce poate fi apelată cu ajutorul variabilei indexate atașate.

DIM Care este lungimea unei liste? Acest lucru se stabilește în funcție de situație, dar trebuie întotdeauna să rezervăm un spațiu suficient de memorie. Pentru aceasta utilizăm ordinul DIM.

```
10 DIM A(4)
```

Această insrtucție declară o listă de patru variabile. Dacă dorim să utilizăm mai multe liste putem scrie:

```
10 DIM A(4)
20 DIM Z(18)
```

sau, mai scurt:

```
10 DIM A(4), Z(18)
```

În continuare, vom rescrie programul care calculează media vîrstelor și abaterea individuală, astfel încît să valorificăm această facilitate a limbajului BASIC.

```
 5 REM * ACEASTA VERSIUNE ESTE DEJA
       ACCEPTABILA —
10 DIM A(20)
20 LET S = 0
30 LET N = 0
40 PRINT „INTRODUCETI VIRSTA DUPA FIECARE?"
50 PRINT „INTRODUCETI 0 PENTRU TERMINAREA
       PROGRAMULUI"
60 FOR K = 1 TO 20
70 INPUT A(K)
80 IF A(K) = 0 THEN 120
90 N = N + 1
100 S = S + A(K)
110 NEXT K
120 LET M = S/N
130 PRINT „VIRSTA MEDIE ESTE"; M; „ANI"
140 PRINT „ABATERILE INDIVIDUALE SINT:"
150 FOR K = 1 TO 20
160 PRINT A(K) — M;
170 NEXT K
180 END
```

După cum se poate vedea, această ultimă variantă deși este cu 5 instrucțiii mai scurtă decît precedenta, prelucrează 20 de valori, în timp ce cea anterioară prelucrează numai 4. Utilizarea variabilelor indexate conferă programului mai multă elasticitate.

Atenție: a nu se confunda A3, de exemplu, cu A(3); ar fi o eroare prea gravă.

4.11.2 — Vectori și matrici

Am văzut în paragraful anterior, care este semnificația unei liste de valori. Generalizînd, putem spune că o listă este o tabelă de valori cu o singură dimensiune. Limbajul BASIC poate prelucra și tablouri de valori cu mai multe dimensiuni. Numele utilizat în matematică pentru a desemna tablourile de valori cu una și două dimensiuni, sînt vectorii și matricele.

În cadrul unei matrice (un tablou cu două dimensiuni) avem întotdeauna un număr oarecare de linii (de vectori) fiecare dintre acestea conținînd un anumit număr de coloane. Să presupunem că avem, de exemplu, o matrice A cu 5 linii de cîte 3 coloane. Vom declara această matrice astfel:

```
10 DIM A(5, 3)
```

Primul număr dintre paranteze indică numărul de linii iar al doilea numărul de coloane. Iată o reprezentare grafică a acestei matrici:

Linii ↓ / Coloane →	1	2	3
1	A(1,1)	A(1,2)	A(1,3)
2	A(2,1)	A(2,2)	A(2,3)
3	A(3,1)	A(3,2)	A(3,3)
4	A(4,1)	A(4,2)	A(4,3)
5	A(5,1)	A(5,2)	A(5,3)

Această matrice, sub forma ei canonică, poate fi scrisă $A(i, j)$ cu $1 \leqslant i \leqslant 5$, $1 \leqslant j \leqslant 3$ (i se confundă foarte ușor cu 1).

Să facem un exercițiu simplu pentru a înțelege mai ușor modul de manipulare al matricilor. Următoarea tabelă de valori înfățișează cantitățile de bunuri de consum vîndute în ultimele trei luni ale anului.

Luna	Produsul: A	B	Tabela T (vînzări) C
octombrie	25	12	30
noiembrie	34	10	42
decembrie	52	16	24

Astfel, $T(1,1) = 25$; $T(1,2) = 12$; $T(2,1) = 34$; etc.

În continuare, vom prezenta un program care lucrează cu o matrice de 12 linii de cîte 5 coloane. Acest program gestionează planul sortimental al vînzărilor a cinci produse în ultimele 12 luni. Programul va cere introducerea valorilor, apoi va afișa tabloul complet. Din cauză că se lucrează în două dimensiuni, s-au introdus variabilele i pentru luni și j pentru produse:

Încercați să înțelegeți bine acest program ajutîndu-vă cu comentariile care urmează:

- în 10 se rezervă memorie pentru matrice;
- în 20 și 30 se cere numărul real al produselor și lunilor (pentru ca programul să cîștige în generalitate);
- în 40 se cere introducerea cantităților din fiecare produs vîndute lunar;
- în linia 60 se ține evidența produselor;
- linia 105 regrupează o secvență de trei instrucții:

```
105 FOR j = 1 TO P
106 PRINT „PRODUS"; j,
107 NEXT j
108 PRINT
```

De notat virgula de la sfîrșitul liniei 106. Ea semnifică afișarea pe coloane normalizate de 16 caractere.

```
 1o  DIM  T(12,5)
 2o  INPUT  "NUMARUL LUNILOR" ; M
 3o  INPUT  "NUMARUL PRODUSELOR" ; P
 4o  FOR  i = 1  TO M
 5o  PRINT  "INTRODUCETI" ; P ; "CANTITATI PENTRU LUNA"; i
 6o  FOR j=1 TO P
 7o  INPUT T(i,j)
 8o  NEXT j
 9o  NEXT i
1oo  PRINT  "TABEL RECAPITULATIV"
1o3  PRINT  ". . . . . ."
1o5  FOR j=1 TO P : PRINT "PRODUS"; j : NEXT j : PRINT
12o  FOR i=1 TO M
13o  FOR j=1 TO P
14o  PRINT T(i,j)
15o  NEXT j
16o  PRINT
17o  NEXT i
18o  END
```

Secvență de introducere a datelor

Secvență de redactare a tabelei

Fig. 4.11.2.1

```
RUN
NUMAR DE LUNI? 3
NUMAR DE PRODUSE? 3
INTRODUCETI 3 CANTITATI PENTRU LUNA 1
? 25
? 12
? 30
INTRODUCETI 3 CANTITATI PENTRU LUNA 2
? 34
? 10
? 42
INTRODUCETI 3 CANTITATI PENTRU LUNA 3
? 52
? 16
? 24
TABEL RECAPITULATIV
PRODUS 1        PRODUS 2        PRODUS 3
  25              12              30
  34              10              42
  52              16              24
```

O chestiune de stil:

decalarea secvențelor de instrucții în funcție de apartenența lor la un ciclu FOR ... NEXT, face programul mai lizibil. Veți aprecia în mod cert această comoditate.

Exerciții

1) Expresiile A3 și A(3) au aceeași semnificație? A — da; B — nu.

2) Este necesară declararea fiecărei liste de valori, împreună cu lungimea sa? A — da; B — nu.

3) Linia 10 DIM Z(99) realizează: A — tabulare; B — declararea unei liste; C — indică lungimea programului.

4) Putem scrie pe aceeași linie: 10 DIM X(25), DIM Z(25) A — da; B — nu.

5) Instrucția: 10 DIM Z(33, 66) definește: A — un tablou cu două dimensiuni; B — două liste de lungimi diferite.

6) Instrucția următoare este corectă? 10 DIM M(5, 33, 66, 87) A — da; B — nu.

7) Care variabilă este identificată prin X(3, 21)? A — cea din coloana 3, linia 21; B — cea din linia 3, coloana 21; C — cea din coloana 21, linia 3.

Răspunsuri

1 = B; 2 = A; 3 = B; 4 = B; 5 = A; 6 = A; 7 = B = C.

4.12 — LOGICĂ ŞI MATEMATICĂ

Prezentăm în acest capitol cîteva noţiuni complementare, necesare programatorului în limbajul BASIC. Vom examina succesiv notaţiile % şi # afectate variabilelor, conversiile în octal şi hexazecimal, împărţirea numerelor întregi, operaţiile modulo operaţiile logice (AND, OR, NOT etc.), operaţiile trigonometrice, funcţiile de bibliotecă (RND, LOG, ABS, FIX, SIGN etc.), făcînd astfel o trecere în revistă rapidă, dar totuşi utilă, a unor probleme considerate în genere ca fiind ingrate.

4.12.1. — Complemente matematice asupra numerelor

Numerele sînt reprezentate, pe mediul extern, în notaţie zecimală obişnuită sau în notaţie ştiinţifică. Astfel, valoarea zecimală 0.0001 este echivalentă cu 1E-4 în notaţie ştiinţifică. În cazul anumitor interpretoare BASIC, putem preciza anumite caracteristici ale variabilelor; astfel:

A sau A! reprezintă o variabilă în simplă precizie;
A% este o variabilă întreagă;
A# este o variabilă în dublă precizie;

(A nu vă lăsa impresionaţi de caracterul „diez" (#) utilizat în muzică. După puţină practică în BASIC vă veţi obişnui cu el.)

Putem dispune în unul şi acelaşi program de următoarele variabile: A, A%, A#, A$

Simplă şi dublă precizie:

fiecare maşină prelucrează numere într-un format de bază, bine definit, în binar, numit „simplă precizie". Toate interpretoarele BASIC utilizează o notaţie în virgulă flotantă în simplă precizie. Aşa cum s-a arătat, un întreg este un număr de maxim cinci cifre a cărui valoare este cuprinsă între −32768 şi +32767.

Un număr real (în virgulă flotantă) în simplă precizie are o valoare cuprinsă între:

−1.701411 E + 38 şi +1.701411 E + 38

Un număr real (în virgulă flotantă) în dublă precizie este un număr cuprins între:

−1.701411834544556 E + 38 şi
+1.701411834544556 E + 38

4.12.2 — Octal şi hexazecimal

Sistemele de numeraţie în baza 8 (octal) sau 16 (hexazecimal) sînt utilizate în special de către limbajele de asamblare. În mod evident, limbajul BASIC nu poate nici el ignora aceste sisteme de numeraţie care se disting prin prefixul „&". Astfel „&" indică o valoare octală (de exemplu &15), iar „&H" indică o valoare hexazecimală (de exemplu: &H7F). Dacă nu sînteţi familiarizaţi cu aceste sisteme de numeraţie, cereţi interpretorului BASIC să traducă valorile de mai sus, în zecimal:

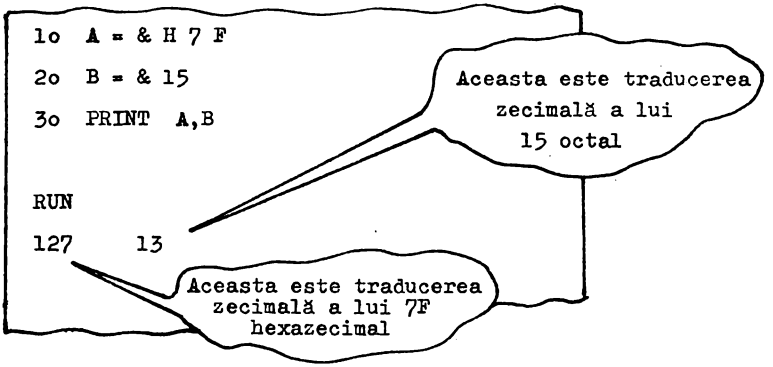


Fig. 4.12.2.1

HEX$** O altă metodă de conversie a constantelor dintr-un **OCT$ sistem de numeraţie într-altul, este aceea de a folosi ordinele HEX$ şi OCT$.

Următorul program generează un tablou de conversie al numerelor întregi (notate cu %):

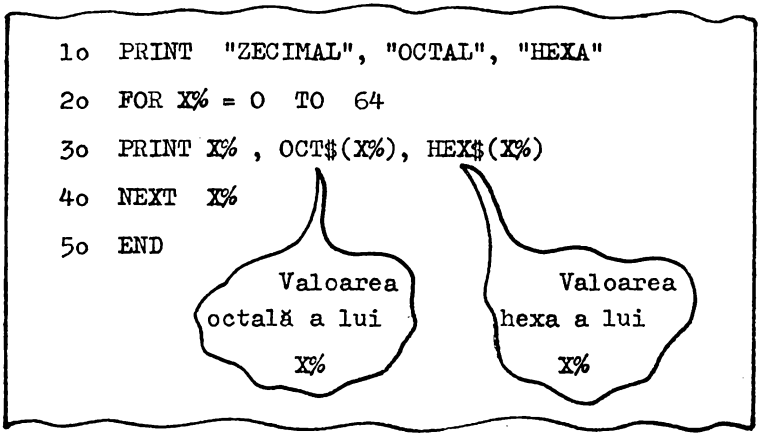


Fig 4.12.2.2

4.12.3 — Împărțirea numerelor întregi

/ Împărțirea este notată, după cum vă amintiți, prin simbolul „bară oblică" (/). Uneori, puteți dispune de simbolul „bară oblică inversă" (\) care servește, în mod exclusiv, la împărțirea întregilor, avantajul utilizării acestui operator rezidă în faptul că operația astfel executată este mai rapidă.

4.12.4 — Operații modulo

MOD O operație modulo furnizează drept rezultat un rest. De exemplu, rezultatul împărțirii 9 modulo 4 furnizează restul 1. Operatorul utilizat în operații modulo este MOD (de exemplu: LET C = A MOD B).

4.12.5 — Operații logice

Operațiile logice fac să intervină operatorii logici: ȘI, SAU, SAU-EXCLUSIV, NU. Celor nefamiliarizați cu acești operatori logici, exemplele următoare li se vor părea puțin cam ermetice.

AND Acesta este operatorul BASIC pentru funcția logică SI. Iată un program:

```
10 INPUT A, B
20 IF A = 3 AND B = 7 THEN 40
3 STOP
40 PRINT „CONDITIE SATISFACUTA"
50 END
```

Dacă introducem valorile 3 și 7 pentru A și B, cele două variabile vor fi verificate, iar linia 20 va trimite la linia 40.

OR Operația SAU poate fi explicată de următoarea secvență:

```
10 INPUT A, B
20 IF A = 9 OR B = 27 THEN 40
30 STOP
40 PRINT „CONDITIE SATISFACUTA"
50 END
```

Este suficient ca una dintre variabilele *A* și *B* să aibă una dintre valorile 9 și 27, pentru ca linia 20 să trimită la 40.

NOT Pentru a examina acțiunea operatorului NU, care inversează condiția, vom modifica programul de mai sus, astfel încît să obținem același rezultat:

```
10 INPUT A, B
20 IF NOT (A = 9 OR B = 27) THEN 40
30 PRINT „CONDITIE NESATISFACUTA"
35 STOP
40 PRINT „CONDITIE SATISFACUTA"
50 END
```

4.12.6 — Funcții trigonometrice

Da, aceste funcții există și ele în limbajul BASIC, indiferent de aversiunea sau amiciția pe care o aveți pentru ele. Astfel veți putea calcula SIN(X), COS(X), TAN(X) pentru sinus, cosinus și tangentă de X, și chiar ATN(X) pentru arc-tangentă de X. Nu vom insista cu mai multe amănunte în legătură cu aceste funcții trigonometrice.

4.12.7 — Alte funcții

Am întîlnit deja funcția SQR(N) care calculează rădăcina pătrată a numărului N. De asemenea am întîlnit funcția RND(X) asupra căreia revenim pentru un moment:

RND În BASIC standard, funcția RND(X) generează un număr aleatoriu cuprins între 0 și 1; dacă X este mai mic ca zero, RND(X) furnizează numărul aleator următor în secvență; dacă $X = 0$, se reia valoarea precedentă; în sfîrșit, dacă X este mai mare decît zero, o nouă secvență de numere aleatoare este lansată.

Există însă și variante particulare de implementare ale limbajului BASIC, în care RND(X) furnizează un număr aleator cuprins între 0 și X. De aici reiese necesitatea de a studia specificația de utilizare a interpretorului BASIC, care însoțește ordinatorul pe care îl folosiți.

LOG(X) Această funcție determină logaritmul natural (în baza e, adică 2,71828) al lui X. Pentru a trece la logaritmul în baza 10 se folosește formula clasică (pentru X mai mare decît zero):

$$A = LOG(X) * .4343$$

EXP(N)
ABS(X)
CINT(X)
CSNG(X)
CDBL(X)
INT
FIX(N)
SGN(X)

Funcția EXP(N) ridică numărul e = 2.71828 la puterea *N*. Comanda ABS(X) furnizează valoarea absolută a lui *X*, adică $\sqrt{X^2}$. Putem, de asemenea, extrage partea întreagă din *X* utilizînd CINT(X). Funcțiile CSNG(X) și CDBL(X) convertesc variabila *X* în simplă precizie, respectiv dublă precizie. Funcția INT care rotunjește valoarea unui număr, a fost deja explicată. Ordinul FIX are aproape același efect, cu deosebirea că în timp ce INT(−63.2) = −64, FIX (−63.2) = −63.

Este cazul să încheiem, pentru a vă permite să vă trageți sufletul, cu funcția SGN(X) care întoarce valoarea −1, dacă argumentul X este negativ, 0 dacă X = 0, +1, dacă X este pozitiv, nenul. Ce simplu este!

Amintiți-vă că dacă aceste funcții nu vă sînt suficiente, puteți să definiți propriile voastre funcții, utilizînd ordinul DEF FN pe care l-am studiat. De asemenea, veți descoperi funcții ale interpretorului BASIC cu care lucrați, care nu sînt prezentate aici.

EXERCIȚII

1) Următoarea secvență este corectă?

```
10 FOR N% = 1 TO 9 STEP .5
20 PRINT N
30 NEXT N
```

A — da; B — nu.

2) Ce va produce operația 29 MOD 5? A — 5; B — 5.8; C — 5.80; D — 4; E — 8.

3) Ce se va afișa la execuția următorului program?

```
10 A = 7
20 IF NOT (A = 7) THEN 50
30 PRINT „DA"
40 STOP
50 PRINT „NU"
60 END
```

A — da; B — nu.

Răspunsuri: 1 = B; 2 = D; 3 = A.

4.13 — FUNCŢII COMPLEMENTARE

În acest ultim capitol al primei secţiuni, vom examina cîteva instrucţii complementare ale limbajului BASIC, cum ar fi ordinele: AUTO, PEEK, POKE, cîteva ordine suplimentare de prelucrare a şirurilor de caractere: STR $, VAL, SPACE $, cît şi instrucţiile de prelucrare grafică: DRAW, WINDOW, VIEWPORT, LABEL, RMOVE, RDRAW, SCALE, ROTATE etc.

4.13.1 — Numerotarea automată a liniilor

AUTO Există interpretoare BASIC care vă scutesc de grija numerotării liniilor. Este suficient să comandaţi AUTO, pentru ca numerotarea liniilor să înceapă de la 10, incrementîndu-se automat cu 10 la fiecare caracter CR, sau ENTER. Dacă nu doriţi ca numerotarea liniilor să înceapă cu 10, ci cu 200, de exemplu, nu aveţi decît să comandaţi AUTO 200. Dacă, în plus, incrementul 10 nu vă satisface, veţi scrie, de exemplu, AUTO 200, 20 şi incrementul va fi 20. Comanda BREAK poate fi utilizată pentru a sparge această secvenţă.

4.13.2 — Instrucţii suplimentare asupra şirurilor de caractere

Există numeroase operaţii care se aplică, de cele mai multe ori, asupra numerelor, dar care se utilizează adeseori şi asupra şirurilor de caractere. Astfel instrucţia:

```
10 IF A$ = „HORIA“ AND B$ = „DUMITRASCU“ THEN 50
```

reprezintă un exemplu edificator de utilizare a operaţiilor logice asupra şirurilor de caractere. Să ne amintim, de asemenea, că putem concatena şiruri de caractere, că putem, adică, să le punem cap la cap.

```
10 A$ = „HORIA“
20 B$ = „DUMITRASCU“
30 PRINT A$ + B$
RUN
HORIA DUMITRASCU
```

Pe de altă parte, numerele pot fi considerate şiruri de caractere de exemplu:

A$ =„1984“

VAL Dacă dorim să restabilim forma numerică a unui şir de caractere vom cere valoarea acestui şir

```
10 A$ = „1984"
20 A = VAL (A$)
30 PRINT A + 17
RUN
2001
```

STR$ Este operaţia inversă lui VAL.

```
1o  A$ = STR$ (1984)
2o  PRINT  A$
RUN
1984
```

Diferenţa este foarte mică dar 1984 este considerat, la afişare, ca un şir de caractere, nu ca un număr întreg

Fig. 4.13.2.1

**SPACE$** Putem spaţia un text folosind această instrucţie. SPACE$ (1) furnizează un şir de 1 spaţii.

```
10 A$ = „LA"
20 B$ = „REVEDERE"
30 FOR X = 1 TO 5
40 PRINT A$; SPACE$(X); B$
50 NEXT X
60 END
RUN
LA REVEDERE
LA  REVEDERE
LA   REVEDERE
LA    REVEDERE
LA     REVEDERE
```

4.13.3 — Instrucţii asupra codurilor ASCII

Pentru a putea fi transmis, fiecare caracter al tastaturii este codificat; codul cel mai uzual este codul ASCII, prescurtare a lui American Standard Code for Information Interchange.

ASC În orice moment putem obține codul ASCII al oricărui caracter, utilizînd această funcție care furnizează codul atașat primului caracter al șirului menționat.

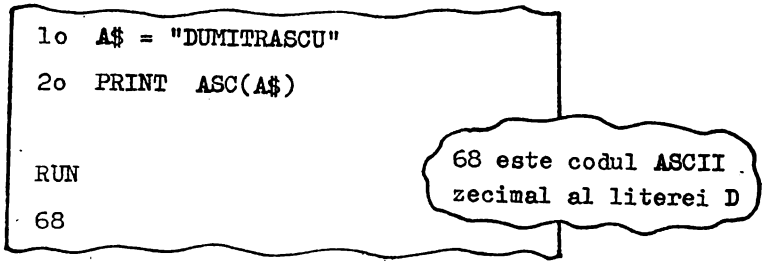


Fig. 4.13.3.1

CHR $ Această comandă este inversa lui ASC, furnizînd caracterul corespunzător codului ASCII menționat în comandă.

```
10 PRINT CHR$(69); CHR$(86); CHR$(65)
RUN
EVA
```

```
69 = E
86 = V
65 = A
```

4.13.4 — PEEK și POKE

Utilizînd limbajul BASIC, utilizatorul are posibilitatea să ignore conținutul fiecărei locații de memorie mașinii sale. Cu toate acestea, în anumite situații utilizatorul poate fi interesat să depună o valoare la o anumită adresă de memorie, sau să examineze o locație de memorie cu o adresă egală cu cea specificată. În aceste situații intervin ordinele PEEK și POKE.

PEEK De exemplu, instrucția PEEK(234) ordonă citirea locației de memorie nr. 234. Putem avea deci

```
10 A = PEEK (234)
```

Aici variabila A primește valoarea înscrisă în memorie la adresa 234.

POKE Dimpotrivă, ordinul POKE este destinat operației de scriere în memorie, la o adresă specificată, a unei valori numerice.

```
10 POKE (7654,65)
```

Această instrucție depune valoarea 65 în locația de memorie de adresă 7654. Putem scrie la fel de bine

```
10 A = 7654
20 B = 65
30 POKE (A, B)
```

și secvența va avea același efect.

4.13.5 — Instfucții de prelucrare grafică

Soluția multor probleme este compusă dintr-un număr impresionant de valori, ceea ce face ca interpretarea lor să fie extrem de dificilă. Reprezentînd grafic această mulțime de valori, vom obține o soluție cu mult mai inteligibilă. Acesta este cel mai la îndemînă exemplu al utilizării graficii în procesul de prelucrare automată a datelor. În afară de acesta, putem enumera încă cîteva dintre cele mai noi aplicații ale graficii interactive: știință și tehnologie, cartografie, proiectare asistată de calculator, construcție și fabricație asistată de calculator, simulare și animație, conducerea automată a proceselor tehnologice, publicistica electronică, artă și comerț. Avînd în vedere multitudinea de aplicații ale graficii cu calculatorul, este deci firesc ca limbajul BASIC, ca unul dintre limbajele de cea mai mare circulație, să posede un puternic set de instrucții de prelucrare grafică. Vom încerca, în cele ce urmează, să facem o scurtă introducere în grafica interactivă, utilizînd, spre exemplificare, setul de instrucții grafice ale limbajului BASIC-118 implementat pe microcalculatorul FELIX M-118.

MOVE Această instrucție este utilizată pentru poziționarea cursorului display-ului grafic, în punctul de coordonate *X*, *Y*. În realitate, — însă, poziționarea nu are loc decît din punct de vedere logic, întrucît cursorul nu este vizibil în poziția de coordonate *X*, *Y*.

```
30 MOVE 100, 75
```

Această instrucție realizează poziționarea cursorului în punctul (100, 75) pe ecranul grafic.

Orice ecran grafic se caracterizează printr-un număr de puncte distincte ce pot fi reprezentate. La microcalculatorul FELIX M-118, acest număr de puncte este 512×256.

Coordonatele X și Y ale punctului referit pot fi exprimate sub forma unor expresii calculabile.

```
10 X = 25
20 Y = 150
30 MOVE X*4, Y/2
```

DRAW Această instrucție este utilizată pentru a trage o linie între punctul în care se află poziționat cursorul (efect al instrucției MOVE) și punctul de coordonate X, Y specificat. Următorul exemplu realizează reprezentarea grafică a datelor conținute în vectorul V.

```
10 DIM V(20)
20 MAT INPUT V
25 MOVE 1, V(1)
30 FOR I = 1 TO 20
40 DRAW I, V(I)
50 NEXT I
60 END
```

Instrucția din linia 25 realizează poziționarea cursorului în punctul (1, $V(1)$), apoi cu ajutorul instrucției DRAW, executată în ciclu, se vor uni complementele vectorului.

Originea suprafeței grafice se consideră a fi punctul din stînga jos.

RMOVE și RDRAW Instrucțiile RMOVE și RDRAW se deosebesc de instrucțiile MOVE și DRAW prin faptul că punctul (X, Y) pe care îl referă nu este exprimat în valori absolute, raportat la originea ecranului, ci este exprimat incremental, raportat la poziția cursorului în momentul execuției instrucției. Aceste două instrucții sînt extrem de utile la scrierea subrutinelor ce desenează figuri în diverse zone ale suprafeței grafice. Prin utilizarea instrucției MOVE înainte de apelul subrutinei se poate realiza translația figurii.

WINDOW Așa cum am arătat, orice dispozitiv grafic se caracterizează printr-un număr de puncte distincte ce pot fi reprezentate, totalitatea acestor puncte alcătuind „spațiul grafic". Atributele principale ale spațiului grafic sînt: precizia (numărul de puncte din care este alcătuit spațiul grafic) și rezoluția (distanța dintre două puncte ale spațiului grafic). Instrucțiile de

prelucrare grafică referă puncte ale spațiului grafic prin coordonatele lor (X, Y), exprimate în unități de măsură naturale, corespunzătoare mărimilor reprezentate grafic (metri, kilograme, volți etc.). Totalitatea acestor valori alcătuiesc „spațiul virtual". Spațiul virtual este practic limitat doar de precizia aritmeticii mașinii (1 E-63, 8 E + 68). Spațiul virtual va putea fi reprezentat la o anumită scară, în cadrul spațiului grafic. Instrucția WINDOW permite utilizatorului să definească limitele spațiului virtual.

```
50 WINDOW A, B, C, D
```

unde A, B, C, D sînt valori numerice ce reprezintă limitele spațiului virtual, respectiv limita stîngă, dreaptă, inferioară, superioară. Un punct de coordonate (X, Y) va putea fi reprezentat pe suprafața grafică dacă următoarele combinații sînt verificate: $A \leqslant X \leqslant B$, $C \leqslant Y \leqslant D$.

În timpul execuției unui program grafic, limitele spațiului virtual sînt definite astfel:
WINDOW −1E−64, 8E 68, −1E−64, 8E 68

VIEWPORT Dacă instrucția WINDOW este destinată pentru limitarea spațiului virtual, limitarea spațiului grafic se poate face utilizînd instrucția VIEWPORT.

```
10 VIEWPORT A, B, C, D
```

Cei patru parametri au aceeași semnificație ca în cadrul instrucției WINDOW. Spre deosebire de WINDOW, unitățile în care sînt exprimate limitele spațiului grafic sînt unități fizice. Alegerea unității fizice în care să se exprime limitele trebuie să satisfacă cerințele de independență față de tipul perifericului grafic (exemplu: unele dispozitive grafice au suprafața pătrată, altele dreptunghiulară).

Independența programului față de tipul dispozitivului grafic are meritul de a-i asigura portabilitatea.

Unitatea în care se exprimă limitele A, B, C, D s-a ales, ca fiind egală cu 1% din latura pătratului cel mai mare ce poate fi înscris în suprafața dispozitivului grafic. Această unitate va fi în continuare numită „unitate grafică" sau prescurtat: UG.

În aceste condiții, instrucția

```
10 VIEWPORT 0, 100, 0, 100
```

va specifica suprafața celui mai mare pătrat înscris în suprafața grafică. Această instrucție se execută implicit la inițializarea sistemului. Tot implicit se execută și instrucția:

```
10 WINDOW 0, 100, 0, 100
```

care realizează o corespondență biunivocă între spațiul virtual și spațiul grafic, sau, altfel spus, între mulțimea de valori reprezentate și UG.

Cu aceste inițializări implicite, un cerc va apărea nedistorsionat, deoarece pe ambele axe de coordonate se folosesc aceleași UG, iar suprafața grafică este pătrată.

Următorul program va trasa un cerc pe orice periferic grafic:

```
10 DIM C(2)
20 MAT INPUT „Introduceti coordonatele centrului", C
30 INPUT „Introduceti dimensiunea razei", R
40 INIT P
50 MOVE C(1), C(X2)
60 I = C(1) − C(2)
70 FOR I = 0 TO 20
80 M = I*PI/10
90 DRAW I*COS(M) + C(2), I*SIN(M) + C(2)
100 NEXT I
110 END
```

Coordonatele centrului și lungimea razei se introduc în timpul execuției programului. Dacă se dorește reprezentarea cercului în colțul din dreapta sus, atunci este necesară introducerea instrucției:

```
45 VIEWPORT 75, 100, 75, 100
```

LABEL Cu ajutorul acestei instrucții se pot scrie mesaje și comentarii pe suprafața grafică, începînd din punctul în care este poziționat cursorul. Instrucția are drept parametru opțional factorul de scală cu care se reprezintă textul. Acest factor de scală este o valoare numerică cuprinsă între 1 și 34. Valoarea 1 corespunde scrierii cu caractere normale, iar valoarea 34 va genera caractere de dimensiunea întregului ecran.

```
10 INIT P
20 MOVE 10, 40
30 LABEL SCL(4) „I.T.C. Bucuresti"
```

Această secvență de program va edita, începînd cu punctul (10, 40) textul „I.T.C. București", utilizînd scara de reprezentare 4.

SCALE Instrucția SCALE este utilă atunci cînd se dorește editarea de cartografii, diagrame, desene, la diferite scări de reprezentare.

```
10 SCALE S1, S2
```

S1 și S2 sînt expresii ce reprezintă factorii de scară pe orizontală și verticală. Factorul de scară se determină prin raportul: S = Unități utilizator/1 UG.

Instrucția SCALE este echivalentă cu o instrucție WINDOW în care parametrii A și C sînt înlocuiți cu coordonatele punctului de origine față de care se dorește scalarea.

ROTATE Această instrucție are efect numai asupra instrucțiilor RHOVE și RDRAW, realizînd rotația cu unghiul a cărui măsură, în radiani, este egală cu valoarea specificată în instrucție. Dacă instrucția ROTATE este utilizată înaintea unei rutine ce generează o figură utilizînd RMOVE și RDRAW, atunci figura va apare rotită cu unghiul specificat. Punctul de origine al rotației este punctul în care se află cursorul suprafeței grafice.

Să considerăm un exemplu. Fie următoarea subrutină:

```
100 RDRAW 10,0
110 RDRAW 0,10
120 RDRAW -10,0
130 RDRAW 0, -10
140 RETURN
```

Această rutină desenează un pătrat. Dacă rutina va fi apelată de următoarea secvență:

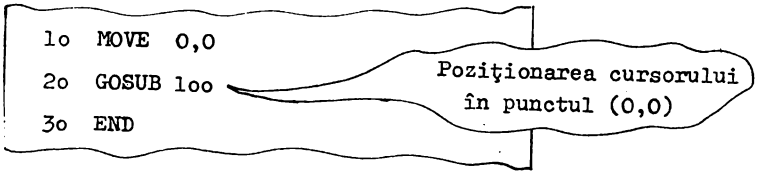


Fig 4.13.5.2

atunci pătratul va apărea în colțul din stînga jos. Dacă dorim să desenăm un pătrat de două ori mai mare decît cel precedent, atunci putem apela astfel rutina de mai sus:

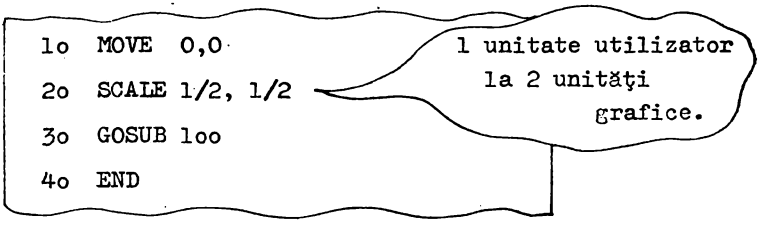


Fig 4.13.5.3

Dacă dorim ca pătratul să fie desenat în centrul ecranului și rotit cu 45°, în sens trigonometric, vom apela astfel rutina:

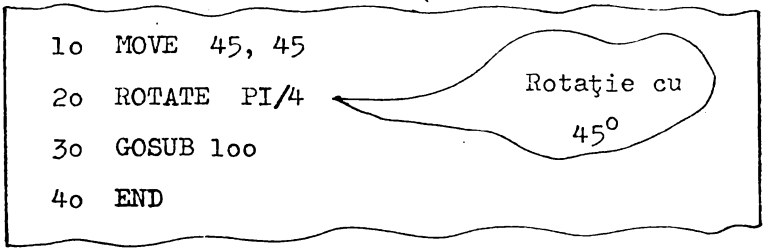


Fig 4.13.5.4

Instrucțiile MOVE și DRAW nu sînt afectate de rotație deoarece coordonatele specificate de aceste instrucții sînt absolute.

GSINPUT Utilizarea acestei instrucții asigură independența totală față de tipul dispozitivului grafic.

```
100 GSINPUT A, B.
110 VIEWPORT 0, A, O, B
```

Execuția instrucției din linia 100 constă în atribuirea variabilelor V1, V2 a dimensiunilor suprafeței grafice, exprimate în UG. De exemplu un plotter poate avea suprafața dreptunghiulară, cu latura orizontală mai mare decît cea verticală. Utilizînd variabilele V1 și V2 în instrucția VIEWPORT, se pot obține programe care utilizează întreaga suprafață a plotterului, indiferent de forma sa.

AXIS Această instrucție este utilizată pentru trasarea axelor pe suprafața grafică. În cazul în care pe nici una din axe mărimile nu iau valori de ambele semne, axele vor fi trasate astfel încît să se intersecteze în colțul din stînga jos al suprafeței grafice. Instrucția are doi parametri opționali, care reprezintă unitățile de măsură pe axele *OX*, respectiv *OY*, și, în cazul cînd apar în instrucțiune, axele vor fi marcate. Pentru ca valorile de pe grafic să fie notate, trebuie folosită instrucția LABEL.

INIT Instrucția INIT este utilizată pentru ștergerea ecranului display-ului grafic și trecerea lui în mod pagină (argument *P*), sau în mod defilare (argument *S*). INIT P trebuie să fie prima instrucție grafică dintr-un program. Această instrucție face și poziționarea cursorului în punctul (0, 0). La terminarea unui program ce utilizează instrucții grafice, trebuie folosită instrucția INIT S.

INIT S va șterge ecranul și imaginea se va pierde. Se poate da INIT S după terminarea programului și studierea graficului, în mod imediat.

Exerciții

1) Ce va produce execuția următoarei secvențe?

```
10 A$ = „20"
20 B$ = „01"
30 PRINT A$ + B$
```

A — 21; B — 20 + 01; C — 2001.

2) Care este instrucția care permite regăsirea unui caracter sau subșir într-un șir complet? A = MID$; B = INSTR; C = LEN

3) Un număr poate fi considerat ca un șir de caractere? A — da; B — nu.

4) Ce va produce execuția instrucției următoare?

```
10 PRINT „**"; SPACE$(2); „**"
```

A — ****; B — * * * *; C — ** **

5) Dacă locația de memorie de adresă 317 conține valoarea 32, ce va conține după execuția următoarei instrucții:

```
10 A = PEEK (317)
```

A — 32; B = 0; C = o valoare aleatoare.

6) Aceeași problemă după execuția instrucției:

```
20 POKE (317,81)
```

A — 32; B — 106; C — 81.

7) Putem scrie:

```
30 POKE (1317,432)
```

A — da; B — nu.

Răspunsuri: 1 = C, 2 = B, 3 = A, 4 = C, 5 = A, 6 = C, 7 = B.

4.14 — INSTRUCȚII DE CALCUL CU MATRICI

Interpretoarele BASIC implementate pe calculatoarele românești (BASIC-AMS, BASIC-18, BASIC-118) conțin un set de instrucții care permit prelucrarea tablourilor bidimensionale ca matrici. Sînt disponibile următoarele instrucții matriciale: MATREAD, MATINPUT, MATPRINT, care realizează respectiv citirea matricilor cu READ sau INPUT și tipărirea lor cu PRINT. De asemenea, există posibilitatea efectuării unor prelucrări matriciale, ca de exemplu: calculul inversei unei matrici, calculul transpusei unei matrici, calculul produsului a două matrici, calculul produsului unei matrici cu un scalar, calculul sumei și diferenței a două matrici, inițializarea matricilor.

MATREAD Această instrucție permite citirea unei matrici prin READ. Fie următoarea secvență:

```
 5 DIM B(3, 3), A(3)
10 MATREAD B(2, 3), A(3)
20 STOP
30 DATA 5, 11, -17, 1, 2, 3, 1 E 7, 0, 1
40 END
```

După execuția acestui program matricile A și B vor conține: $A(1) = 10^7$; $A(2) = 0$; $A(3) = 1$; $B(1, 1) = 5$; $B(1, 2) = 11$; $B(1, 3) = -17$; $B(2, 1) = 1$; $B(2, 2) = 2$; $B(2, 3) = 3$.

În exemplul de mai sus instrucția DIM nu era absolut necesară, matricile putînd fi alocate în momentul întîlnirii ordinului MATREAD.

MATINPUT Toate regulile și observațiile de mai sus sînt valabile și pentru această instrucție. Deosebirea constă în faptul că datele sînt introduse de la terminal.

MATPRINT Tipărirea matricilor se face linie cu linie. În cadrul liniei spațierea între elemente se face conform separatorului utilizat în lista de intrare-ieșire (separatorul poate fi „," sau „;"). După tipărirea fiecărei linii a matricii se lasă o linie vidă pentru ca imaginea să cîștige în claritate. Dacă același MATPRINT afișează mai multe matrici, atunci între ultima linie a unei matrici și prima linie a matricii următoare se lasă două linii vide. Să considerăm următorul exemplu:

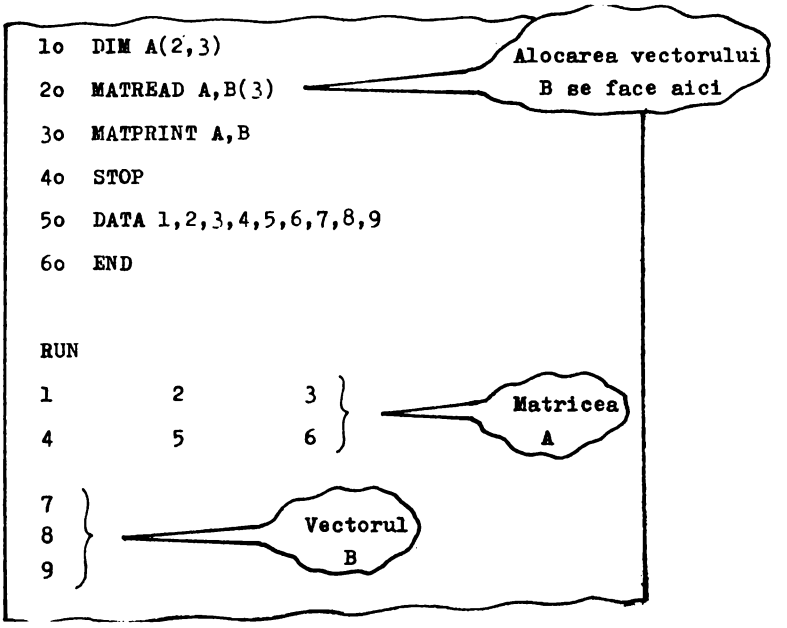


Fig. 4.14.1

INV Orice matrice pătrată nesingulară poate fi inversată. De exemplu următoarea instrucție:

```
10 MAT A = INV(B)
```

are drept efect inversarea matricei pătrate, nesingulare B și atribuirea valorilor astfel obținute, matricii A. Dacă se dorește și calculul determinantului matricii B, atunci se poate scrie:

```
10 MAT A = INV(B), C
```

În acest caz, variabila C va primi o valoare egală cu determinantul matricii B.

O matrice poate fi inversată chiar în ea însăși. De exemplu, instrucția:

```
10 MAT A = INV(A)
```

este corectă.

TRN Transpusa unei matrici $A(m, n)$ este matricea $A(n, m)$. De asemenea, un vector coloană are drept transpusă un vector linie. Fie următorul exemplu:

```
1o  MATINPUT  V(4) ;
2o  MATPRINT  TRN(V)
3o  END

RUN
? 1o8, 2, 56, 4
  1o8,  2,  56,  4
```

Aici caracterul ";" a fost necesar pentru a putea introduce valorile vectorului pe aceeași linie

Aici caracterul ";" nu a mai fost necesar, întrucît prin transpunere vectorul a devenit linie.

Fig. 4.14.2

O matrice nu poate fi transpusă în ea însăși. Instrucția:

```
10 MAT A = TRN(A)
```

este incorectă.

PRODUSUL MATRICILOR Pentru a putea înmulți două matrici, este necesar ca numărul de coloane al primei matrici să fie egal cu numărul de linii al celei de-a 2-a.

Fie următorul program:

```
1o  DIM B(2,3), C(3,4), A(2,4)
2o  MATREAD  B,C
3o  MAT A = B * C
4o  MATPRINT  A
5o  STOP
6o  DATA  15,4,3,2,7,1o
7o  DATA  8,7,5,6,9,1o,2o,15,12,25,6,1
8o  END

RUN
144       141       115
36        39        35
116       139       135
```

Matricea produs are dimensiunile 2 x 4

Fig. 4.14.3

PRODUSUL UNEI MATRICI CU UN SCALAR Prin înmulţirea unei matrici cu un scalar se obţine o matrice rezultat care are dimensiunile matricei produs. De exemplu instrucţia:

```
10 MAT A = (COS(X) + SIN(X))*B
```

va genera în *A* o matrice rezultat ale cărei elemente vor fi egale cu elementele matricii *B*, înmulţite cu expresia dintre paranteze.

ADUNAREA ŞI SCĂDEREA MATRICILOR Nu pot fi adunate sau scăzute decît matricile care au aceleaşi dimensiuni. Cu aceleaşi dimensiuni va fi dimensionată şi matricea rezultat. Matricea din membrul stîng al tribuirii poate figura şi în membrul drept. De exemplu:

```
10 MAT A = A + B
```

Într-o singură instrucţie nu se poate efectua decît o singură operaţie. De exemplu, instrucţia

```
10 MAT A = B + C - D
```

este incorectă.

INIȚIALIZAREA UNEI MATRICI Generarea unei matrici cu toate elementele nule, poate fi obținută cu instrucția:

```
10 MAT A = ZER(3, 2)
```

Dacă matricea A fusese deja alocată înainte de execuția instrucției 10, ea va fi redimensionată cu dimensiunile (3, 2), în caz contrar, va fi alocată cu aceleași dimensiuni.

Pentru a genera o matrice cu toate elementele 1 se va folosi, de exemplu, instrucția:

```
10 MAT A = CON(3, 3)
```

Pentru a se genera o matrice unitate (cu 1 pe diagonala principală) se va folosi funcția IDN. Exemplu:

```
10 MAT A = IDN(3, 3)
```

5. PROGRAME SCRISE ÎN LIMBAJUL BASIC

În continuare sînt prezentate programe scrise în limbajul BASIC, care îndeplinesc diferite funcții de utilitate curentă. Programele sînt scrise în limbajul BASIC standard, compatibil cu cea mai mare parte a versiunilor acestui limbaj disponibil pentru utilizatorii de mini și microcalculatoare.

Fiecare program conține comentarii pentru a-l ajuta pe cititor să urmărească cu ușurință modul de funcționare a programului. Comentariile sînt, în egală măsură, necesare și pentru identificarea secvențelor ce pot fi recuperate și reutilizate în programele pe care cititorul dorește să le creeze. Întotdeauna comentariile vor precede liniile la care se referă. Anumite programe conțin opțiuni. O opțiune este o modificare de natură să schimbe formatul de intrare sau de ieșire al programului. Aceste opțiuni pot, de asemenea, să sugereze modificări ulterioare pe care le puteți aduce programelor. Am inclus, în fiecare program, o descriere sumară, un exemplu de utilizare și o listare parțială a fiecărei variante de utilizare care rezidă din selectarea uneia sau alteia dintre opțiunile posibile. Listingul programului cuprinde instrucțiunile ce trebuie schimbate cu prilejul trecerii de la versiunea originală de

bază la una dintre versiunile opționale. Liniile care trebuie sau pot fi modificate, eliminate sau completate, sînt încadrate într-un chenar.

În legătură cu corectitudinea programelor, în afara erorilor de programare și de tipar, mai există și așa-zisele erori de compatibilitate, mult mai susceptibile de a se produce decît primele. Aceste erori de compatibilitate se referă la diferențele care există între diverse variante de implementare ale limbajului BASIC, diferențe care fac ca același program să fie executat în mod diferit de la un ordinator la altul.

Erorile de compatibilitate pot fi rezolvate după cum urmează:

1) Anumite programe execută ieșiri forțate din cicluri FOR/NEXT. Aceasta înseamnă că, în funcție de anumite condiții, se efectuează o ieșire din buclă, înainte ca numărul de iterații să fie epuizat. Dacă interpretorul BASIC pe care-l utilizați semnalează, în această situație, o eroare, atunci este necesar ca, din momentul detectării condiției de ieșire forțată din ciclu, să efectuați un salt la instrucția NEXT aferentă, și să exilați deci instrucțiile care, modifică variabila de control a ciclului.

2) La terminarea unui ciclu FOR/NEXT variabila de indexare rămîne poziționată la ultima sa valoare. Dacă lucrați cu un BASIC care n-are posibilitatea de a conserva valoarea variabilei în afara ciclului, trebuie ca la ieșirea din ciclu să atribuiți acestei variabile valoarea finală.

3) O variantă a comenzii RESTORE este RESTORE n, unde n reprezintă numărul de ordine, în cadrul instrucției sau instrucțiilor DATA, al constantei numerice ce va fi citită cu următorul READ. Dacă interpretorul dv. nu admite RESTORE n, modificați această parte a programului introducînd un RESTORE obișnuit, apoi efectuați un ciclu pentru a citi, cu ajutorul instrucției READ, primele $n-1$ constante numerice introduse prin DATA.

Dacă detectați vreo eroare sau vreo dificultate de programare care, după părerea dumneavoastră, nu provine dintr-o stîngăcie de utilizare, voi fi bucuros să fiu informat. Vă puteți adresa autorului, prin intermediul editurii, furnizînd următoarele informații:

a. Descrierea erorii.

b. Datele introduse care au provocat eroarea.

c. Listingul sursă al programului dv.

d. Cauza pe care o considerați susceptibilă de a fi produs eroarea.

În ceea ce privește forma de prezentare, fiecare program conține trei categorii de informații:

— logica de prelucrare propriu-zisă, transpusă în limbajul de programare BASIC, cu cuvintele sale rezervate, notațiile și sintaxa specifice;

— descrierea părții de intrare-ieșire care constă în întrebările adresate de către program utilizatorului și în răspunsurile acestuia din urmă;

— comentariile, semnalate printr-o instrucție REM la începutul fiecărei linii și care au ca scop facilitarea înțelegerii.

Forma de prezentare a fiecărui program, va fi deci următoarea:

— mai întîi o explicație care descrie obiectul programului și metodele folosite;

— apoi, unul sau mai multe exemple de utilizare;

— în sfîrșit, listingul sursă al programului; evident înțelegerea sa necesită cunoașterea de către cititor a limbajului BASIC, deci, parcurgerea și însușirea primei părți a cărții.

5.1 — DESCOMPUNEREA UNUI ÎNTREG ÎN FACTORI PRIMI

Acest program determină descompunerea în factori primi a unui întreg. Nu funcționează pentru numărul întreg 0. Exemplul următor prezintă modul de funcționare al programului, avînd ca date de intrare numerele: —49 și 92.

```
: RUN
DESCOMPUNEREA IN FACTORI PRIMI A UNUI
INTREG (INTRODUCETI 0 PT TERMINARE
PROGRAM)
NUMARUL ? — 49
— 1
7↑ 2
NUMARUL ? 92
1
2↑2
23↑1
NUMARUL? 0
END PROGRAM AT LINE 200
```

```
10  PRINT „Descompunerea in factori primi a unui intreg."
20  PRINT
30  PRINT „(Introduceti 0 pt terminare program.)"
40  PRINT „Numarul"
50  INPUT Z
59  REM — Sfirsit program ? —
60  IF Z = 0 THEN 200
69  REM — Semnul numarului este tot un factor —
70  PRINT SGN(Z)
79  REM — Utilizeaza valoarea absoluta in calcule —
80  Z = ABS(Z)
89  REM — Testeaza toti intregii intre 2 și Z —
90  FOR I = 2 TO Z
100 S = 0
110 IF Z/I<>INT(Z/I) THEN 150
120 Z = Z/I
130 S = S + 1
140 GOTO 110
150 IF S = 0 THEN 170
158 REM — Afisare factor prim cu exponent —
159 REM — I^S = I la puterea S
160 PRINT I; "^"; S
170 NEXT I
180 PRINT
189 REM — Restartare program —
190 GOTO 40
200 END
```

5.2 — ARIA UNUI POLIGON

Acest program calculează aria unui poligon. Pentru aceasta este necesar să se indice coordonatele X și Y ale tuturor vîrfurilor. Aceste coordonate trebuie indicate în ordinea vîrfurilor succesive. Formula de calcul utilizată este:

$$\text{Aria} = \frac{(X_1+X_2)\cdot(Y_1-Y_2)+(X_2+X_3)\cdot(Y_2-Y_3)+\ldots+(X_n+X_1)\cdot(Y_n-Y_1)}{2}$$

unde n este numărul de vîrfuri.

În varianta prezentată, programul lucrează pentru un număr de vîrfuri mai mic sau egal cu 24. Pentru a modifica numărul de vîrfuri maxim admis putem modifica instrucția 30 astfel:

```
30 DIM X(N + 1), Y(N + 1)
```

În exemplul următor se face estimarea suprafeței unui teren:

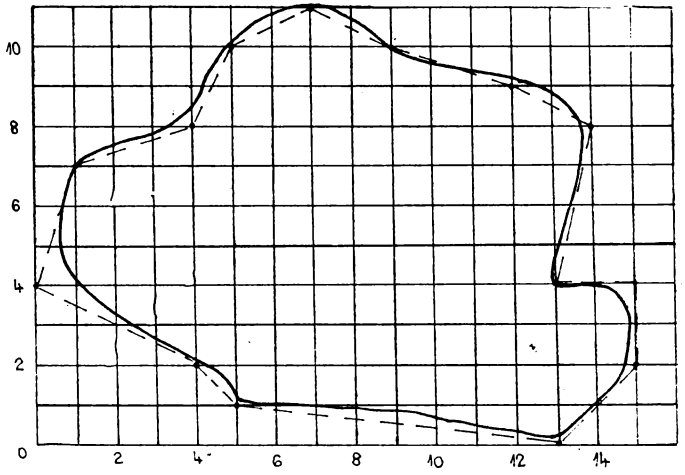


Fig. 5.2

```
: RUN
ARIA UNUI POLIGON
NUMARUL DE VIRFURI (INTRODUCETI 0 PT
    TERMINAREA PROGRAMULUI)  ? 14
COORDONATELE VIRFULUI  1 ?  0,4
             VIRFULUI  2 ? 1,7
             VIRFULUI  3 ?  4,8
             VIRFULUI  4 ?  5,10
             VIRFULUI  5 ?  7,11
             VIRFULUI  6 ?  9,10
             VIRFULUI  7 ? 12,9
             VIRFULUI  8 ? 14,8
             VIRFULUI  9 ? 13,4
             VIRFULUI 10 ? 15,4
             VIRFULUI 11 ? 15,1
             VIRFULUI 12 ? 13,0
             VIRFULUI 13 ?  5,1
             VIRFULUI 14 ?  4,2
ARIA = 108
NUMARUL DE VIRFURI (INTRODUCETI 0 PT
TERMINAREA PROGRAMULUI)  ? 0
END PROGRAM AT LINE 230
```

```
10  PRINT „Aria unui poligon."
20  PRINT
28  REM — Tabloul coordonatelor se dimensioneaza astfel
29  REM incit sa fie egal cu nr. virfurilor + 1.
30  DIM X(25), Y(25)
40  PRINT „Numarul de virfuri (introduceti 0 pt terminarea
    programului)"
50  INPUT N
59  REM — Sfirsit program? —
60  IF N = 0 THEN 230
69  REM — Ciclu pt introducerea coordonatelor în ordinea
    vîrfurilor
70  FOR I = 1 TO N
79
80      IF I>1 THEN 110
90      PRINT „Coordonatele virfului „;I
100     GOTO 120
110     PRINT "      virfului ";I
120     INPUT X(I), Y(I)
130 NEXT I
139 REM — Primul virf este ultimul virf —
140 X(N + 1) = X(1)
150 Y(N + 1) = Y(1)
160 A = 0
169 REM — Calculeaza si afiseaza aria —
170 FOR I = 1 TO N
180     A = A + (X(I) + X(I + 1))* (Y(I) — Y(I + 1)
190 NEXT I
200 PRINT „Aria = "; ABS(A)/2
210 PRINT
219 REM — Restartare program —
220 GOTO 40
230 END
```

5.3 — ELEMENTELE UNUI TRIUNGHI

Acest program calculează trei elemente necunoscute ale unui triunghi, atunci cînd se dau trei elemente cunoscute. Printre cele trei elemente cunoscute ale triunghiului, trebuie să se numere cel puţin o latură. Există cinci posibilităţi de a introduce datele:

1. Unghi, latură, unghi.
2. Latură, unghi, latură.

3. Unghi, unghi, latură.
4. Latură, latură, unghi.
5. Latură, latură, latură.

Datele trebuiesc introduse în ordinea în care ele apar în triunghi, în sensul acelor de ceasornic, sau în sens contrar (trigonometric).

Exemplu: Baza unui triunghi măsoară 14 cm. Unghiurile bazei măsoară respectiv 0,45 și 2,1 radiani. Care sînt celelalte dimensiuni ale triunghiului?

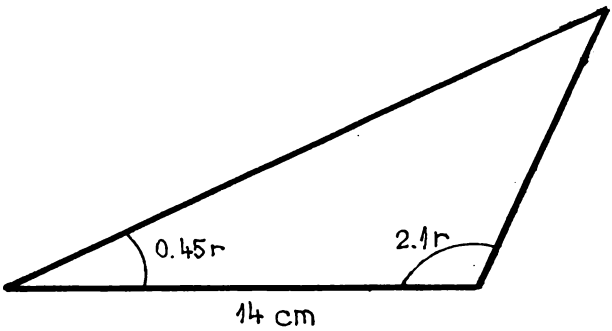
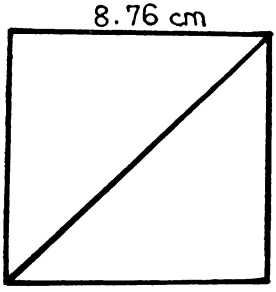


Fig. 5.3.1

```
:RUN
ELEMENTELE UNUI TRIUNGHI
TIPUL PROBLEMEI: 1 = ULU, 2 = LUL, 3 = UUL,
                 4 = LLU, 5 = LLL, 6 = SFIRȘIT
                 PROGRAM
INTRODUCETI TIPUL PROBLEMEI? 1
INTRODUCETI VALORILE IN ORDINEA UNGHI
LATURA UNGHI ?.45, 14, 2.1
LATURA 1 = 10.919
UNGHIUL OPUS = .45 RADIANI
LATURA 2 = 21.67
UNGHIUL OPUS = 2.16 RADIANI
LATURA 3 = 14
UNGHIUL OPUS = .592 RADIANI
INTRODUCETI TIPUL PROBLEMEI? 6
END PROGRAM AT LINE 560
```

Opțiune: Poate fi mai interesant să exprimăm unghiurile în grade mai degrabă decît în radiani. Modificările necesare în acest sens sînt prezentate după următorul exemplu.

Exemplu: Un pătrat are latura de 8.76 cm. Care este lungimea diagonalei sale?



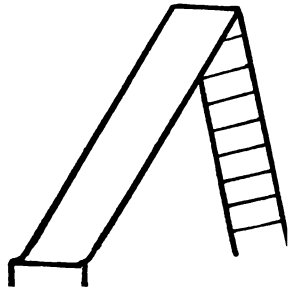


Fig. 5.3.2

Scara unui tobogan măsoară 3.048 m; toboganul măsoară 4.2672 m și acoperă 3.9624 m de teren de la baza scării pînă la capătul toboganului. Care este înclinarea toboganului?

```
:RUN
ELEMENTELE UNUI TRIUNGHI
TIPUL PROBLEMEI: 1 = ULU, 2 = LUL, 3 = UUL,
   4 = LLU, 5 = LLL, 6 = SFIRSIT PROGRAM
INTRODUCETI TIPUL PROBLEMEI ? 2
INTRODUCETI VALORILE IN ORDINEA LATURA,
   UNGHI, LATURA ? 8.76, 90, 8.76
LATURA 1 = 12.389
UNGHIUL OPUS = 90 GRADE
LATURA 2 = 8.76
UNGHIUL OPUS = 45 GRADE
LATURA 3 = 8.76
UNGHIUL OPUS = 45 GRADE
INTRODUCETI TIPUL PROBLEMEI ? 5
INTRODUCETI VALORILE ÎN ORDINEA LATURA,
   LATURA, LATURA ? 3.048, 4.2672, 3.9624
LATURA 1 = 3.048
UNGHIUL OPUS = 43.279 GRADE
LATURA 2 = 4.2672
UNGHIUL OPUS = 63.027 GRADE
LATURA 3 = 3.9624
UNGHIUL OPUS = 73.694 GRADE
INTRODUCETI TIPUL PROBLEMEI ? 6
END PROGRAM AT LINE 560
```

```
 10  PRINT „Elementele unui triunghi."
 20  PRINT
 30  DIM A(3), S(3)
 31  REM — Stabileste valoarea lui pi —
 40  P = 3.1415927
 48  REM — Introduceti numarul tipului problemei —
 49  REM — U = unghi, L = latura —
 50  PRINT „Tipul problemei: 1 = ULU, 2 = LUL, 3 = UUL,
       4 = LLU, 5 = LLL, 6 = sfîrsit program"
 60  PRINT „Introduceti tipul problemei"
 70  INPUT X
 79  REM — Salt la secventa de calcul corespunzatoare —
 80  IF X = 6 THEN 560
 90  IF X = 5 THEN 390
100  IF X = 4 THEN 300
110  IF X = 3 THEN 260
120  IF X = 2 THEN 190
130  PRINT „Introduceti valorile in ordinea: UNGHI, LATURA,
       UNGHI"
140  INPUT A (1), S(3), A(2)
150  A(3) = P — A(1) — A(2)
160  S(1) = S(3)*SIN(A(1))/SIN(A(3))
170  S(2) = S(3)*SIN(A(2))/SIN(A(3))
180  GOTO 440
190  PRINT „Introduceti valorile in ordinea: LATURA, UNGHI,
       LATURA"
200  INPUT S(3), A(1), S(2)
210  S(1) = SQR(S(3)^2 + S(2)^2 — 2*S(3)*S(2) + COS(A(1)))
220  A(2) = SIN(A(1))/S(1)*S(2)
230  A(2) = ATN(A(2)/SQR(1 — A(2)^2))
240  A(3) = P — A(1) — A(2)
250  GOTO 440
260  PRINT „Introduceti valorile in ordinea: UNGHI, UNGHI,
       LATURA"
270  INPUT A(3), A(2), S(3)
280  A(1) = P — A(2) — A(3)
290  GOTO 160
300  PRINT „Introduceti valorile in ordinea: LATURA, LATURA,
       UNGHI"
310  INPUT S(1), S(2), A(1)
320  T = S(2)*SIN(A(1))
330  IF S(1) < T THEN 520
340  S(3) = SQR(S(2)^2 — T^2)
```

```
350  IF S(1) <= T THEN 380
360  Y = SQR(S(1)^2 - T^2)
370  S(3) = S(3) + Y
380  GOTO 220
390  PRINT „Introduceti valorile in ordinea: LATURA, LATURA,
         LATURA"
400  INPUT S(1), S(2), S(3)
410  A(1) = S(2)^2 + S(3)^2 - S(1)^2)/2/S(2)/S(3)
420  A(1) = ATN(SQR(1 - A(1)^2)/A(1))
430  GOTO 220
440  PRINT
449  REM Afisare rezultate
450  FOR I = 1 TO 3
459       REM - Unghiul unui triunghi > 0 -
460       IF A(I) < 0 THEN 520
470       PRINT „LATURA"; I;" = „;INT(S(I)*1000 + .5)/1000
480       PRINT „Unghiul opus = „;INT(A(I)*1000 + .5)/1000;"
          radiani"
490  NEXT I
500  PRINT
510  GOTO 60
520  PRINT
530  PRINT „Solutie inexistenta"
540  PRINT
550  GOTO 60
560  END
```

Iată modificările ce trebuie făcute asupra programului:

— se introduc următoarele linii:

```
 44 REM - FACTOR DE CONVERSIE DIN RADIANI IN
    GRADE
 45 C = .0174532927
145 A(1) = A(1)*C
146 A(2) = A(2)*C
205 A(1) = A(1)*C
275 A(3) = A(3)*C
276 A(2) = A(2)*C
315 A(1) = A(1)*C
```

— se modifică linia 480 după cum urmează:

```
480 PRINT „UNGHIUL OPUS = „;INT(A(i)/C*1000 + .5)/1000;
    „GRADE"
```

5.4. ANALIZA A DOI VECTORI

Acest program calculează unghiul cuprins între doi vectori dați, unghiurile formate de fiecare dintre vectori cu axele de coordonate și modulul fiecărui vector. Vectorii sînt considerați în spațiul tridimensional.

Exemplu: Să se determine unghiul format de diagonala unui cub cu diagonala uneia dintre fețele sale. Latura cubului măsoară 4 cm.

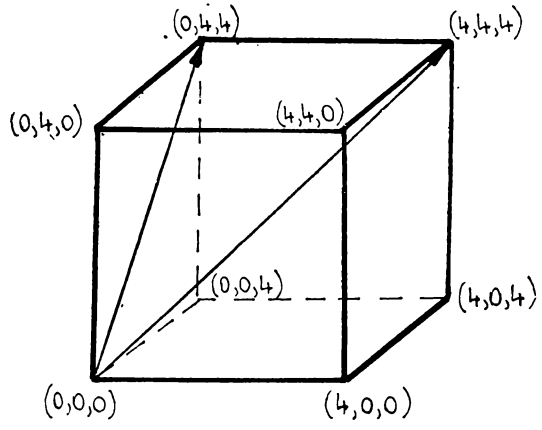


Fig. 5.4.1

```
: RUN
ANALIZA A DOI VECTORI
VECTOR 1 : X, Y, Z ? 0, 4, 4
VECTOR 2 : X, Y, Z ? 4, 4, 4
VECTOR 1:
MODUL : 5.6568542495
UNGHIUL CU AXA X: 90.00000076485
UNGHIUL CU AXA Y: 45.00000033257
UNGHIUL CU AXA Z: 45.00000033257
VECTOR 2:
MODUL: 6.9282032303
UNGHIUL CU AXA X: 54.73561078261
UNGHIUL CU AXA Y: 54.73561078261
UNGHIUL CU AXA Z: 54.73561078261
UNGHIUL DINTRE VECTORI: 35.26438998282
CONTINUAȚI (1 = DA, 0 = NU)? 0
END PROGRAM AT LINE 360
```

```
 10  PRINT „Analiza a doi vectori"
 20  PRINT
 30  DIM X(2), Y(2), Z(2), M(2)
 39  REM — Se introduc elementele vectorilor —
 40  PRINT „Vector 1 : X, Y, Z"
 50  INPUT X(1), Y(1), Z(1)
 60  PRINT „Vector 2 : X, Y, Z"
 70  INPUT X(2), Y(2), Z(2)
 80  PRINT
 89  REM — In acest ciclu se analizeaza vectorii —
 90  FOR I = 1 TO 2
 99      REM — Calculeaza si afiseaza modulul —
100      M(I) = SQR(X(I)^2 + Y(I)^2 + Z(I)^2)
109      REM — Daca vectorul e un punct nu putem calcula
         unghiul —
110      IF M(I) = 0 THEN 220
120      PRINT „Vector" ;I;" :"
130      PRINT „Modul : ";M(I)
139      REM— Factorul de conversie radian — grade —
140      S = 57.29578
149      REM — Calculeaza si afiseaza unghiul dintre vector s
         axa X—
150      J = X(I)/M(I)
160      PRINT „Unghiul cu axa X:"; ATN(SQR(1 — J^2)/J)*S
169      REM — Calculeaza si afiseaza unghiul dintre vector
         și axa Y —
170      J = Y(I)/M(I)
180      PRINT „Unghiul cu axa Y: "; ATN(SQR(1 — J^2)/J)*S
180      PRINT „Unghiul cu axa Y"
189      REM — Calculeaza si afiseaza unghiul dintre vector
         și axa Z —
190      J = Z(I)/M(I)
200      PRINT „Unghiul cu axa Z:"; ATN(SQR(1 — J^2)/J)*S
210      PRINT
220  NEXT I
230  J = 0
239  REM — Daca unul dintre vectori e punct nu se poate calcula
     unghiul —
240  IF M(1) = 0 THEN 210
250  IF M(2) = 0 THEN 310
259  REM — Calculul unghiului dintre vectori —
260  J = (X(1)*X(2) + Y(1)*Y(2) + Z(1)*Z(2))/M(1)/M(2)
269  REM — Vectorii sint perpendiculari? —
270  IF J◇0 THEN 300
280  J = 90
290  GOTO 310
```

```
299  REM — Calculeaza si affseaza unghiul in grade —
300  J = ATN  (SQR(1 — J^2)/J)*S
310  PRINT „Unghiul dintre vectori:";J
320  PRINT
329  REM — Restart program sau sfirsit program —
330  PRINT „Continuati  (1 = DA,  0 = NU)"
340  INPUT  Z
350  IF  Z = 1  THEN  20
360  END
```

5.5 — OPERAȚII CU DOI VECTORI

Acest program efectuează patru operații cu doi vectori dați, în spațiul tridimensional. Cele patru operații sînt:

— adunare;
— scădere;
— produs scalar;
— produs vectorial.

Exemplu: Fie două puncte A și B de coordonate (5, —1, 2) și (1, 4, 9). Să se efectueze adunarea, scăderea, produsul scalar și produsul vectorial al celor doi vectori obținuți prin unirea originii cu punctele A și B.

```
:RUN
OPERATII CU DOI VECTORI
VECTOR A: X, Y, Z? 5, —1, 2
VECTOR B: X, Y, Z? 1, 4, 9
A+B = 6,  3,  11
A—B=4,  —5,  —7
A.B = 19
A × B = 17,  —43,  21
CONTINUATI  (1 = DA,  0 = NU) ? 0
END PROGRAM AT LINE 160
```

```
10 PRINT „Operatii cu doi vectori"
20 PRINT
30 PRINT „Vector  A : X,  Y,  Z"
40 INPUT X1, Y1, Z1
50 PRINT „Vector  B:  X,  Y,  Z"
60 INPUT X2, Y2, Z2
70 PRINT
79 REM — Efectueaza adunarea vectorilor și afiseaza coordonatele vectorului rezultat —
80 PRINT „A+B = „;X1+X2;", ";Y1+Y2;", ";Z1+Z2
```

```
 89 REM — Efectueaza scaderea vectorilor și afiseaza coordonatele vectorului rezultat —
 90 PRINT „A−B = „;X1−X2;“, „;Y1−Y2“, “;Z1−Z2
 99 REM — Efectuează produsul scalar și afiseaza —
100 PRINT „A·B = „;X1*X2+Y1*Y2+Z1*Z2
109 REM — Efectueaza și afiseaza produsul vectorial —
110 PRINT „A*B = “;Y1*Z2−Z1*Y2;“, “;Z1*X2−X1*Z2;“, “;
    X1*Y2 − _Y1*X2
120 PRINT
129 REM — Restart sau sfirsit program —
130 PRINT „Continuati (1=DA, 0=NU)“
140 INPUT X
150 IF X=1 THEN 20
160 END
```

5.6 — CONVERSIA UNGHIURILOR DIN RADIANI ÎN GRADE

Acest program realizează conversia unui unghi dat din radiani în grade, minute, secunde.

Exemplu: Să se transforme în grade, minute și secunde următoarele valori exprimate în radiani: 2,5; 118;

```
:RUN
CONVERSIA UNGHIURILOR DIN RADIANI IN GRADE
MASURA UNGHIULUI IN  RADIANI (INTRODUCETI
0 PT TERMINARE PROGRAM)? 2.5
   GRADE = 143
   MINUTE = 14
   SECUNDE = 22.01
MASURA UNGHIULUI IN RADIANI? 118
   GRADE = 280
   MINUTE = 54
   SECUNDE = 6,78
MASURA UNGHIULUI IN RADIANI? 0
END PROGRAM AT LINE 170
```

```
 10 PRINT „Conversia unghiurilor din radiani în grade“
 20 PRINT
 30 PRINT „Măsura unghiului in radiani (introduceți 0 pt terminare program)“
 40 GOTO 60
 50 PRINT „Măsura unghiului in radiani“
 60 INPUT R
```

```
 69 REM — Test de sfirsit program —
 70 IF R=0 THEN 170
 79 REM — Conversie radiani in secunde —
 80 A=3600*180*R/3. 1415927
 89 REM — Calculeaza numarul total de grade —
 90 D=INT(A/3600)
 99 REM — Calculeaza numarul circumferintelor intregi —
100 D1=INT(D/3600)
110 PRINT "Grade" = ;D-360*D1
119 REM — Calculeaza și afiseaza minutele —
120 PRINT „Minute="; INT((A—D*3600)/60)
129 REM — Calculeaza secundele rotunjeste si afiseaza"
130 S=A—D*3600—(INT((A—D*3600)/60))*60
140 PRINT „Secunde=";INT(100*S+.5)/100
150 PRINT
159 REM — Restart program —
160 GOTO 50
170 END
```

5.7 — CONVERSIA UNGHIURILOR din GRADE în RADIANI

Acest program convertește valoarea unui unghi din grade, minute, secunde, în radiani.

Exemplu: Un unghi măsoară 30 grade, 5 minute, 3 secunde. Care va fi măsura acestui unghi exprimată în radiani? Care va fi măsura în radiani a două unghiuri care măsoară respectiv 278°, 19', 54" și 721°, 0' 0"?

```
:RUN
CONVERSIA UNGHIURILOR DIN GRADE IN RADIANI
(PENTRU TERMINAREA PROGRAMULUI INTRODUCETI 0, 0, 0)
MASURA UNGHIULUI IN GRADE, MINUTE, SECUNDE? 30, 5, 3,
RADIANI=.5250676852416
MASURA UNGHIULUI IN GRADE, MINUTE, SECUNDE? 278, 19, 54
RADIANI = 4.857803294516
MASURA UNGHIULUI IN GRADE, MINUTE, SECUNDE? 721, 0, 0
RADIANI = 1.74514900 E-2
MASURA UNGHIULUI IN GRADE, MINUTE, SECUNDE? 0, 0, 0
END PROGRAM AT LINE 150
```

```
10 PRINT „Conversia unghiurilor din grade in radiani"
20 PRINT
30 PRINT „(Pentru oprirea programului introduceți 0, 0,0)"
40 PRINT „Masura unghiului in grade, minute, secunde"
50 INPUT D, M, S
59 REM — Test de sfirsit program —
60 IF D<>0 THEN 100
70 IF M<>0 THEN 100
80 IF S<>0 THEN 100
90 GOTO 150
99 REM — Conversie grade, minute, secunde in grade —
100 A=D+M/60+S/3600
109 REM — Calculeaza nr de circumferinte intregi —
110 R=INT(A/360)
119 REM — Calculeaza si afiseaza unghiul cuprins in interiorul a
360 grade —
120 PRINT „Radiani"; A*.01745329—R*6.2831853
130 PRINT
139 REM — Restart program —
140 GOTO 40
150 END
```

Opțiune: programul ar putea fi mai util dacă măsura unghiului nu ar fi exprimată în grade, minute, secunde, ci în grade și fracțiuni de grade. Modificările ce trebuie, făcute în program sînt prezentate după următorul exemplu:

Exemplu: Să se exprime în radiani măsura unui unghi de 33.08°.

```
:RUN
CONVERSIA UNGHIURILOR DIN GRADE IN RADIANI
(PENTRU TERMINAREA PROGRAMULUI INTRODU-
CEȚI 0)
MASURA UNGHIULUI IN  GRADE? 33.08
RADIANI = .5773548332
MASURA UNGHIULUI  IN GRADE? 0
END PROGRAM AT LINE 150
```

Iată cum trebuie modificate liniile programului cuprinse între 30 și 60:

```
30 PRINT „(PENTRU TERMINAREA PROGRAMULUI
INTRODUCETI 0)"
40 PRINT „MASURA UNGHIULUI ÎN GRADE";
50 INPUT A
60 IF A=0 THEN 150
```

5.8 — CONVERSIA COORDONATELOR

Acest program realizează conversia unui punct exprimat în coordonate carteziene, în coordonate polare și invers.

Formulele de conversie sînt următoarele:

$$r = \sqrt{X^2 + Y^2}$$
$$A = \text{arctg } (Y/X)$$
$$X = r \cos(A)$$
$$Y = r \sin (A)$$

unde: X = abscisă; Y = ordonata (în coordonate carteziene), iar r = modulul; A = argumentul (în grade) (în coordonate polare).

Exemplu: Să se determine coordonatele carteziene ale punctului (2,30.15°) exprimat în coordonate polare.

Dacă un punct are coordonatele carteziene (7,18), care sînt coordonatele polare?

Să se determine poziția, în coordonate polare, a punctului situat la (0, −46.8) coordonate carteziene.

```
: RUN
CONVERSIA COORDONATELOR
          (1 = CARTEZIAN-POLAR)
          (−1 = POLAR-CARTEZIAN)
          ( 0 = SFIRSIT PROGRAM)
SENSUL CONVERSIEI? −1
R, A? 2,30.15
X = 1.72,       Y = 1.02
SENSUL CONVERSIEI? 1
X, Y? 7,18
R = 19.31,      A = 68.75
SENSUL CONVERSIEI? 1
X, Y? 0, −46.8
R = 46.8,       A = 270
SENSUL CONVERSIEI? 0
END PROGRAM AT LINE 380
```

```
10  PRINT „Conversia coordonatelor"
20  PRINT
30  PRINT „(1 = Cartezian — polar)"
40  PRINT „(−1 = Polar — cartezian)"
50  PRINT (0 = Sfirsit program)"
60  PRINT „Sensul conversiei
```

```
 70  INPUT D
 79  REM — Sfirsit program —
 80  IF D=0 THEN 380
 89  REM — Salt la secventa de conversie solicitata —
 90  IF D=-1 THEN 320
 98  REM — Conversie cartezian — polar —
 99  REM — Introduceti coordonate carteziene (abscisa, ordo-
     nata)"
100  PRINT „X, Y"
110  INPUT X, Y
119  REM — Punct pe axa Y —
120  IF X=0 THEN 170
129  REM — Punct pe axa X —
130  IF Y=0 THEN 260
139  REM — Calculeaza rotunjește si afiseaza coordonatele
     polare —
140  PRINT „R = "; INT(SGN(X)*SQR(X^2+Y^2)*100++.5)/
     /100; „,";
150  PRINT „A = "; INT(ATN(Y/X)*180/3.1415927*100+.5)/100
160  GOTO 60
169  REM — Punctul e pe axa Y; la origine? —
170  IF Y=0 THEN 240
180  PRINT „R = "; ABS(Y); ",";
189  REM — Punctul se afla deasupra sau dedesubtul originii? —
190  IF Y<0 THEN 220
200  PRINT „A = 90"
210  GOTO 60
220  PRINT „A = 270"
230  GOTO 60
239  REM — Punctul e in origine —
240  PRINT „R = 0, A = 0"
250  GOTO 60
259  REM — Punctul e pe axa X —
260  PRINT „R = "; ABS(X); „,";
269  REM — Punctul se afla la stinga sau la dreapta originii —
270  IF X<0 THEN 300
280  PRINT „A = 0"
290  GOTO 60
300  PRINT „A = 180"
310  GOTO 60
318  REM Introduceti ccordonate polare (modul, argument) —
319  REM Conversie polar — cartezian —
```

```
320 PRINT „R, A"
330 INPUT R, A
339 REM — Conversie grade — radiani —
340 M=(A-INT(A/360)*360)*3.1415927/180
349 REM — Calculeaza coordonatele carteziene, rotunjeste si
    afiseaza —
350 PRINT „X = "; INT(R*COS(M)*100+.5)/100; „„ ;
360 PRINT „Y = "; INT(R*SIN(M)*100+.5)/100
370 GOTO 60
380 END
```

5.9 — TRASAREA CURBELOR

Acest program trasează puncte într-un sistem de axe de coordonate. Este necesar să se indice coordonatele X și Y ale fiecărui punct de trasat, valorile extremităților din stînga și dreapta pe axele X și Y și incrementul dintre puncte pe fiecare axă.

Așezarea în pagină este neconvențională, în sensul că axa X este orientată vertical, iar axa Y orizontal. De asemenea, cele două axe nu se intersectează neapărat în origine. Numărul punctelor de trasat poate fi mărit sau diminuat modificînd instrucția 30 astfel:

```
30 DIM X(N+1), Y(N+1)
```

unde N este numărul maxim de puncte de trasat.

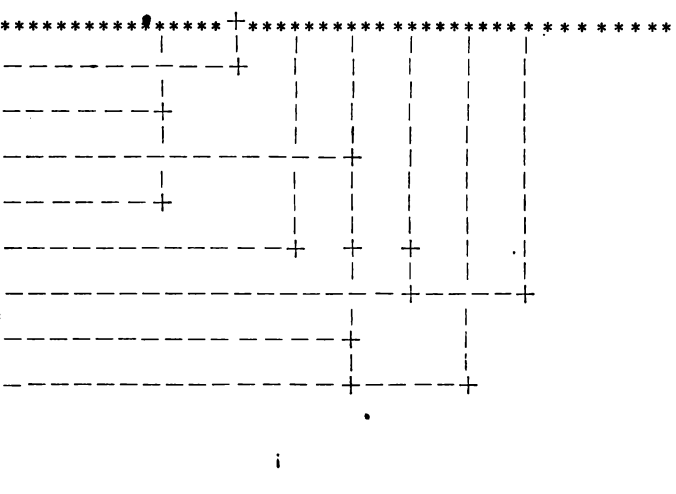
Lungimea axei Y este limitată de lățimea suportului de ieșire. Programul verifică ca lungimea axei Y să nu depășească 70 de caractere. Pentru a adopta programul la un alt mediu extern de ieșire este necesar să se modifice instrucția 90. De exemplu, pentru un suport de ieșire a cărui linie are o lungime de 132 de caractere, instrucția 90 va fi următoarea:

```
90 IF B2 < = 132 THEN 120
```

Exemplu: Tabloul de mai jos conține măsura, în centimetri, a taliei a 12 bărbați și a fiilor lor. Să se traseze punctele corespunzătoare.

65	63	67	64	68	62	70	66	68	67	69	71
68	66	68	65	69	66	68	65	71	67	68	70

```
: 30 DIM X(13), Y(13)
: RUN
TRASAREA CURBELOR
AXA X: EXTREMA STINGA, EXTREMA DREAPTA,
INCREMENT? 62, 73, .5
AXA Y: EXTREMA INFERIOARA, EXTREMA SUPE-
RIOARA, INCREMENT? 62, 73, .25
NUMARUL PUNCTELOR DE TRASAT? 12
COORDONATELE PUNCTULUI 1  ? 65, 68
              PUNCTULUI 2  ? 63, 66
              PUNCTULUI 3  ? 67, 68
              PUNCTULUI 4  ? 64, 65
              PUNCTULUI 5  ? 68, 69
              PUNCTULUI 6  ? 62, 66
              PUNCTULUI 7  ? 70, 68
              PUNCTULUI 8  ? 66, 65
              PUNCTULUI 9  ? 68, 71
              PUNCTULUI 10 ? 67, 67
              PUNCTULUI 11 ? 69, 68
              PUNCTULUI 12 ? 71, 70
AXELE DE COORDONATE SE INTERSECTEAZA IN
PUNCTUL (62, 62)
```



```
END PROGRAM AT LINE 1070
```

```
10  PRINT „Trasarea curbelor"
20  PRINT
28  REM — Dimensiunea lui X() și Y() va fi limitata la N+1
29  REM unde N este nr punctelor de trasat; limita maxima=99
30  DIM X(100), Y(100)
39  REM — Se introduc informatii asupra axelor —
40  PRINT „Axa x: extrema stinga, extrema dreapta, increment"
50  INPUT A1, A2, A3
60  PRINT „Axa y: extrema inferioara, extrema superioara,
    increment"
70  INPUT B1, B2, B3
80  B2=(B2-B1)/B3
88  REM — Axa Y este prea lunga? Daca da se schimba extre-
    mitatea
89  REM — Salt la secventa de conversie solicitata —
90  IF B2<=70 THEN 120
100 PRINT „Axa Y prea mare"
110 GOTO 60
120 PRINT „Numarul punctelor de trasat"
130 INPUT N
139 REM — Nu sint puncte de trasat? Sfirsit program
140 IF N=0 THEN 1070
149 REM — Prea multe puncte. Daca da se introduce nr de
    puncte— —
150 IF N<=99 THEN 180
160 PRINT „Prea multe puncte"
169 REM — Punctul e pe axa Y; la origine? —
170 GOTO 120
179 REM — Se introduc coordonatele X, Y pt fiecare punct —
180 FOR I = 1 TO N
190             IF I>1 THEN 220
200             PRINT „Coordonatele punctului "; I
210             GOTO 230
220             PRINT „              punctului "; I
230             INPUT X(I), Y(I)
239             REM — Rotunjeste fiecare X, Y la cel mai
                apropiat increment al axei—
240             X(I) = INT((X(I)-A1)/A3+.5)
250             Y(I) = INT((Y(I) - B1)/B3+.5)
260 NEXT I
269 REM — Calculeaza coordonatele X, Y aditionale —
270 Y(N+1) = INT(B2+.5)+1
280 X(N+1) = INT((A2-A1)/A3+.5)+1
```

```
290 PRINT
299 REM — Punctul de intersectie al axelor —
300 PRINT „Intersectia axelor la („;A1;““;B1;
310 PRINT
319 REM — Sortare coordonate in ordine crescatoare —
320 FOR J=1 TO N
330           FOR I=1 TO N-J
340                   A=X(I)
350                   B=Y(I)
360                   C=X(I+1)
370                   D=Y(I+1)
380                   IF A<C THEN 430
390                   X(I)=C
400                   Y(I)=D
410                   X(I+1)=A
420                   Y(I+1)=B
430           NEXT I
440 NEXT J
449 REM — Urmatorul punct de trasat este depus in T
450 T=1
460 FOR P=0 TO N-1
470           IF X(P+1) >=0 THEN 490
480 NEXT P
490 FOR I=1 TO INT((A2-A1)/A3+.5)
500           T=T+P
509           REM — Determina nr de puncte de trasat pe
              fiecare linie
510           P=0
519           REM — Toate punctele au fost trasate?
520           IF T>N THEN 540
529           REM — Valoarea lui X pe axa Y? Daca da
              testeaza Y
530           IF (X(T)=I THEN 590
539           REM — Prima linie? Daca da axa Y trebuie
              trasata —
540           IF I=0 THEN 570
549           REM — Trasare axa X —
550           PRINT „*“
560           GOTO 1040
570           S=N+1
580           GOTO 920
590           FOR L=T TO N
```

```
599                 REM — Urmatorul punct e pe
                    aceeasi linie —
600                 IF X(L)>X(T) THEN 630
609                         REM — Determina nr de-
                    puncte ce trebuie trasat pe fiecare linie
610                 P=P+1
620           NEXT L
629           REM — Afiseaza un punct —
630           IF P=1 THEN 730
638           REM — Sortare coordonate Y cu coordonate X
              egale —
639           REM — Reordonare crescatoare —
640           FOR J=1 TO P
650                 FOR L=1 TO P-J
660                         D=Y(T+L-1)
670                         B=Y(T+L)
680                         IF D<=B THEN 710
690                         Y(T+L-1)=B
700                         Y(T+L)=D
710                 NEXT L
720           NEXT J
730           FOR L=0 TO P-1
740                 Z=Y(T+L)
749                 REM — Se testeaza daca Y in afara
                    rangului —
750                 IF Z>=0 THEN 770
760           NEXT L
769           REM — Punctul trebuie trasat pe axa X? —
770           IF I=0 THEN 910
779           REM — Punctul trebuie trasat pe axa Y? —
780           IF Z=0 THEN 800
789           REM — Afiseaza axa X —
790           PRINT „*"
800           IF L=P-1 THEN 870
810           FOR J=L TO P-1
819                 REM —Se testeaza daca Y in afara
                    rangului —
820                 IF Z>B2 THEN 1040
829                 REM — Ignora coordonatele duble—
830           IF Y(T+J)=Z THEN 860
839                 REM — Afiseaza punctul —
840                 PRINT TAB(Z); „+";
850                 Z=Y+(TJ)
```

```
860             NEXT J
869             REM — Se testeaza daca Y inafara rangului —
870             IF Z<0 THEN 1040
880             IF Z>B2 THEN 1040
889             REM — Afiseaza punctul —
890             PRINT TAB(Z); „+“;
900             GOTO 1040
910             S=T+L
920             FOR J=0 TO B2
                        REM — Punct de afisat? —
930                     IF Y(S)<>J THEN 1010
939                     REM — Afisare punct —
940                     PRINT „+“;
949                     REM Ignora coordonatele duble —
950                     FOR K=S TO T+P-1
960                         IF Y(K)=Y(S) THEN 990
970                             S=K
980                             GOTO 1020
990                     NEXT K
1000                    GOTO 1020
1009                    REM — Deseneaza axa Y —
1010                    PRINT „*“;
1020            NEXT J
1029            REM — Eticheteaza axa Y —
1030             PRINT „Y“;
1040            PRINT
1050 NEXT I
1059 REM — Eticheteaza axa X —
1060 PRINT „X“;
1070 END
```

5.10 — TRASAREA CURBELOR ÎN COORDONATE POLARE

Acest program trasează o funcţie dată în coordonate polare. Pot fi trasate maximum 90 de puncte, dintre care anumite puncte se pot suprapune. Graficul este convenţional, în sensul că axa *X* este orizontală, axa *Y* este verticală, iar cele două axe se intersectează în origine. Trebuie indicată numai valoarea absolută a extremităţilor axelor. Incrementul dintre puncte pe axele *X* şi *Y* este de asemenea ajustat, astfel încît trasarea funcţiei să se facă cu o distorsiune minimă. Datorită faptului că, pe mediul extern, spaţierea orizontală este diferită de cea verticală, ajustarea incrementului pe axele *X* şi *Y* se face separat. Programul presupune că pe

axa X pot fi reprezentate 60 de puncte (30 de puncte negative și 30 pozitive), iar pe axa Y 36 de puncte (18 negative și 18 pozitive). Înainte de lansarea programului trebuie introdusă funcția de trasat, $f(d)$. Această funcție va fi introdusă pe linia 130 și va fi atribuită variabilei F.

Exemplu: Să se traseze funcția: $f(d) = 2\ (1 - \cos(d))$

```
: 130 F = 2*(1 - cos(1))
:RUN
TRASAREA CURBELOR IN COORDONATE POLARE
VALOAREA ABSOLUTA A EXTREMITATILOR AXE-
LOR? 4
INCREMENTUL AXEI X = .1333333333333
INCREMENTUL AXEI Y = .2222222222222
```

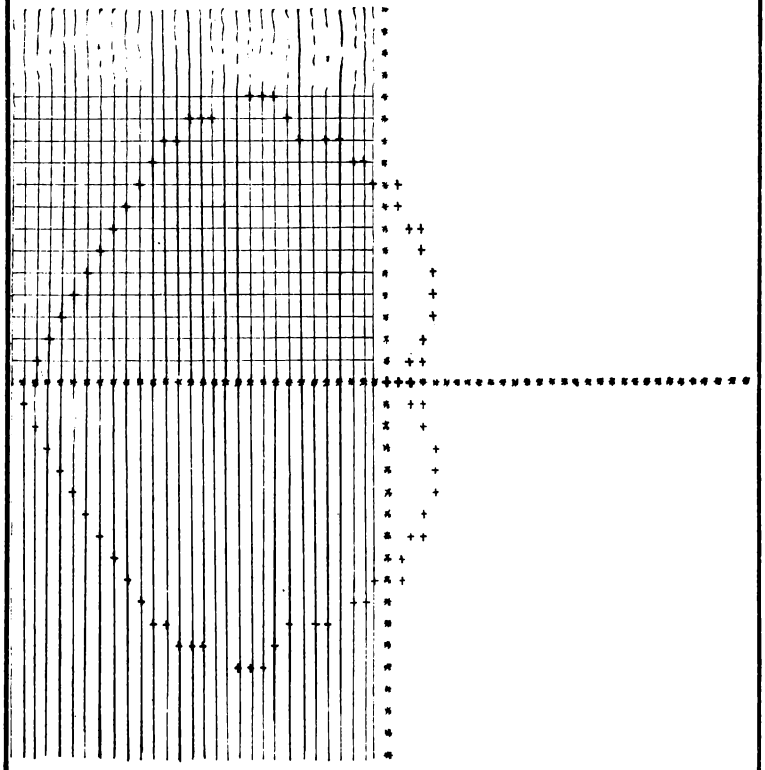


Fig. 5.10

```
 10 PRINT „Trasarea curbelor in coordonate polare"
 20 PRINT
 28 REM — Tabela de coordonate este dimensionata pt 90 de
    puncte —
 29 REM — O coordonata X suplimentara e calculata in pro-
    gram —
 30 DIM X(91), Y(90)
 39 REM — Nr de puncte de calculat —
 49 N=90
 49 REM — Valoarea absoluta a punctelor de pe extremitati
    este egala
 50 PRINT „Valoarea absolută a extremitatilor axelor"
 60 INPUT Z
 70 PRINT
 79 REM — Calculul incrementului pe axa in functie de nr
    punctelor pe axa
 80 PRINT „Incrementul pe axa X = „;Z/30
 90 PRINT „Incrementul pe axa Y = „;Z/18
100 PRINT
110 FOR I=1 TO N
119          REM — Conversie grade in radiani —
120          D=.06981317*I
130          REM — Aici se introduce functia —
139          REM — Calculeaza coordonatele carteziene —
140          X(I)=INT(((F*COS(D)/Z+1)*30)+.5)
150          Y(I)=INT(((-F*SIN(D)/Z+1)*18)+.5)
160          NEXT I
169 REM — Sortare coordonate Y(1)...Y(n) crescator —
170 FOR J=1 TO N
180          FOR I=1 TO N-J
190                    A=X(I)
200                    B=Y(I)
210                    IF B<=Y(I+1) THEN 260
220                    X(I)=X(I+1)
240                    X(I+1)=A
250                    Y(I+1)=B
260               NEXT I
270          NEXT J
```

```
279  REM — Urmatorul punct de trasat stocat in T —
280  T=1
290  FOR P=0 TO N-1
300          IF Y(P+1)>=0 THEN 320
310  NEXT P
320  FOR I=0 TO 36
330          T=T+P
339          REM — Nr punctelor de trasat pe fiecare linie
             stocate in P —
340          P=0
349          REM — Au fost trasate toate punctele —
350          IF T>N THEN 370
359          REM — Valoarea lui Y se afla pe axa Y —
360          IF Y(T)=I THEN 420
370          IF I=18 THEN 400
379          REM — Afiseaza axa Y —
380          PRINT TAB(30); „*“;
390          GOTO 860
400          S=N+1
410          GOTO 740
420          FOR L=T TO N
429                  REM — Urmatorul punct de afisat se
                     afla pe aceeasi linie? —
430                  IF Y(L)>Y(T) THEN 450
440                  P=P+1
450          NEXT L
460          IF P=1 THEN 560
468          REM — Sortare coordonate X cu coordonate Y
             egale —
469          REM — Ordonare crescatoare —
470          FOR J=1 TO P
480                  FOR L=1 TO P-J
490                          C=X(T+L-1)
500                          A=X(T+L)
510                          IF C<=A THEN 540
520                          X(T+L-1)=A
530                          X(T+L)=C
540                  NEXT L
```

```
550         NEXT J
559         REM — Se afiseaza axa Y? —
560         IF I=18 THEN 730
570         L=-1
580         S=0
590         FOR K=0 TO P-1
599               REM — Exista mai multe puncte de
                  trasat in acelasi punct de pe grafic —
600               IF X(T+K)=L THEN 690
610               L=X(T+K)
619               REM — Afisare punct pe axa Y? —
620               IF L=30 THEN 660
629               REM — Afisare punct la stinga axei
                  Y? —
630               IF L<30 THEN 670
640               IF S=1 THEN 670
649               REM — Afisare axa Y —
650               PRINT TAB (30); „*“;
660               S=1
670               IF L>60 THEN 860
680               PRINT TAB(L); „+“;
690         NEXT K
700         IF S=1 THEN 860
709         REM — Afisare axa Y —
710         PRINT TAB (30); „*“;
720         GOTO 860
730         S=T
739         REM — Ciclu de afisare al liniilor axei X —
740         FOR J=0 TO 60
750               IF X(S)<>J THEN 830
759               REM — Afisare punct pe axa X —
760               PRINT „+“;
770                     FOR K=S TO T+P-1
780                  IF X(K) = X(S) THEN 810
790                     S=K
800                     GOTO 840
810               NEXT K
820                     GOTO 840
```

```
829                                   REM — Afisare axa X—
830                                   PRINT „*";
840                          NEXT J
850                          PRINT „X";
860                          PRINT
870  NEXT I
879  REM — Eticheteaza axa Y —
880  PRINT TAB(30); „Y";
890  END
```

5.11 — TRASAREA FUNCȚIILOR

Acest program determină și trasează graficele a maximum 9 funcții diferite. Toate aceste funcții sînt funcții de o singură variabilă, de argument X și sînt trasate în cadrul aceluiași sistem de axe de coordonate.

Pentru a poziționa axele, trebuie indicate extremitățile axelor X și Y și, de asemenea, incrementul dintre puncte pe fiecare axă. Graficul este neconvențional, în sensul că axa X este verticală, iar axa Y orizontală. Pentru a putea f i citit, graficul trebuie rotit cu 90° în sens trigonometric.

De asemenea, graficul este neconvențional datorită faptului că axele nu se intersectează, în mod obligatoriu, în origine. Funcțiile de trasat trebuie introduse ca instrucții ale programului, înaintea lansării programului. Liniile de la 221 la 229 sînt special rezervate în acest scop. Funcțiile trebuie atribuite variabilelor $Y(1)$, $Y(2)$, ..., $Y(9)$. Dacă, de exemplu, avem de trasat graficele a două funcții $f(X) = 2X + 1$ și $f(x) = \sqrt{X}$, vom introduce:

```
221  Y(1) = 2*X + 1
222  Y(2) = SQR(X)
```

Lungimea axei Y este limitată de lungimea liniei suportului de ieșire. Programul verifică ca aceasta să nu fie mai mare de 70 de caractere. Testul din linia 140 va trebui modificat pentru o eventuală adaptare a programului la un mediu extern particular. De exemplu, dacă suportul de ieșire are lungimea liniei de 132 de caractere, vom modifica astfel linia 140:

```
140 IF Y2 < = 132 THEN 170
```

Exemplu: Să se traseze graficul funcțiilor $f(X) = \cos(X)$ și $f(X) = \sin(X)$

```
: 221 Y(1) = cos(X)
: 222 Y(2) = sin (X)
: RUN
TRASAREA FUNCTIILOR

NUMARUL FUNCTIILOR DE TRASAT? 2
AXA X : EXTREMA STINGA, EXTREMA DREAPTA,
INCREMENT? −5, 5, .25
AXA Y : EXTREMA INTERIOARA, EXTREMA SUPE-
RIOARA, INCREMENT? −2, 2, .1
AXA X INTERSECTEAZA AXA Y ÎN Y = −2
AXA Y INTERSECTEAZA AXA X ÎN X = −5
```

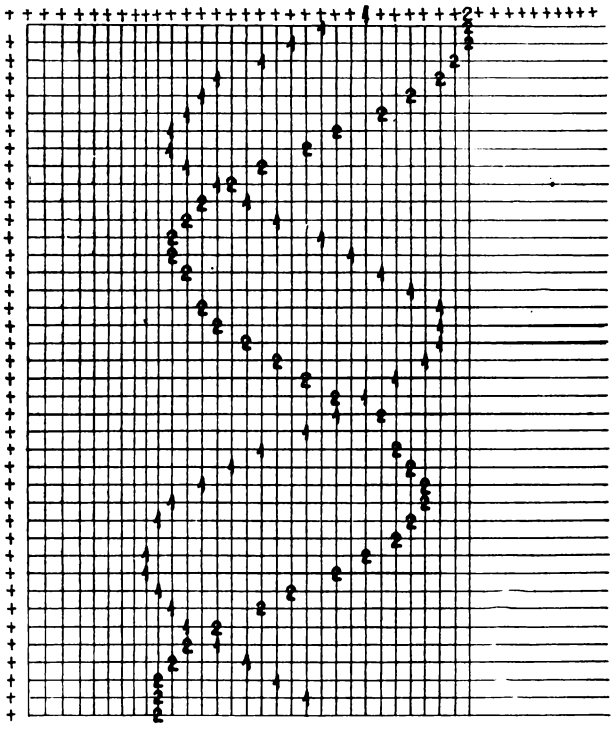


Fig. 5.11

```
 10  PRINT „Trasarea functiilor“
 20  PRINT
 29  REM — Numarul functiilor care pot fi trasate este limitat
     la 9 —
 30  DIM Y(9), A$(11)
 40  FOR I=1 TO 11
 50          READ A $(I)
 60  NEXT I
 70  PRINT „Numarul functiilor de trasat“
 80  INPUT N
 90  PRINT „Axa X : extrema stinga, extrema dreapta, incre-
     ment“
100  INPUT X1, X2, X3
110  PRINT „Axa Y : extrema inferioara, extrema superioara,
     increment“
120  INPUT Y1, Y2, Y3
130  Y2=(Y2−Y1)/Y3
140  IF Y2<=70 THEN 170
150  PRINT „Axa Y prea lunga“
160  GOTO 110
170  PRINT
180  PRINT
190  PRINT „Axa X intersecteaza axa Y in punctul Y = „;Y1
200  PRINT „Axa Y intersecteaza axa X in punctul X = „;X1
210  PRINT
220  FOR X=X1 TO X2 STEP X3
230          FOR I=1 TO N
240                  Y(I)=INT((Y(I)−Y1)/Y3+.5)
250          NEXT I
260          FOR I=0 TO Y2
270                  S=0
280                  FOR J=1 TO N
290                          IF Y(J)<>I THEN320
300                          S=S+1
310                          T=J
320                  NEXT J
330                  IF S>0 THEN 360
                     PRINT A $ (SGN(I)+10);
350                  GOTO 400
360                  IF S>1 THEN 390
370                  PRINT A $ (T);
380                  GOTO 400
390                  PRINT „*“;
400          NEXT I
410          IF X>X1 THEN 430
420          PRINT „Y“;
430          PRINT
440          A $(11) = „“
450  NEXT X
460  PRINT „X“
470  DATA „1“, „2“, „3“, „4“, „5“, „6“, „7“, „8“, „9“. „+“, „+“
480  END
```

5.12 — INTERPOLARE LINIARĂ

Acest program calculează coordonatele Y ale punctelor pe o linie, fiind date coordonatele X. Este necesar să cunoaștem coordonatele Y pentru cel puțin două puncte dintr-o linie.

Formula de interpolare utilizată este următoarea:

$$Y = Y_1 + \frac{(Y_2 - Y_1) \cdot (X - X_1)}{(X_2 - X_1)}$$

unde:

X_1, Y_1 — coordonatele primului punct cunoscut al liniei;
X_2, Y_2 — coordonatele celui de-al doilea punct cunoscut al liniei;
X — abscisa punctului de interpolat;
Y — ordonata punctului de interpolat.

Exemplu: O tabelă de conversie a gradelor Faraday în grade Celsius indică 15.56°C pentru 60°F și 32.22°C pentru 90°F.
Să se transforme în grade Celsius 73°F și 85.6°F.
O întreprindere de comerț cu amănuntul percepe un comision de 17,5% asupra mărfurilor vîndute. Care va fi suma încasată din vînzarea unui obiect în valoare de 455.68 lei?

```
: RUN
INTERPOLARE LINIARĂ
X, Y ALE PRIMULUI PUNCT? 60, 15.56
X, Y ALE CELUI DE-AL DOILEA PUNCT? 90, 32.22
INTERPOLARE:  X = ? 73
              Y = 22.779
ALT PUNCT DE INTERPOLAT PE ACEEASI LINIE
(1=DA, 0=NU)? 1
INTERPOLARE X = ? 85.6
            Y = 29.779
ALT PUNCT DE INTERPOLAT PE ACEEASI LINIE
(1=DA, 0=NU)? 0
ALTA LINIE DE INTERPOLAT (1=DA, 0=NU)? 1
X, Y ALE PRIMULUI PUNCT? 0, 0
X, Y ALE CELUI DE-AL DOILEA PUNCT? 100, 17.5
INTERPOLARE:  X = ? 455.68
              Y = 79.744
ALT PUNCT DE INTERPOLAT PE ACEEASI LINIE
(1=DA, 0=NU)? 0
ALTA LINIE DE INTERPOLAT (1=DA, 0=NU)? 0
END PROGRAM AT LINE 190
```

```
10  PRINT „Interpolare liniara"
20  PRINT
29  REM — Se introduc coordonatele X, Y pt doua puncte de pe
    linie —
30  PRINT „X, Y ale primului punct"
40  INPUT X1, Y1
50  PRINT „X, Y ale celui de-al doilea punct"
60  INPUT X2, Y2
69  REM — Se introduce coordonata X a punctului de interpo-
    lat —
70  PRINT „Interpolare: X = "
80  INPUT X
89  REM — Calculeaza coordonata Y corespunzatoare —
90  Y=Y1+(Y2-Y1)/(X2-X1)*(X-X1)
99  REM — Rotunjeste si afiseaza —
100 PRINT „                Y="; INT(Y*1000+.5)/1000
110 PRINT
120 PRINT „Alt punct de interpolat pe aceeasi linie? ((1=da
    0=nu)"
130 INPUT Z
140 PRINT
150 IF Z=1 THEN 70
159 REM — Interpolarea altei linii —
160 PRINT „Alta linie de interpolat (1=da, 0=nu)"
170 INPUT Z
180 IF Z=1 THEN 20
190 END
```

5.13 — INTERPOLARE CURBILINIARĂ

Acest program calculează coordonatele Y ale unor puncte situate pe o curbă, fiind cunoscute coordonatele X ale acelorași puncte.

Se introduc coordonatele cunoscute ale punctelor de pe curbă, puncte al căror număr nu poate depăși 50. Se poate suplimenta sau diminua această limită, modificînd instrucția 30 astfel:

```
30 DIM X(P), Y(P)
```

unde P — numărul punctelor de pe curbă cunoscute.

Interpolarea se efectuează utilizînd polinomul lui Lagrange

Exemplu: Să considerăm curba $Y = X^3 - 3X + 3$. Cunoaștem următoarele puncte: (−3, −15), (−2, 1), (−1, 5), (0, 3), (1, 1), (2, 5) și (3, 21). Care este valoarea lui Y pentru $X = -1.65$ și 0.2?

Fiind date următoarele puncte pe o curbă sinusoidală, să se determine sinusul lui −2,47 și 1.5.

(−1, 0.54)	(0.25, 0.969)
(−0.75, 0.73)	(0.5, 0.878)
(−0.5, 0.878)	(0.75, 0.73)
(−0.25, 0.969)	(1, 0.54)
(0, 1)	

```
: 30 DIM X(11), Y(11)
: RUN
INTERPOLARE CURBILINIARA
NUMARUL PUNCTELOR CUNOSCUTE? 7
X, Y PT PUNCTUL 1? −3, −15
X, Y PT PUNCTUL 2? −2,   1
X, Y PT PUNCTUL 3? −1,   5
X Y PT PUNCTUL 4? 0,     3
X, Y PT PUNCTUL 5? 1,    1
X, Y PT PUNCTUL 6? 2,    5
X, Y PT PUNCTUL 7? 3,   21
INTERPOLARE:  X=?−1.65
              Y=3.457874999999
ALT PUNCT DE INTERPOLAT (1=DA, 0=NU)? 1
INTERPOLARE:  X=? .2
              Y=2.408
ALT PUNCT DE INTERPOLAT (1=DA, 0=NU)? 0
ALTA CURBA DE INTERPOLAT (1=DA, 0=NU)? 1
NUMARUL PUNCTELOR CUNOSCUTE? 11
X, Y PT PUNCTUL 1? −5, .958
X, Y PT PUNCTUL 2? −4, .757
X, Y PT PUNCTUL 3? −3, −.141
X, Y PT PUNCTUL 4? −2, −.909
X, Y PT PUNCTUL 5? −1, −.841
X, Y PT PUNCTUL 6? 0, 0
X, Y PT PUNCTUL 7? 1, .841
X, Y PT PUNCTUL 8? 2, .909
X, Y PT PUNCTUL 9? 3, .141
X, Y PT PUNCTUL 10? 4, −.757
X, Y PT PUNCTUL 11? 5, −.959
INTERPOLARE:  X=? −2.47
              Y=.6218395970637
ALT PUNCT DE INTERPOLAT (1=DA, 0=NU)? 1
INTERPOLARE:  X=? 1.5
              Y= .9971637992869
ALT PUNCT DE INTERPOLAT (1=DA, 0=NU)? 0
ALTA CURBA DE INTERPOLAT (1=DA, 0=NU)? 0
END PROGRAM AT LINE 300
```

```
10  PRINT „Interpolare curbiliniara“
20  PRINT
28  REM — Tablourile X() și Y() limiteaza nr de puncte cunoscute
29  REM ce se afla pe curba —
30  DIM X(50), Y(50)
40  PRINT „Numarul punctelor cunoscute“
50  INPUT P
60  FOR I=1 TO P
69          REM — Introduceti coordonatele punctelor cu-
                noscute de pe curba —
70  PRINT „X, Y pentru punctul “; I
80          INPUT X(I), Y(I)
90  NEXT I
100 PRINT
109 REM — Introduceti coordonata X a punctului de interpolat—
110 PRINT „Interpolare: X=“
120 INPUT A
130 B=0
139 REM — Calculeaza coordonata Y corespunzatoare prin me-
    toda Lagrange —
140 FOR J=1 TO P
150         T=1
160         FOR I=1 TO P
170                 IF I=J THEN 190
180                 T=T*(A-X(I))/(X(J)-X(I))
190         NEXT I
200         B=B+T*Y(J)
210 NEXT J
219 REM — Afisare rezultate —
220 PRINT „          Y=“;B
230 PRINT
240 PRINT „Alt punct de interpolat pe aceeasi curba? (1=da,
0=nu)“
250 INPUT C
260 IF C=1 THEN 100
270 PRINT „Alta curba de interpolat? (1=da, 0=nu)“
280 INPUT C
290 IF C=1 THEN 20
300 END
```

5.14 — INTEGRARE PRIN METODA SIMPSON

Acest program determină valoarea aproximativă a unei integrale definite. Integrarea se face utilizînd metoda Simpson. Progarmul permite fie calcularea valorii aproximative a integralei definite cu ajutorul unei funcții, fie prin intermediul unor valori cunoscute ale funcției pe intervale definite. În ambele cazuri, se introduc limitele domeniului pe care are loc integrarea, cît și incrementul dintre punctele cuprinse între aceste limite. Dacă funcția de integrat este cunoscută, aceasta va fi introdusă în program înaintea lansării acestuia. Funcția va fi definită în linia 50. De exemplu, funcția $f(x) = X^3$ va fi introdusă astfel:

```
50 DEFFNC(X) = X↑3
```

Exemple: Să se determine integrala definită a funcției $f(x) = X^3$ pe intervalul [0, 2], cu incrementele 0.2 și 0.1.
Care este valoarea integralei definite pe intervalul [−1, 1] a unei curbe, dacă punctele cunoscute sînt următoarele:

(−1, 0.54)
(−0.75, 0.73)
(−0,5, 0.878)
(−0.25, 0.969)
(0, 1)
(0.25, 0.969)
(0.5, 0.878)
(0.75, 0.73)
(1, 0.54)

```
: 50 DEFFNC(X) = X↑3
: RUN
INTEGRARE PRIN METODA SIMPSON
SELECTATI: 1 = FORMULA CUNOSCUTA, 0 = FORMULA NECUNOSCUTA? 1
LIMITELE INFERIOARA ŞI SUPERIOARA ALE DOMENIULUI? 0,2
INCREMENT? .2
INTEGRALA ESTE 4
END PROGRAM AT LINE 310
```

```
: RUN
INTEGRARE PRIN METODA SIMPSON
SELECTATI: 1 — FORMULA CUNOSCUTA, 0 = FORMULA NECUNOSCUTA? 1
LIMITELE INFERIOARA ŞI SUPERIOARA ALE DOMENIULUI? 0,2
INCREMENT? .1
INTEGRALA ESTE 4
END PROGRAM AT LINE 310
```

```
: RUN
INTEGRARE PRIN METODA SIMPSON
SELECTIE: 1 = FORMULA CUNOSCUTA, 0 = FOR-
MULA NECUNOSCUTA? 0
LIMITELE INFERIOARA ŞI SUPERIOARA ALE DO-
MENIULUI? −1, 1
INCREMENT? .25
PRIMA, ULTIMA VALOARE A FUNCTIEI? .54, .54
VALOAREA FUNCTIEI PE INTERVALUL 1(X = −.75)?
.73
VALOAREA FUNCTIEI PE INTERVALUL 1 (X=−.75)?
.878
VALOAREA FUNCTIEI PE INTERVALUL 3 (X =−.25)?
.969
VALOAREA FUNCTIEI PE INTERVALUL 4 (X = 0)? 1
VALOAREA FUNCTIEI PE INTERVALUL 5 (X=.25)?
.969
VALOAREA FUNCTIEI PE INTERVALUL 6 (X=.5)? .078
VALOAREA FUNCTIEI PE INTERVALUL 7(X=.75); .73
INTEGRALA ESTE 1.682
END PROGRAMAT LINE 310
```

```
10  PRINT „Integrare prin metoda SIMPSON"
20  PRINT
30  PRINT „Selectati: 1=formula cunoscuta, 0=formula ne-
    cunoscuta"
40  INPUT S
49  REM — Daca functia este cunoscuta introduceti in linia 50
    DEFFNC(X) —
50  DEFFNC(X) = X
60  PRINT „Limitele inferioara si superioara a domeniului"
70  INPUT A, B
80  PRINT „Increment"
90  INPUT X1
98  REM — Incrementul trebuie sa divida intervalul in subinter-
    vale egale —
99  REM — Daca nu se schimba incrementul —
100 IF (B−A)/X1<>INT ((B−A)/X1) THEN 80
110 IF S=1 THEN 150
119 REM — Formula necunoscuta: introduceţi valorile functiei—
```

```
120  PRINT „Prima, ultima valoare a lui F(X)"
130  INPUT Y1, Y2
140  GOTO 170
149  REM — Formula cunoscuta; calcul F(X) —
150  Y1=FNC(A)
160  Y(2)=FNC(B)
170  C=0
180  D=0
189  REM REM — Ciclu pt fiecare subinterval —
190  FOR I=1 (B-A)/X1-.5
200            IF S=1 THEN 240
209  REM — Introduceţi valoarea functiei pe fiecare subinterval—
210            PRINT „Valoarea functiei pe intervalul „;1;"
               (X=„;A+I*X1;")"
220            INPUT Y
230            GOTO 250
239            REM — Calculul lui F(X) PE FIECARE SUBIN-
               TERVAL —
240            Y=FNC(A+I*X1)
249            REM — Interval par sau impar —
250            IF I/2=INT (I/2) THEN 280
260            C=C+Y
270            GOTO 290
280            D=D+Y
290            NEXT I
299  REM — Calculeaza si afiseaza integrala —
300  PRINT „Integrala este"; X1/3*(Y1+4*C+2*D+Y2)
310  END
```

5.15 — INTEGRARE PRIN METODA TRAPEZELOR

Acest program calculează valoarea aproximativă a integralei definite de o funcţie. Integrarea se face conform metodei trapezelor. Se introduc limitele domeniului de integrare şi numărul de intervale cuprinse între aceste limite.

Funcţia de integrat trebuie introdusă înaintea lansării programului în linia 30.

Exemplu: Să se determine integrala definită a funcției $f(x) = X^3$ pe intervalul [0, 2] cu 10 și 20 de intervale.

Să se determine integrala definită a funcției $f(x) = X^2$ pe intervalele [1, 2] și [2, 3] utilizînd 10 intervale.

```
: DEFFNC(X) = X↑3
: RUN
INTEGRARE PRIN METODA TRAPEZELOR
(INTRODUCETI, 0, 0 PENTRU TERMINAREA PRO-
GRAMULUI)
LIMITE DE INTEGRARE (INFERIOARA, SUPERIOA-
RA)? 0,2
NUMAR DE INTERVALE? 10
INTEGRALA = 4.04
LIMITE DE INTEGRARE (INFERIOARA, SUPERIOA-
RA)? 0,2
NUMAR DE INTERVALE; 20
INTEGRALA = 4.01
LIMITE DE INTEGRARE (INFERIOARA, SUPERIOA-
RA)? 0,0
END PROGRAM AT LINE 190
```

```
: 30 DEFFNC(X) = 1/X↑2
: RUN
INTEGRARE PRIN METODA TRAPEZELOR
(INTRODUCEȚI 0, 0 PENTRU TERMINAREA PRO-
GRAMULUI)
LIMITE DE INTEGRARE (INFERIOARĂ, SUPERI-
OARĂ)? 1, 2
NUMĂR DE INTERVALE? 10
INTEGRALA = .5014551274644
LIMITE DE INTEGRARE (INFERIOARA, SUPERI-
OARA? 2, 3
NUMĂR DE INTERVALE? 10
INTEGRALA = .16681318133
LIMITE DE INTEGRARE (INFERIOARA, SUPERI-
OARA)? 0, 0
END PROGRAM AT LINE 190
```

```
10  PRINT „Integrare prin metoda trapezelor"
20  PRINT
30  REM — Introduceti functia aici (DEFFNC(X) = „functia") —
40  PRINT „(Introduceti 0, 0 pentru terminare program)"
50  PRINT „Limitele de integrare (inferioara, superioara)"
60  INPUT A, B
69  REM — Sfirsit program ? —
70  IF A = B THEN 190
80  PRINT „Numar de intervale"
90  INPUT N
100 I = 0
109 REM — D este lungimea fiecarui interval —
110 D = (B — A)/N
119 REM — Aduna aria fiecarui trapez —
120 FOR J = A TO B STEP D
130     I = I + FNC(J)
140 NEXT J
149 REM — Calculeaza si afiseaza integrala —
149 REM — Calculeaza si afiseaza valoarea integralei —
150 I = (I — (FNC(A) + FNC(B))/2)*D
160 PRINT „Integrala = ";I
170 PRINT
180 GOTO 50
190 END
```

5.16 — INTEGRARE PRIN METODA LUI GAUSS

Acest program calculează valoarea aproximativă a integralei definite a unei funcții. Trebuie indicate limitele domeniului de integrare și numărul intervalelor cuprins între aceste limite. Intervalul de integrare este divizat în subintervale egale. Integrala definită este calculată pe fiecare subinterval cu ajutorul formulei lui Gauss. Aceste rezultate obținute pe subintervale sînt adunate pentru a se obține integrala definită pe întregul interval. Funcția de integrat va fi introdusă în linia 30, înainte ca programul să fie lansat.

Exemplu: Să se determine integrala definită a funcției $f(x) = X^3$ pe intervalul [0, 2] cu un număr de 10 și 20 de subintervale.

Să se determine integrala definită a funcției $f(x) = X^2$ pe intervalul [1, 2] utilizînd 10 subintervale.

```
: 30 DEFFNC(X) = X↑3
: RUN
INTEGRARE PRIN METODA GAUSS
LIMITE DE INTEGRARE (INFERIOARA, SUPERI-
OARA)? 0, 2
NUMAR DE INTERVALE? 10
INTEGRALA = 4.000000027887
ALTE DATE?
(0 = NU, 1 = ALTE LIMITE, 2 = ALT NR DE INTER-
VALE);? 2
NUMAR DE INTERVALE? 20
INTEGRALA = 4.000000027968
ALTE DATE?
(0 = NU, 1 = ALTE LIMITE, 2 = ALT NR DE INTER-
VALE)? 0
END PROGRAM AT LINE 320
```

```
: 30 DEFFNC(X) = 1/X↑2
: RUN
INTEGRARE PRIN METODA GAUSS
LIMITE DE INTEGRARE (INFERIOARA SUPERI-
OARA)? 1, 2
NUMAR DE INTERVALE? 10
INTEGRALA = .5000000034951
ALTE DATE?
(0 = NU, 1 = ALTE LIMITE, 2 = ALT NR DE INTER-
VALE)? 1
LIMITE DE INTEGRARE (INFERIOARA SUPERI-
OARA)? 2, 3
NUMAR DE INTERVALE? 10
INTEGRALA = .1666666678324
ALTE DATE?
(0 = NU, 1 = ALTE LIMITE, 2 = ALT NR DE INTER-
VALE)? 0
END PROGRAM AT LINE 320
```

```
10  PRINT „Integrare prin metoda GAUSS"
20  PRINT
30  REM — Introduceti aici functia (DEFFNC(X) = „functia") —
39  REM — Abscise si factori de finete pt 20 de puncte —
40  DATA .076526521, .15275339, .22778585, .14917299, .37370609
50  DATA .14209611, .510867, .13168864, .63605368, .11819453
60  DATA .74633191, .10193012, .83911697, .083276742, .91223443
70  DATA .062672048, .96397193, .04060143, .9931286, .017614007
80  PRINT „Limitele de integrare (inferioara, superioara)"
90  INPUT X, Y
100 PRINT „Numar de intervale"
110 INPUT N
120 S = (Y — X)/N/2
130 T = X + S
140 R = 0
149 REM — Calculeaza integrala pt fiecare subinterval —
150 FOR I = 1 TO N
160     P = 0
169     REM — Calculeaza suma pt fiecare interval —
170     FOR J = 1 TO 10
180         READ A, B
190         P = P + B*(FNC(S*A + T) + FNC(T — S*A))
200     NEXT J
210     RESTORE
220     R = R + P*S
230     T = T + 2*S
240 NEXT I
250 PRINT „Integrala = ";R
260 PRINT
270 PRINT „Alte date?"
280 PRINT „(0 = nu, 1 = alte limite, 2 = alt număr de intervale)"
290 INPUT S
300 IF S = 1 THEN 80
310 IF S = 2 THEN 100
320 END
```

5.17 — DERIVAREA FUNCȚIILOR

Acest program calculează derivata unei funcții date, pentru o anumită valoare a variabilei funcției. Funcția va fi introdusă în linia 30 înainte de lansarea programului.

Exemplu: Să se calculeze derivata funcției $f(x) = X^2 + \cos X$ pentru $X = -1$, $X = 0$, $X = 1$.

```
: 30 DEFFNC(X) = X↑2 + cos(X)
: RUN
DERIVAREA FUNCTIILOR
(INTRODUCEȚI X = 99999 PENTRU TERMINAREA
PROGRAMULUI)
X = ? − 1
      DERIVATA = −1.158528756224
X = ? 0
      DERIVATA = 1.53600000  E−09
X = ? 1
      DERIVATA = 1.158528979696
X = ? 99999
END PROGRAM AT LINE 160
```

```
10  PRINT „Derivarea functiilor"
20  PRINT
30  REM — Introduceti aici functia (DEFFNC(X) = „functia") —
40  PRINT „(Introduceti X = 99999 pt terminare program)'
50  PRINT „Derivare pt X = "
60  INPUT X1
69  REM — Test de sfirsit program —
70  IF X1 = 99999 THEN 160
80  D = 0
90  FOR N = 1 TO 10
100       D1 = D
110       X = X1 + .5^N
120       D = (FNC(X) − FNC(X1))/(X − X1)
130 NEXT N
139 REM — Aproximeaza si afiseaza derivata functiei in punctul
    dorit —
140 PRINT „ = "; 2*D − D1
149 REM — Restart program —
150 GOTO 50
160 END
```

5.18 — RĂDĂCINILE REALE ALE UNUI POLINOM. METODA LUI NEWTON

Acest program calculează rădăcinile reale ale polinoamelor cu coeficienți reali. Programul solicită introducerea unei aproximați pentru fiecare rădăcină a polinomului. Calculele sînt efectuate prin

metoda lui Newton de aproximare a rădăcinilor. Valoarea erorii și a derivatei sînt afișate pentru fiecare rădăcină calculată.

Gradul polinomului pe care programul îl poate rezolva este în prezent limitat la 10. Putem mări această limită modificînd instrucțiile 30 și 40 ale programului astfel:

```
30 DIM A(N + 1), B(N + 1)
40 FOR i = 1 TO N + 1
```

unde N este gradul polinomului.

Exemplu: Să se găsească o rădăcină reală a polinomului:

$$P(X) = 4X^4 - 2{,}5X^2 - X + 0{,}5$$

```
: RUN
RADACINILE REALE ALE POLINOAMELOR:
   NEWTON
GRADUL POLINOMULUI ? 4
COEFICIENT A(0) ? .5
COEFICIENT A(1) ? -1
COEFICIENT A(2) ? -2.5
COEFICIENT A(3) ? 0
COEFICIENT A(4) ? 4
ESTIMAȚIE ? -.8
RADACINA          EROARE              DERIVATA
.3035763402058    -1.40000000 E-13    -2.070247000453
ALTA VALOARE?  (1 = DA, 0 = NU)  ? 0
ALTA FUNCTIE   (1 = DA, 0 = NU)  ? 0
END PROGRAM AT LINE 550
```

```
 10  PRINT „Radacinile reale ale polinoamelor: NEWTON"
 20  PRINT
 28  REM — Dimensionati tablourile A( ) și B( ) si ciclul din
     linia 40
 29  REM la valoarea N + 1 (N nr gradelor de libertate)
 30  DIM A (11), B(11)
 40  FOR I = 1 TO 11
 50      A(I) = 0
 60      B(I) = 0
 70  NEXT I
 80  PRINT „Gradul polinomului"
 90  INPUT N
100  FOR I = 1 TO N + 1
109      REM — coeficientii in ordinea crescatoare a gradelor —
```

```
110     PRINT „Coeficientul a(„; I−1;")"
120     INPUT A(I)
130 NEXT I
140 FOR I = 1 TO 10
149     REM — Calculul coeficientilor derivatei polinomului —
150     B(I) = A(I + 1)*I
160 NEXT I
170 PRINT
179 REM — Initializare factor de estimatie —
180 PRINT „Estimatie"
190 INPUT X
200 Q = 0
210 S = 1
220 F1 = 0
230 F0 = 0
239 REM — Contor de iteratii —
240 Q = Q + 1
250 FOR I = 1 TO N + 1
259     REM — Calculeaza valoarea functiei —
260     F0 = F0 + A(I)*S
269     REM — Calculeaza valoarea derivatei —
270     F1 = F1 + B(I)*S
280     S = S*X
290 NEXT I
299 REM — Testeaza derivata in punctul 0 —
300 IF F1 = 0 THEN 360
310 S = X − F0/F1
320 IF X = S THEN 380
330 X = S
340 IF Q > 100 THEN 490
350 GOTO 210
360 PRINT „Derivata = 0 in punctul X = "; X
370 GOTO 180
380 PRINT
390 PRINT „Radacina", „Eroare", „Derivata"
400 PRINT X, F0, F1
410 PRINT
420 REM — Alta radacina pe aceeasi functie ? —
425 PRINT „Alta valoare (1 = da, 0 = nu)"
430 INPUT A
440 IF A = 1 THEN 170
449 REM — Restart sau sfirsit program —
450 PRINT „Alta functie (1 — da, 0 = nu)"
```

```
460 INPUT A
470 IF A = 1 THEN 30
480 GOTO 550
489 REM — Afisarea valorilor calculate dupa 100 de iteratii;
    cauta urmatoarele 100 ?
490 PRINT „100 iteratii complete:"
500 PRINT „X = ";X;" F(X) = ";F0
510 PRINT „Continuati (1 = da, 0 = nu) ?"
520 INPUT A
530 IF A = 1 THEN 200
540 GOTO 420
550 END
```

5.19 — RĂDĂCINILE POLINOAMELOR. METODA DIHOTOMIEI

Acest program determină rădăcinile polinoamelor în interiorul unui interval dat. Inițial programul efectuează o căutare aleatoare a două puncte de semn contrar în intervalul dat. Dacă este detectată o schimbare de semn, rădăcina este calculată prin metoda dihotomiei. Dacă nu este detectată nici o astfel de schimbare de semn, se solicită un alt interval.

Erorile ce pot surveni în execuția acestui program se pot datora următoarelor cauze:

— detectarea unei rădăcini într-un punct în care polinomul nu are rădăcină; acest lucru poate apărea atunci cînd punctul din extremitatea stîngă a intervalului este atît de aproape de zero încît el este forțat la zero printr-o eroare de rotunjire sau trunchiere;

— două rădăcini ale unui polinom pot fi atît de apropiate între ele încît, prin eroare de rotunjire sau trunchiere programul să le confunde și să nu găsească între ele nici o schimbare de semn; în acest caz nu va fi detectată nici o rădăcină.

Polinomul trebuie introdus în linia 30 înainte de lansarea programului.

Exemplu: Să se determine o rădăcină a polinomului $f(x) = 4X^4 - 2{,}5X^2 - X + 0{,}5$.

```
10 PRINT „Radacinile polinoamelor: dihotomie"
20 PRINT
30 REM — Introduceti functia (DEFFNR(X) = „functia") —
40 DIM D(3)
50 PRINT „(Pentru terminarea programului introduceti 0, 0)"
```

```
 59  REM — Stabilire interval pt cautare aleatoare —
 60  PRINT „Interval (limite inferioara, superioara)"
 70  INPUT A, B
 79  REM — Test limite —
 80  IF A<>B THEN 120
 89  REM — Sfirsit program ? —
 90  IF A = 0 THEN 430
100  PRINT „— Limitele intervalului nu pot fi egale —"
110  GOTO 60
120  IF A<B THEN 150
130  PRINT „— Limita inferioara trebuie introdusa prima —"
140  GOTO 60
150  A1 = SGN(FNR(A))
160  B1 = SGN(FNR(B))
169  REM — Test de radacina la fiecare limita —
170  IF A1*B1 = 0 THEN 360
179  REM — Test de schimbare de semn la limitele intervalului —
180  IF A1*B1<0 THEN 280
189  REM — Ciclu de cautare a 1000 de numere cu semn opus —
190  FOR I = 1 TO 1000
200      X = A + RND(2)*(B — A)
210      X1 = SGN(FNR(X))
220      IF X1 = 0 THEN 400
230      IF A1*X1<0 THEN 270
240  NEXT I
250  PRINT „Nu s-au detectat schimbari de semn pe acest interval"
260  GOTO 60
269  REM — A fost gasita o schimbare de semn —
270  B = X
278  REM — Depune punctul pozitiv in D(3) punctul negativ
     in D(1)
279  REM — D(1) si D(3) devin limitele intervalului —
280  D(2 + A1) = A
290  D(2 — A1) = B
299  REM — Calculeaza mijlocul intervalului —
300  Y = D(1) + D(3))/2
310  Y1 = SGN(FNR(Y))
319  REM — Test de radaina la mijlocul intervalului —
320  IF Y1 = 0 THEN 400
330  D(2 + Y1) = Y
340  IF ABS(D(1) — D(3))/ABS(D(1) + ABS(D(3))<5E — 6
     THEN 400
349  REM — Reluare test cu alte limite —
```

```
350  GOTO 300
359  REM — Exista radacina la limita intervalului. Care limita ? —
360  IF A1 = 0 THEN 390
370  Y = B1
380  GOTO 400
390  Y = A1
400  PRINT „Radacina = “;Y
410  PRINT
419  REM — Restart program —
420  GOTO 60
430  END
```

```
: 30 DEFFNC(X) = 4*X↑4 — 2.5*X↑2 — X + .5
: RUN
RADACINILE POLINOAMELOR: DIHOTOMIE
(PENTRU TERMINAREA PROGRAMULUI INTRODUCEȚI 0, 0)
INTERVAL (LIMITE INFERIOARA, SUPERIOARA) ?
—1, 0
NU S-AU DETECTAT SCHIMBĂRI DE SEMN PE ACEST INTERVAL
INTERVAL (LIMITE INFERIOARA, SUPERIOARA) ?
0, 1
RADACINA = .3035792010268
INTERVAL (LIMITE INFERIOARA, SUPERIOARA) ?
0, 0
END PROGRAM AT LINE 430
```

5.20 — POLINOAME TRIGONOMETRICE

Acest program calculează valoarea unei funcții trigonometrice pentru un unghi dat. Funcția trebuie să se prezinte sub următoarea formă:

$$f(x) = A_1 \sin(X) + B_1 \cos(X) + A_2 \sin(2X) + B_2 \cos(2X) + \ldots + A_n \sin(nX) + B_n \cos(nX)$$

unde n = numărul perechilor de coeficienți.

Coeficienții funcției trebuie introduși cu instrucții DATA începînd din linia 30. Instrucția DATA trebuie să conțină numărul perechilor de coeficienți (n) și apoi coeficienții funcției.

```
30 DATA n, A1, B1, A2, B2, ..., An, Bn
```

Exemplu: Să se determine valoarea funcției de mai jos pentru unghiuri egale cu respectiv 45°, 90°, 105°.

$$f(x) = \sin(X) + 2\ \cos(X) - 2\ \sin(2X) + \cos(2X) + 5\ \sin(3X) - 3\ \cos(3X).$$

```
: 30 DATA 3, 1, 2, -2, 1, 5, -3
: RUN
POLINOAME TRIGONOMETRICE
(INTRODUCETI UNGHI = 99999 PT TERMINARE
    PROGRAM)
UNGHI ? 45
F(45) = 3.095587494888
UNGHI ? 90
F(90) = -2.831680950826
UNGHI ? 105
F(105) = -1.5468488370549
UNGHI ? 99999
END PROGRAM AT LINE 180
```

```
 10  PRINT „Polinoame trigonometrice"
 20  PRINT
 30  REM — Introduceti nr de perechi ai termenilor si coeficientilor —
 40  PRINT „(Introduceti unghi = 99999 pt terminare program)"
 50  PRINT „UNGHI:"
 60  INPUT R
 69  REM — Sfirsit program —
 70  IF R = 99999 THEN 180
 79  REM — Citeste nr de perechi de termeni in polinom —
 80  READ N
 89  REM — Ciclu pt citirea coeficientilor —
190  FOR I = 1 TO N
100      READ A, B
110      Z = Z + A*SIN(I*R) + B*COS(I*R)
120  NEXT I
129  REM — Afisare rezultate —
130  PRINT „F(„;R;") = ";Z
139  REM — Reluare citire coeficienti functie —
140  RESTORE
150  PRINT
160  Z = 0
169  REM — Restart program —
170  GOTO 50
180  END
```

5.21 — ECUAŢII LINIARE

Acest program rezolvă un sistem de ecuaţii liniare. Numărul necunoscutelor trebuie să fie egal cu numărul ecuaţiilor din sistem. Programul solicită introducerea coeficienţilor fiecărei ecuaţii a sistemului. Instrucţia din linia 30 a programului limitează numărul de ecuaţii ale sistemului de rezolvat. Se poate modifica această instrucţie în maniera următoare:

```
30 DIM A(R, R + 1)
```

unde R = numărul maxim de ecuaţii ale sistemului.

Exemplu: Să se rezolve următorul sistem de ecuaţii:

$$\begin{cases} X_1 + 2X_2 + 3X_3 = 4 \\ 3X_1 + 6X_2 \quad\quad = 1 \\ -3X_1 + 4X_2 - 2X_3 = 0 \end{cases}$$

```
: 30 DIM A(3, 4)
: RUN
REZOLVAREA SISTEMELOR DE ECUATII LINIARE
NUMARUL ECUATIILOR ? 3
MATRICEA COEFICIENTILOR:
ECUATIA 1
   COEFICIENT 1 ? 1
   COEFICIENT 2 ? 2
   COEFICIENT 3 ? 3
   CONSTANTA ? 4
ECUATIA 2
   COEFICIENT 1 ? 3
   COEFICIENT 2 ? 6
   COEFICIENT 3 ? 0
   CONSTANTA ? 1
ECUATIA 3
   COEFICIENT 1 ? -3
   COEFICIENT 2 ? 4
   COEFICIENT 3 ? -2
   CONSTANTA ? 0
X1 = -.356
X2 =  .344
X3 =  1.222
END PROGRAM AT LINE 440
```

```
10  PRINT „Rezolvarea sistemelor de ecuatii liniare“
20  PRINT
29  REM — Se rezerva A(R, R+1) unde R=nr max de ecuatii —
30  DIM A(9, 10)
40  PRINT „Numarul ecuatiilor“
50  INPUT R
60  PRINT „Matricea coeficientilor“
70  FOR J = 1 TO R
80      PRINT „Ecuatia“;J
90      FOR I = 1 TO R + 1
100         IF I = R + 1 THEN 130
110         PRINT „Coeficient“;I
120         GOTO 140
130         PRINT „Constanta“
140       INPUT A(J, I)
150     NEXT I
160 NEXT J
170 FOR J = 1 TO R
178     REM — Instructiile 180—220 determina prima ecuatie —
179     REM — cu coeficientul <>0 pe coloana curenta —
180     FOR I = J TO R
190         IF A(I, J)<>0 THEN 230
200     NEXT I
210     PRINT „Sistemul nu are solutie unica“
220 GOTO 440
229 REM — Instructiile 230—270 transfera ecuatia pe linia
      curenta —
230     FOR K = 1 TO R + 1
240           X = A(J, K)
250     A(J, K) = A(I, K)
260       A(I, K) = X
270     NEXT K
279     REM — Instructiile 280—310 pun 1 in prima coloana
        <>0 pe linia curenta —
280     Y = 1/A(J, J)
290     FOR K = 1 TO R + 1
```

```
300          A(J, K) = Y*A(J, K)
310       NEXT K
319       REM — Instructiile 320—380 scad ecuatia curenta din
          celelalte linii —
320       FOR I = 1 TO R
330          IF I = J THEN 380
340          Y = —A(I, J)
350          FOR K = 1 TO R + 1
360             A(I, K) = A(I, K) + Y*A(J, K)
370          NEXT K
380       NEXT I
389       REM — Secventa se executa pt fiecare ecuatie —
390    NEXT;
400    PRINT
409    REM — Afisare solutii —
410    FOR I = 1 TO R
420       PRINT „X";I;" = ";INT(A(I, R + 1)*1000 + .5)/1000
430    NEXT I
440    END
```

5.22 — PROGRAMARE LINIARĂ

Acest program rezolvă o problemă de programare liniară utilizînd algoritmul simplex. Programul solicită introducerea coeficienților funcției obiectiv împreună cu coeficienții, relația și constanta fiecărei restricții. Toate aceste informații sînt introduse prin intermediul instrucțiilor DATA înainte de lansarea programului. După încărcarea programului în memorie, introduceți instrucțiile DATA respectînd regulile prezentate în continuare. Dacă doriți să rezolvați mai multe probleme de programare liniară, nu uitați să ștergeți, în prealabil, toate instrucțiile DATA ale problemei precedente. Instrucțiile DATA vor începe întotdeauna în linia 3000.

1) Aranjați restricțiile problemei de programare liniară în așa manieră încît inegalitățile de tip „mai mic decît" să preceadă egalitățile, care, la rîndul lor, să preceadă inegalitățile de tip „mai mare decît".

2) Introduceți atîtea instrucții DATA cîte restricții conține problema de programare liniară de rezolvat. Fiecare instrucție DATA conține coeficienții unei restricții, în ordinea în care acestea

apar în problemă, ținînd cont de punctul 1. Nu vor fi incluși, în aceste instrucțiii, coefi cienții variabilelor libere, ai variabilelor de surplus sau ai variabilelor artificiale. Pentru fiecare variabilă care nu apare într-o restricție dată, se va pune coeficientul zero.

3) Introduceți instrucțiii DATA pentru coeficienții vectorului termenilor liberi. Numărul acestor coeficienți trebuie să fie egal cu numărul restricțiilor problemei. Valorile acestor coeficienți nu pot fi negative.

4) Introduceți instrucții DATA pentru coeficienții funcției obiectiv. Trebuie să indicați dacă problema este o problemă de minimizare sau de maximizare. Programul solicită, de asemenea introducerea numărului total de restricții, numărului de variabile, numărului de restricții de tip „mai mic decît", al celor de tip „egal" și al celor de tip „mai mare decît".

Instrucția DIM din linia 20 limitează numărul de variabile și restricții ce poate fi introd us. Aceste limite pot fi modificate astfel:

```
20 DIM A(C ⊕ 2, V + C + 1), B(C + 2)
```

unde: C = numărul de restricții
V = numărul de variabile.

Exemplu: O fabrică trebuie să producă 100 kg dintr-un aliaj care conține 83% plumb, 14% cupru și 3% antimoniu. Pentru aceasta fabrica dispune de cinci aliaje cu compoziția și prețurile indicate în următorul tablou. Cum trebuie combinate aceste aliaje pentru a obține produsul dorit la un cost minim?

	aliaj 1	aliaj 2	aliaj 3	aliaj 4	aliaj 5
Pb.	90	80	195	70	30
Cu	5	5	2	30	70
A	5	15	3	0	0
Unități monetare	6.13	7.12	5.85	4.57	3.96

Se obține următoarea problemă de programare liniară:

$X_1 \oplus X_2 \oplus X_3 \oplus X_4 + X_5 = 100$

$.90X_1 + .80X_2 \oplus .95X_3 \oplus .70X_4 + .30X_5 = 83$

$.05X_1 + .05X_2 \oplus .02X_3 \oplus 30.X_4 + .70X_5 = 14$

$.05X_1 \oplus .15X_2 \oplus .03X_3 = 3$

$6.13X_1 \oplus 7.12X_2 \oplus 5.85X_3 \oplus 4.57X_4 \oplus 3.96X_5 = Z$ (min)

```
: 3000 DATA 1, 1, 1, 1, 1
: 3010 DATA .9, .8, .95, .7, .3
: 3020 DATA .05, .05, .02, .3, .7
: 3020 DATA .05, .15, .03, .0,0
: 3040 DATA 100, 83, 14, 3
: 3050 DATA 6.13, 7.12, 5.85, 4.57, 3.96
: RUN
PROGRAMARE LINIARA
INTRODUCEŢI „1" PENTRU MAXIMIZARE, „—1" PT
MINIMIZARE ? — 1
INTRODUCEŢI NR DE RESTRICTII, NUMAR DE
VARIABILE ? 4, 5
NR DE RESTRICTII < =, =, > = ? 0, 4, 0
VARIABILE PROPRII DE LA 1 LA 5
VARIABILE ARTIFICIALE DE LA 6 LA 9
RASPUNSURI:
VARIABILE PRIMARE:
VARIABILE      VALOARE
   2           10.4347826087
   3           47.82608695654
   4           41.73913043478
VARIABILE DUALE:
VARIABILE    VALOARE
VALOAREA FUNCT IEI OBIECTIV 544.8260869565
END PROGRAM AT LINE 3060
```

```
10  PRINT „Programare liniara"
15  PRINT
19  REM — Programare liniara metoda SIMPLEX —
20  DIM A(6, 10), B(6)
30  PRINT
40  PRINT „Introduceti 1' pt maximizare" — 1' pt minimizare
50  INPUT Z
60  Z = — Z
70  PRINT „Introducèti nr de restrictii, nr de variabile"
80  INPUT M, N
90  PRINT „Numar de restrictii de tip <,=,>"
100 INPUT L, E, G
110 IF M = L + E + G THEN 140
120 PRINT „Date eronate. Reintroduceti tipul restrictiilor"
130 GOTO 90
140 C = N + M + G
```

```
150  C1 = C + 1
160  C2 = N + L + G
170  M1 = M + 1
180  M2 = M + 2
190  PRINT
200  FOR I = 1 TO M2
210          FOR J = 1 TO C1
220          A(I, J) = 0
230      NEXT J
240  NEXT I
250  FOR I = 1 TO M
260      B(I) = 0
270  NEXT I
280  FOR I = 1 TO M
290      FOR J = 1 TO N
300          READ A(I, J)
310          IF I < = L THEN 330
320          A(M1,J) = A(M1, J) - A(I, J)
330  NEXT J
340  IF I>L THEN 380
350      B(I) = N + I
360      A(I, N + I) = 1
370      GOTO 440
380      B(I) = N + G + I
390      A(I, J + G + I) = 1
400      IF I>L + E THEN 420
410      GOTO 440
420      A(I, N + I - E) = -1
430      A(M1, N + I - E) = 1
440  NEXT I
450  FOR I = 1 TO M
460      READ A(I, C1)
470  NEXT I
480  FOR J = 1 TO N
490      READ A(M2, J)
500      A(M2, J) = Z*A(M2, J)
510  NEXT J
520  PRINT
530  P1 = 1
540  PRINT „Variabile proprii „;P1;" - „;N
550  IF L = 0 THEN 570
560  PRINT „Variabile de ecart ";N + 1" - ";N + L
570  IF G = 0 THEN 590
```

```
580  PRINT „Variabile suplimentare“ ;N + L + 1;“ − “;C
590  IF L = M THEN 770
600  PRINT „Variabile artificiale“; C2 + 1;“ − “;C
610  M3 = M1
620  GOSUB 1040
630  PRINT
640  FOR I1 = 1 TO M
650      IF B(I1) < = C2 THEN 760
660      IF A(I1, C1) < = .00001 THEN 690
670      PRINT „Problema nu are solutii admisibile“
680      GOTO 3060
690      FOR J1 = 1 TO C2
700          IF ABS(A(I1, J1)) < = .00001 THEN 750
710          R = I1
720          S = J1
730          GOSUB 1260
740          J1 = C2
750      NEXT J1
760  NEXT I1
770  P1 = 2
780  PRINT
790  M3 = M2
800  GOSUB 1040
830  PRINT
840  PRINT „Raspunsuri:“
850  PRINT „Variabile primale:“
860  PRINT „Variabila:“, „Valoare:“
870  FOR J = 1 TO C2
880      FOR I = 1 TO M
890          IF B(I)<>J THEN 920
900          PRINT J, A(I, C1)
910          I = M
920      NEXT I
930  NEXT J
940  PRINT „Variabile duale:“
950  PRINT „Variabila:“, „Valoare:“
960  IF L = 0 THEN 1000
970  FOR I = 1 TO L
980      PRINT I, −Z*A(M2, N + 1)
990  NEXT I
1000 PRINT „Valoarea functiei obiectiv =“; −Z*A(M2, C1)
1010 PRINT
1020 PRINT
1030 GOTO 3060
```

```
1038  REM — Rutina de optimizare —
1040  P = -.00001
1050  FOR J = 1 TO C2
1060      IF A(M3, J) > = P THEN 1090
1070      S = J
1080      P = A(M3, J)
1090  NEXT J
1100  IF P = -.00001 THEN 1440
1110  GOSUB 1130
1120  GOSUB 1210
1125  GOTO 1040
1129  REM — Determina variabila care paraseste baza —
1130  Q = 1.E + 38
1140  FOR I = 1 TO M
1150      IF A(I, S) < = .00001 THEN 1190
1160      IF A(I, C1)/A(I, S) = Q THEN 1190
1170      R = I
1180      Q = A(I, C1)/A(I, S)
1190  NEXT I
1200  RETURN
1210  IF Q = 1.E + 38 THEN 1240
1220  GOSUB 1260
1230  RETURN
1240  PRINT „Problema are optim infinit“
1250  GOTO 3060
1259  REM — Pivotare —
1260  P = A(R, S)
1270  FOR I = 1 TO M2
1280      IF I = R THEN 1350
1290      FOR J = 1 TO C1
1300          IF J = S THEN 1340
1310          A(I, J) = A(I, J) - A(I, S)*A(R, J)/P
1320          IF ABS(A(I, J)) > = .00001 THEN 1340
1330          A(I, J) = 0
1340      NEXT J
1350  NEXT I
1360  FOR J = 1 TO C1
1370      A(R, J) = A(R, J)/P
1380  NEXT J
1390  FOR I = 1 TO M2
1400      A(I, S) = 0
1410  NEXT I
1420  A(R, S) = 1
```

```
1430  B(R) = S
1440  RETURN
2996  REM — *** Executati urmatoarele operatii inainte de
      executia programului:
2997  REM — Introduceti coeficientii restrictiilor <,=,> in
      instructia DATA CARE INCEPE IN 3000
2998  REM — In linia imediat urmatoare restrictiilor introduceti
      coeficientii functiei obiectiv
2999  REM — In linia urmatoare introduceti coeficientii functiei
      obiectiv
3000  DATA 1, 1, 1, 1, 1
3010  DATA .9, .8, .95, .7, .3
3020  DATA .05, .05, .02, .3, .7
3030  DATA .05, .15, .03, 0, 0
3040  DATA 100, 83, 14, 3
3050  DATA 6.13, 7.12, 5.85, 4.57, 3.96
3060  END
```

5.23 — PERMUTĂRI ŞI COMBINĂRI

Acest program calculează numărul de permutări de N elemente şi numărul combinărilor de n elemente luate cîte K. Evident este necesară condiţia $K \leqslant n$.

Exemplu: În cîte feluri putem permuta cele 26 de litere ale alfabetului şi cîte combinări putem efectua cu aceste litere luîndu-le cîte cinci? În cîte feluri se pot aşeza pe o bancă 12 persoane, ştiind că pe această bancă nu încap decît trei persoane?

```
: RUN
PERMUTARI ŞI COMBINARI
(INTRODUCETI 0 PENTRU TERMINARE PROGRAM)
NR. TOTAL DE ELEMENTE ? 26
NR. DE ELEMENTE DIN SUBGRUPA ? 5
   7893600 PERMUTARI
   65780 COMBINARI
NR. TOTAL DE ELEMENTE ? 12
NR. DE ELEMENTE DIN SUBGRUPA ? 2
   132 PERMUTARI
   66 COMBINARI
NR. TOTAL DE ELEMENTE ? 0
END PROGRAM AT LINE 280
```

```
 10  PRINT „Permutari si combinari"
 20  PRINT
 30  PRINT „(Introduceti 0 pt terminare program)"
 40  PRINT „Numar total de elemente"
 50  INPUT  N
 59  REM — Test de sfirsit program —
 60  IF N = 0 THEN 280
 70  PRINT „Nr de elemente din subgrupa"
 80  INPUT  D
 89  REM — Dimensiunea subgrupului nu poate depasi dimen-
     siunea grupului —
 90  IF  D < = N  THEN  130
100  PRINT „Subgrup prea mare"
110  PRINT
120  GOTO 40
129  REM — Calculul permutarilor —
130  P = 1
140  C = 1
150  FOR  I = N — D + 1  TO  N
190  P = P*I
200  NEXT I
209  REM — Calculeaza factorialul intermediar pt combinari
210  For  J = 2  TO  D
220       C = C*J
230  NEXT  J
240  PRINT P; „Permutari"
250  PRINTP/C;  „Combinari"
260  PRINT
269  REM — Restart program —
270  GOTO 40
280  END
```

5.24 — TESTUL U AL LUI MANN-WHITNEY

Acest program efectuează testul U al lui Mann-Whitney asupra unui eşantion alcătuit din două populaţii.

Instrucţia din linia 30 limitează dimensiunea eşantionului. Această dimensiune poate fi modificată astfel:

```
30  DIM  X(M),  Y(N)
```

unde: M = dimensiunea maximă a primei populaţii;
N = dimensiunea maximă a celei de-a doua populaţii.

Exemplu: S-au format două eșantioane alcătuite din cîte 10 persoane cărora li s-a cerut să acorde note cuprinse între 1 și 10, unor produse ale industriei ușoare. Tabela de mai jos conține rezultatele. Să se efectueze testul U asupra acestor date și să se determine de cîte ori notația primului eșantion este inferioară celui de-al doilea și invers.

1	3	4	3	6	8	9	7	8	4
7	9	8	5	10	9	10	6	5	2

```
: 30 DIM X(10), Y(10)
: RUN
TESTUL U MANN-WHITNEY
ESANTION 1:
    MARIME ESANTION ? 10
        DATA 1 ? 1
        DATA 2 ? 3
        DATA 3 ? 4
        DATA 4 ? 3
        DATA 5 ? 6
        DATA 6 ? 8
        DATA 7 ? 9
        DATA 8 ? 7
        DATA 9 ? 8
        DATA 10 ? 4
ESANTION 2:
    MARIME ESANTION ? 10
        DATA 1 ? 7
        DATA 2 ? 9
        DATA 3 ? 8
        DATA 4 ? 5
        DATA 5 ? 10
        DATA 6 ? 9
        DATA 7 ? 10
        DATA 8 ? 6
        DATA 9 ? 5
        DATA 10 ? 2
PRIMUL ESANTION, U = 70
AL DOILEA ESANTION, U = 30
END PROGRAM AT LINE 710
```

```
 10  PRINT „Testul U MANN-WHITNEY“
 20  PRINT
 28  REM — Se rezerva X(M), Y(N) unde M = lungimea esantio-
     nului 1
 29  REM    N = LUNGIMEA ESANTIONULUI 2 —
 30  DIM X(25), Y(25)
 40  DIM N(2)
 49  REM — Introduceti cele doua esantioane —
 50  FOR I = 1 TO 2
 60     PRINT „Esantion“; I;“ :“
 70     PRINT „Dimensiune“
 80     INPUT N(I)
 90     FOR J = 1 TO N(I)
100        PRINT „Data“; J
110        INPUT Y(J)
120     NEXT J
129  REM — Sortare esantioane —
130     FOR J = 1 TO N(I)
140        FOR K = 1 TO N(I) — J
150        C = Y(K)
160        D = Y(K + 1)
170        IF Y(K) < Y(K + 1) THEN 200
180        Y(K) = Y(K + 1)
190        Y(K + 1) = C
200        NEXT K
210     NEXT J
220     PRINT
229  REM — Se transfera primul esantion in vectorul X —
230     IF I = 2 THEN 270
240     FOR J = 1 TO N(1)
250        X(J) = Y(J)
260     NEXT J
270  NEXT I
280  R = 1
290  I = 0
300  J = 0
310  I = I + 1
320  J = J + 1
330  IF I > N(1) THEN 580
340  IF J > N(2) THEN 620
350  IF X(I) < Y(J) THEN 620
```

```
360  IF Y(J) < X(I) THEN 590
370  K = 2
380  M = I
390  L = J
400  R1 = 2*R + 1
410  R = R + 2
420  I = I + 1
430  J = J + 1
440  IF I > N(1) THEN 480
450  IF X(I)<>X(I - 1) THEN 480
460  I = I + 1
470  GOTO 510
480  IF J > N(2) THEN 550
490  IF Y(J)<>Y(J - 1) THEN 550
500  J = J + 1
510  R1 = R1 + R
520  R = R + 1
530  K = K + 1
540  GOTO 440
550  X = X + (I - M)*R1/K
560  Y = Y + (J - L)*R1/K
570  GOTO 330
580  IF J > N(2) THEN 660
590  Y = Y + R
600  J = J + 1
610  GOTO 640
620  X = X + R
630  I = I + 1
640  R = R + 1
650  GOTO 330
660  U1 = N(1)*N(2) + N(1)*(N(1) + 1)/2 - X
670  U2 = N(1)*N(2) + N(2)*(N(2) + 1)/2 - Y
680  PRINT
690  PRINT „Primul esantion, U = "; U1
700  PRINT „Al doilea esantion, U ="; U2
710  END
```

5.25 — MEDIE, VARIAȚIE, ABATERE STANDARD

Acest program calculează media aritmetică, variația și abaterea standard a unor date statistice grupate sau negrupate. Datele pot reprezenta o întreagă populație sau numai un eșantion.

Exemplu: În holul unui hotel se află 10 persoane în vîrstă de 87, 53, 35, 42, 9, 48, 51, 60, 39 și 44 ani. Care va fi media, variația și abaterea standard a vîrstelor întregii populații, utilizînd persoanele din hol drept eșantion.

```
: RUN
MEDIA, VARIATIA, ABATEREA STANDARD
CE METODA (0 = POPULATIE, 1 = ESANTION) ? 1
TIPUL DATELOR (0 = GRUPATE, 1 = NEGRUPATE) ?1
NUMAR DE OBSERVATII ? 10
ELEMENT 1 ? 87
ELEMENT 2 ? 53
ELEMENT 3 ? 35
ELEMENT 4 ? 42
ELEMENT 5 ? 9
ELEMENT 6 ? 48
ELEMENT 7 ? 51
ELEMENT 8 ? 60
ELEMENT 9 ? 35
ELEMENT 10 ? 44
MEDIA          VARIATIA      ABATEREA STANDARD
46.9           389.7333      19.741664908
CONTINUATI (0 = NU, 1 = DA) ? 0
END PROGRAM AT LINE 380
```

```
 10  PRINT „Media, variatia, abaterea standard"
 20  PRINT
 30  PRINT „Ce metoda (0 = populatie, 1 = esantion) ?"
 40  INPUT S
 50  PRINT „Tipul datelor (0 = grupate, 1 = negrupate)"
 60  INPUT K
 70  PRINT „Nr de observatii"
 80  INPUT N
 90  R = 0
100  M = 0
110  P = 0
120  IF K = 1 THEN 230
129  REM – Date grupate –
130  FOR I = 1 TO N
140     PRINT „Element, frecventa": I
```

```
150     INPUT A, B
159     REM — Acumulare valori introduse —
160     R = R + B*A
169     REM — Se acumuleaza valori intermediare pt variatie —
170     P = P + B
180     M = M + B*A^2
190 NEXT I
199 REM — Calculeaza media si variatia —
200 R = R/P
210 V = (M — P*R^2)/(P — S)
219 REM — Afisare rezultate —
220 GOTO 310
229 REM — Date negrupate —
230 FOR I = 1 TO N
240     PRINT „Element"; I
250     INPUT D
259     REM — Acumulare valori introduse —
260     P = P + D
270     M = M + D^2
280 NEXT I
289 REM — Calculeaza media, variatia si le afiseaza —
290 R = P/N
300 V = (M — N*R^2)/(N — S)
310 PRINT
319 REM — Afiseaza rezultatele —
320 PRINT „Media", „Variatia", „Abaterea standard"
330 PRINT R, V, SQR(V)
340 PRINT
349 REM — Restart program —
350 PRINT „Continuati (1 = da, 0 = nu)"
360 INPUT S
370 IF S = 1 THEN 20
380 END
```

5.26 — MEDIA ŞI ABATEREA GEOMETRICĂ

Acest program calculează media şi abaterea geometrică pentru un ansamblu de date.

Exemplu: Să se determine media și abaterea geometrică pentru următoarele date: 3, 5, 8, 3, 7, 2.

```
: RUN
MEDIA ŞI ABATEREA GEOMETRICA
(PT TERMINARE PROGRAM INTRODUCETI 0
OBSERVATII)
NUMAR DE OBSERVAŢII ? 6
DATA 1 ? 3
DATA 2 ? 5
DATA 3 ? 8
DATA 4 ? 3
DATA 5 ? 7
DATA 6 ? 2
MEDIA GEOMETRICA = 4.140680833732
ABATEREA GEOMETRICA = 1.723689564961
NUMAR DE OBSERVAŢII ? 0
END PROGRAM AT LINE 200
```

```
 10  PRINT „Media si abaterea geometrica"
 20  PRINT
 30  PRINT „(Pt terminare program introduceti 0 observatii)"
 40  PRINT „Nr de observatii"
 50  INPUT N
 59  REM — Test de sfirsit program —
 60  IF N = 0 THEN 200
 70  P = 1/N
 80  M = 1
 90  FOR I = 1 TO N
100  PRINT „Data"; I
110  INPUT D
119  REM — Calculul iterativ al mediei —
120  M = M*D^P
130  Q = Q + LOG(D) 2
140  NEXT I
150  R = EXP(SQR(Q/(N — 1) — (N/(N — 1)*(LOG(M))^2)))
160  PRINT „Media geometrica ="; M
170  PRINT „Abaterea geometrica ="; R
180  PRINT
189  REM — Restart program —
190  GOTO 40
200  END
```

5.27 — DISTRIBUȚIA BINOMIALĂ

Acest program calculează probabilitatea de a se obține un număr dat de reușite în situația efectuării unui număr dat de experimente de tip Bernoulli. Este necesar să se indice probabilitatea de reușită a unui singur experiment.

Exemplu: Care este probabilitatea de a obține de trei ori stema efectuînd cinci aruncări cu o monedă?

Care este probabilitatea ca din cinci aruncări cu zarul să obținem de 2 ori cifra 1?

```
: RUN
DISTRIBUTIA BINOMIALA
(INTRODUCEȚI 0 PENTRU TERMINARE PROGRAM)
NUMAR DE EXPERIMENTE ? 5
NUMAR DE REUSITE ? 3
PROBABILITATEA DE REUSITĂ ? .5
PROBABILITATEA A 3 REUSITE IN 5 EXPERIMENTE
= .312499999998
NUMAR DE EXPERIMENTE ? 5
NUMAR DE REUSITE ? 2
PROBABILITATEA DE REUSITĂ ? .166666667
PROBABILITATEA A 2 REUSITE IN 5 EXPERIMENTE
= .1607510292571
NUMAR DE EXPERIMENTE ? 0
END PROGRAM AT LINE 270
```

```
10 PRINT „Distributia binomiala"
20  PRINT
30  DIM  M(3)
40  PRINT „(Introduceti 0 pt terminare program)"
50  PRINT „Numar de experiente"
60  INPUT N
70  IF N = 0 THEN 270
80  PRINT „Numar de reusite"
90  INPUT X
100  PRINT „Probabilitatea de reusita"
110  INPUT P
119  REM — Calculul factorialului —
120  M(1) = N
130  M(2) = X
140  M(3) = N — X
```

```
150  FOR I = 1 TO 3
160     IF M(I) = 0 THEN 220
170     A = 1
180     FOR J = 1 TO M(I)
190        A = A*J
200     NEXT J
210     M(I) = LOG(A)
220  NEXT I
229  REM — Calculul probabilitatii —
230  R = EXP(M(1) — M(2) — M(3) + X * LOG(P) + (N — X) *
     * LOG(1 — P))
240  PRINT „Probabilitatea a"; X;" reusite in ";N;" incercari
     = ";R
250  PRINT
260  GOTO 50
270  END
```

5.28 — DISTRIBUȚIA POISSON

Acest program calculează probabilitatea ca un eveniment să apară de un număr de ori dat, utilizînd distribuția Poisson. Este necesar să se indice frecvența calculată a evenimentului așteptat.

Exemplu: Un număr de 2 000 de persoane au fost injectate cu un ser. Probabilitatea ca o persoană să manifeste o reacție nefavorabilă față de acest ser, este de 0.001. Este, deci, de așteptat ca două persoane (2 000 × 0.001) să manifeste o astfel de reacție nefavorabilă. Care este probabilitatea ca patru persoane să manifeste o astfel de reacție?

Să se efectueze același calcul pentru o singură persoană.

```
: RUN
DISTRIBUTIA POISSON
(PT TERMINARE PROGRAM INTRODUCETI 0)
FRECVENTA CALCULATA ? 2
FRECVENTA DORITĂ ? 2
PROBABILITATEA A 4 APARITII =
= 9.02235221 E —02
FRECVENTA CALCULATA ? 2
FRECVENTA DORITA ? 1
PROBABILITATEA 1 A APARIȚII = .270670566473
FRECVENTA CALCULATA ? 0
END PROGRAM AT LINE 180
```

```
10  PRINT „Distributia Poisson"
20  PRINT
30  PRINT „(Pt terminare program introduceti
40  PRINT „Frecventa calculata"
50  INPUT L
59  REM — Sfirsit program ? —
60  IF L = 0 THEN 180
70  PRINT „Frecventa dorita"
80  INPUT X
89  REM — Calculul factorialului —
90  A = 1
100 FOR I = 1 TO X
110     A = A * I
120 NEXT I
129 REM — Calculul probabilitatii —
130 A = LOG(A)
140 A = EXP(—L + X * LOG(L) — A)
150 PRINT „Probabilitatea a"; X;" aparitii ="; A
160 PRINT
169 REM — Restart program —
170 GOTO 40
180 END
```

5.29 — DISTRIBUȚIA NORMALĂ

Acest program calculează probabilitatea și frecvența valorilor date pe o curbă de distribuție normală standard. Pot fi utilizate și variabile nestandard atunci cînd se cunoaște media și abaterea standard.

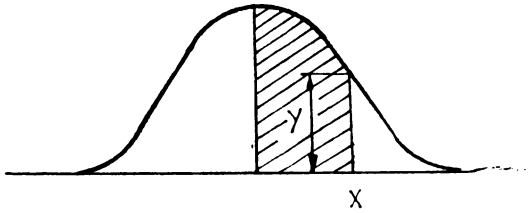


Fig. 5.29

Această curbă reprezintă distribuția normală standard a variabilei X, Y reprezintă frecvența lui X.

Probabilitatea este aproximată de următoarea formulă:

$$\text{Probabilitate} = \frac{1}{2} - r(a_1 t + a_2 t^2 + a_3 t^3) + \varepsilon(X)$$

unde: $a_1 = 0.4361836$; $a_2 = -0.1201676$; $a_3 = 0.9372980$;

$r = (e^{-\frac{X^2}{2}})(2\pi)^{-\frac{1}{2}}$;

$t = (1 + 0.3326X)^{-1}$; $|e(X)| < 10^{-5}$

Exemplu: Greutatea medie a unui grup de persoane este de 75 kg cu o abatere standard de 8 kg. Dacă greutatea este normal distribuită, care este probabilitatea ca o persoană să cîntărească între 75 şi 90 de kg?

Dar între 65 şi 75 kg?

```
: RUN
DISTRIBUTIA NORMALA
(0 = STANDARD, 1 = NESTANDARD)
CE TIP DE VARIABILA ? 1
MEDIA? 75
ABATEREA STANDARD? 8
(PR TERMINARE PROGRAM INTRODUCETI
X = 99999)
X = ? 90
FRECVENTA =
PROBABILITATE =
X = ? 65
FRECVENTA =
PROBABILITATEA =
X = ? 99999
END PROGRAM AT LINE 290
```

```
 10  PRINT „Distributia normală"
 20  PRINT
 30  PRINT „(0 = standard,  1 = nestandard)"
 40  PRINT „Ce tip de variabila"
 50  INPUT S
 60  IF S = 0 THEN 120
 70  PRINT „Media"
 80  INPUT M
 90  PRINT „Abaterea standard"
100  INPUT S
110  GOTO 130
120  S = 1
130  PRINT
```

```
140 PRINT „Pt terminare program introduceti X = 99999"
150 PRINT „X ="
160 INPUT Y
170 IF Y = 99999 THEN 290
179 REM — Ajustarea variabilelor nestandard —
180 Y = (Y - M)/S
189 REM — Calculul frecventei —
190 R = EXP(- Y^2/2)/2.5066232746
200 PRINT „Frecventa =";R
210 Z = Y
219 REM — Aproximatie probabilitate —
220 Y = 1/(1 + .33267*ABS(Y))
230 T = .5 - R*(.4361836*Y - .1201676*Y^2 +
    + .937298*Y^3)
240 IF Z> = 0 THEN 260
250 T = 1 - T
260 PRINT „Probabilitate = "; T
270 PRINT
280 GOTO 150
290 END
```

5.30. — DISTRIBUȚIA X^2

Acest program calculează valoarea părții drepte pentru punctele unei curbe de distribuție X^2. Este necesar să se indice valoarea lui X^2 și numărul gradelor de libertate.

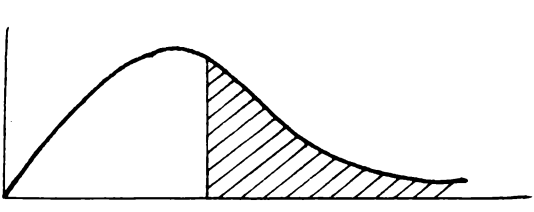


Fig. 5.30

Zona haşurată reprezintă valoarea părții drepte a curbei X^2. Funcția de distribuție X^2 este calculată cu ajutorul formulelor următoare:

— cu ν impar,

$$\text{valoarea părții drepte} = 1 - \frac{(X^2)^{(\nu+1)/2} \cdot e^{-\frac{X^2}{2}}}{1 \cdot 3 \cdot 5 \ldots \nu} \cdot \left(\frac{2}{X^2\pi}\right)^{\frac{1}{2}} \cdot Z$$

— cu ν par,

$$\text{valoarea părții drepte} = 1 - \frac{(X^2)^{\nu/2} \cdot e^{-\frac{X^2}{2}}}{2 \cdot 4 \ldots \nu} \cdot Z$$

unde ν = numărul gradelor de libertate

$$Z = 1 + \sum_{m=1}^{\infty} \frac{(X^2)^m}{(\nu + 2) \cdot (\nu + 4) \ldots (\nu + 2m)}$$

Datorită faptului că pentru calculul variabilei Z suma nu poate fi extinsă la infinit, se va folosi o precizie fixată la valoarea 10^{-7}.

Exemplu: În cadrul unui grup de 168 de persoane, care se plîngeau de insomnie, s-au administrat somnifere la 54 dintre ele, iar celorlalte li s-a aplicat tratament placebo. În urma sondajelor ulterioare, efectuate asupra rezultatelor tratamentului, statistica X^2 a fost calculată ca fiind egală cu 2.571108, cu un grad de libertate. Care este valoarea părții drepte?

```
: RUN
DISTRIBUTIA KHI-PATRAT
(PT TERMINARE PROGRAM INTRODUCETI 0)
NR GRADELOR DE LIBERTATE? 1
CHI-PATRAT? 2.5711108
VALOAREA PARȚII DREPTE = .108831484618
NR GRADELOR DE LIBERTATE? 0
END PROGRAM AT LINE 280
```

```
10  PRINT „Distributia KHI-patrat"
20  PRINT
30  PRINT „(Pt terminare program introduceti 0)"
40  PRINT „Nr gradelor de libertate"
50  INPUT V
60  IF V = 0 THEN 280
70  PRINT „KHI-patrat"
80  INPUT W
90  R = 1
100 FOR I = V TO 2 STEP −2
110 R = R*I
120 NEXT I
130 K = W^(INT((V + 1)/2))*EXP(−W/2)/R
```

```
139 REM — Nr PI este folosit numai cind nr gradelor de libertate
    e impar —
140 IF INT(V/2) = V/2 THEN 170
150 J = SQR(2/W/3.1415926535599)
160 GOTO 180
169 REM — Liniile 170—240 calculeaza valoarea lui L —
170 J = 1
180 L = 1
190 M = 1
200 V = V + 2
210 M = M*W/V
219 REM — Verificarea sfirsitului operatiei de insumare —
220 IF M<.0000001 THEN 250
230 L = L + M
240 GOTO 200
250 PRINT „Valoarea partii drepte ="; 1 — J*K*L
260 PRINT
270 GOTO 40
280 END
```

5.31 — TESTUL X^2

Acest program calculează statistica X^2 și numărul gradelor de libertate asociat unei tabele de contingență. Valoarea așteptată a fiecărui element și contribuția sa în cadrul statisticii X^2, este, de asemenea, afișată.

Instrucția de dimensionare din linia 30 limitează dimensiunea tabloului de contingență. Această dimensiune se poate modifica în felul următor:

```
30 DIM V1(R, C), V2(C), A(R)
```

unde: R = număr de linii în tabela de contingență; C = număr de coloane în tabela de con tingență.

Exemplu: Într-un grup de persoane care se plîngeau de insomnie, unora li s-au administrat somnifere, iar altora tratamente placebo. Mai tîrziu aceste persoane au fost întrebate dacă medicamentele au avut efect favorabil. Rezultatele sînt prezentate în următoarea tabelă de contingență. Care este valoarea statisticii X^2?

	au dormit bine	au dormit rău
somnifere	44	10
placebo	81	35

```
: 30 DIM V1(4), V2(2), A(2)
: RUN
TESTUL KHI-PATRAT
NR DE LINII ÎN TABELA DE CONTINGENTA? 2
NUMĂR DE COLOANE IN TABELA DE CONTIN-
GENTA? 2
TABELA DE CONTINGENTA:
LINIA 1
   ELEMENT 1 ? 44
   ELEMENT 2 ? 10
LINIA 2
   ELEMENT 1 ? 81
   ELEMENT 2 ? 35
VALOARE          VALOARE          CONTRIBUTIE
OBSERVATA        ASTEPTATA        KHI 2
COLOANA 1
44               39.70588235294   .4644008714652
81               85.29411764706   .21 61866125786
COLOANA 2
10               14.29411764706   1.290002420737
35               30.70588235294   .600518368274
KHI-PATRAT = 2.571108273054
NR GRADE DE LIBERTATE = 1
END PROGRAM AT LINE 470
```

```
 10  PRINT „Testul KHI-patrat"
 20  PRINT
 28  REM — Tabela de contingenta este limitata V1(R*C),
     V2(C), A(R)
 29  REM — unde R = nr de linii, C = nr de coloane —
 30  DIM V1(25), V2(5), A(5)
 40  PRINT „Nr de linii"
 49  REM — Liniile 50—150 citesc tabela de contingenta —
 50  INPUT R
 60  PRINT „Nr de coloane"
 70  INPUT C
 80  PRINT „Tabela de contingenta:"
 90  FOR I = 1 TO R
100     PRINT „Linie"; I
110     FOR J = 1 TO C
```

```
120        PRINT „Element“; J
130        INPUT V1((I − 1)*C + J)
140     NEXT J
150  NEXT I
160  PRINT
169  REM − Aduna frecventele marginale pt fiecare linie −
170  L = 0
180  M = 1
190  FOR I = 1 TO R
200     FOR J = 1 TO C
210        A(I) = A(I) + V1(M)
220        M = M + I
230     NEXT J
240     L = L + A(I)
250  NEXT I
260  N = R*C
270  FOR I = 1 TO C
280     FOR J = 1 TO N STEP C
290        V2(I) = V2(I) + V1(J)
300     NEXT J
310  NEXT I
320  Z = 0
269  REM − Aduna frecventele marginale pt fiecare coloana −
330  PRINT „Valoarea observata“, „Valoarea estimata“, „Contri-
     butia KHI 2“
340  FOR I = 1 TO C
350     PRINT „Coloana“; I
360     FOR J = 1 TO R
370        P = A(J)*V2(I)/L
380        X = I + (J − 1)*C
390        V = (V1(X) − P)^2/P
400        Z = Z + V
410        PRINT „ “; V1(X), P, V
420     NEXT J
430  NEXT I
440  PRINT
450  PRINT „KHI-patrat =“; Z
460  PRINT „Grade de libertate =“; (C−1)*(R−1)
470  END
```

5.32 — DISTRIBUȚIA STUDENT

Acest program calculează valorile unor puncte situate pe partea dreaptă a unei curbe de distribuție *t*. Trebuie indicate numărul gradelor de libertate și valoarea lui *t*.

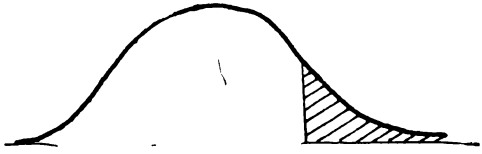


Fig. 5.32

În figura de mai sus, zona haşurată reprezintă valoarea părții drepte corespunzătoare lui *t*. Această valoare este aproximată cu ajutorul formulei următoare:

$$\text{valoarea părții drepte} = \frac{1}{2}(1 + a_1X + a_2X^2 + a_3X^3 + a_4X^4)^{-4} + e(X)$$

unde: $a_1 = 0.196854$
$a_2 = 0.115194$
$a_3 = 0.000344$
$a_4 = 0.019527$
d = nr. gradelor de libertate
$|e(X)| < 2 \cdot 5 \cdot 10^{-4}$

$$X = \left[t^{2/3}\left(1 - \frac{2}{9d}\right) - \frac{7}{9} \right] \left(\frac{2}{9} + t^{4/3} \cdot \frac{2}{9d} \right)^{-1/2}$$

Exemplu: Care este valoarea părții drepte pentru $t = 2.921$ în situația în care avem 16 grade de libertate?

Care este valoarea părții drepte pentru $t = 11.178$ și există 5 grade de libertate?

```
: RUN
DISTRIBUTIA STUDENT
(INTRODUCETI 0 PENTRU TERMINARE PROGRAM)
VALOAREA LUI T? 2.921
GRADE DE LIBERTATE? 16
VALOAREA PARTII DREPTE = 9.70000000 E-03
VALOAREA LUI T? 11.178
GRADE DE LIBERTATE? 5
VALOAREA PARTII DREPTE = 4.00000000 E-04
VALOAREA LUI T? 0
END PROGRAM AT LINE 340
```

```
10  PRINT „Distributia STUDENT"
20  PRINT
30  PRINT „(Introduceti 0 pt terminare program)"
40  PRINT „Valoarea lui T ="
50  INPUT T
60  IF T = 0 THEN 340
70  PRINT „Grade de libertate"
80  INPUT D
90  X = 1
100 Y = 1
110 T = T^2
120 IF T < 1 THEN 170
130 S = Y
140 R = D
150 Z = T
160 GOTO 200
170 S = D
180 R = Y
190 Z = 1/T
200 J = 2/9/S
210 K = 2/9/R
219 REM — Calculeaza cu ajutorul formulei de aproximare —
220 L = ABS((1-K)*Z^(1/3) - 1 + J)/SQR(K*Z^(2/3) + J)
230 IF R < 4 THEN 270
240 X = .5/(1 + L*(.196854 + L*(.115194 + L*(.000344 + L*
    *.019527))))^4
250 X = INT(X*10000 + .5)/10000
260 GOTO 290
270 L = L*(1 + .08*L^4/R^3)
280 GOTO 240
290 IF T > = 1 THEN 310
300 X = 1 - X
310 PRINT „Valoarea partii drepte ="; X
320 PRINT
330 GOTO 40
340 END
```

5.33 — TESTUL STUDENT

Acest program calculează statistica *t* și numărul gradelor de libertate ale unei distribuții Student. Calculele se fundamentează pe una dintre următoarele trei ipoteze prezentate mai jos.

Prima dintre ipoteze presupune că media unei populații este egală cu o valoare dată. Trebuie introduse elementele eșantionului și valoarea mediei.

Celelalte două ipoteze compară două populații. Mediile celor două populații sînt egale, dar abaterea standard poate fi egală sau diferită, de la o populație la cealaltă. Pentru aceste ipoteze este necesar să se introducă numai elementele fiecărui eșantion.

Instrucția de dimensionare din linia 30, limitează dimensiunea eșantionului care poate fi introdus. Această limită poate fi modificată astfel:

```
30 DIM P(N, 2)
```

unde N = dimensiunea maximă a eșantionului.

Exemplu: Pentru un eșantion de copii Q_i este dat de următoarea listă: 101, 99, 120, 79, 111, 98, 106, 112, 87, 97. Să se determine statistica t presupunînd că media este 100.

Pentru un al doilea eșantion rezultatele sînt următoarele: 101, 95, 130, 150, 75, 79, 111, 100, 98, 91. Să se calculeze statistica t în ipoteza în care ambele eșantioane au media și abaterea standard identică.

```
: RUN
TESTUL STUDENT
TEST 1: MEDIA = X
TEST 2: MEDIA EGALA, ABATEREA STANDARD
                                    EGALA
TEST 3: MEDIA EGALA, ABATEREA STANDARD
                                    DIFERITA
TIP TEST? 1
ESANTION 1:
  NUMAR DE ELEMENTE? 10
     ELEMENT  1 ? 101
     ELEMENT  2 ?  99
     ELEMENT  3 ? 120
     ELEMENT  4 ?  79
     ELEMENT  5 ? 111
     ELEMENT  6 ?  98
     ELEMENT  7 ? 106
     ELEMENT  8 ? 112
     ELEMENT  9 ?  87
     ELEMENT 10 ?  97
VALOAREA MEDIEI? 100
VALOARE T = .26151301641
GRADE DE LIBERTATE = 9
END PROGRAM AT LINE 450
```

```
: RUN
TESTUL STUDENT
TEST 1: MEDIA = X
TEST 2: MEDIA EGALA, ABATEREA STANDARD
                                    EGALA
TEST 3: MEDIA EGALA, ABATEREA STANDARD
                                   DIFERITA
TIP TEST ? 2
ESANTION 1:
   NUMAR DE ELEMENTE ? 10
      ELEMENT  1 ? 101
      ELEMENT  2 ?  99
      ELEMENT  3 ? 120
      ELEMENT  4 ?  79
      ELEMENT  5 ? 111
      ELEMENT  6 ?  98
      ELEMENT  7 ? 106
      ELEMENT  8 ? 112
      ELEMENT  9 ?  87
      ELEMENT 10 ?  97
ESANTION
   NUMAR DE ELEMENTE ? 10
      ELEMENT  1 ? 101
      ELEMENT  2 ?  95
      ELEMENT  3 ? 130
      ELEMENT  4 ? 150
      ELEMENT  5 ?  75
      ELEMENT  6 ?  79
      ELEMENT  7 ? 111
      ELEMENT  8 ? 100
      ELEMENT  9 ?  98
      ELEMENT 10 ?  91
VALOARE T = .246515212849
GRADE DE LIBERTATE = 18
END PROGRAM AT LINE 450
```

```
10  PRINT „Testul STUDENT"
20  PRINT
29  REM — Dimensiunea tabloului limitata P(N, 2) N = max—
30  DIM P(10, 2)
40  DIM V(2), R(2), M(2), D(2)
```

```
 50  PRINT „Test 1: media = X"
 60  PRINT „Test 2:  media = media, abaterea standard =
     = abaterea standard"
 70  PRINT „Test 3: media = media, abaterea standard <>
     abaterea standard"
 80  PRINT „Selectati tipul testului"
 90  INPUT T
100  PRINT
109  REM — Se introduc 1 sau 2 esantioane in functie de tipul
     testului —
110  FOR I = 1  TO SGN(T - 1) + 1
120     V(I) = 0
130     D(I) = 0
140     PRINT „Esantionul"; I;" :"
150     PRINT „Nr de elemente"
160     INPUT R(I)
170     FOR J = 1 TO R(I)
180        PRINT „Element"; J
190        INPUT P(J, I)
199        REM — Acumulare esantion —
200        V(I) = V(I) + P(J, I)
210        D(I) = D(I) + P(J, I)^2
220     NEXT J
229     REM — Calculeaza valorile intermediare —
230     M(I) = V(I)/R(I)
240     V(I) = (D(I) - V(I)^2/R(I))/R(I) - 1)
250  NEXT I
260  PRINT
270  IF T = 2 THEN 340
280  IF T = 3 THEN 380
289  REM — Se introduc valorile pt primul test —
290  PRINT „Valoarea mediei"
300  INPUT M
309  REM — Calculeaza T si nr de grade de libertate pt primul
     test —
310  A(M(1) - M)*SQR(R(1)/V(1))
320  B = R(1) - 1
330  GOTO 420
339  REM — Calculează T si nr de grade de libertate pt al doilea
     test —
340  A = (M(1) - M(2))/SQR(1/R(1) + 1/R(2))
350  B = R(1) + R(2) - 2
360  A = A/SQR(((R(1) - 1)*V(1) + (R(2) - 1)*V(2))/B)
```

```
370 GOTO 420
379 REM — Calculează T si nr de grade de libertate pt al treilea
    test —
380 A = (M(1) — M(2))/SQR(V(1)/R(1) + V(2)/R(2))
390 B = (V(1)/R(1) + V(2)/R(2))^2
400 B = B/((V(1)/R(1))^2/(R(1) + 1) + (V(2)/R(2))^2/
    /R(2) + 1)) — 2
410 B = INT(B + .5)
420 PRINT
430 PRINT „Valoarea lui T ="; ABS(A)
440 PRINT „Grade de libertate ="; B
450 END
```

5.34 — DISTRIBUȚIA F

Acest program calculează valorile centilei pentru anumite puncte situate pe o curbă de distribuție F. Trebuie indicată valoarea lui F, numărul gradelor de libertate de la numărător și numărul gradelor de libertate de la numitor.

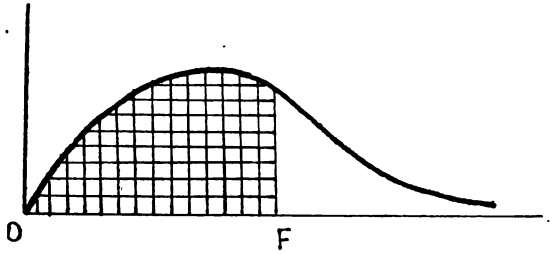


Fig. 5.34

Zona haşurată reprezintă centila.

Funcția de distribuție F este aproximată cu ajutorul formulei următoare:

$$\text{centila} = 1 - \frac{1}{2}(1 + a_1 Y + a_2 Y^2 + a_3 Y^3 + a_4 Y^4)^{-4} + e(Y)$$

$$Y = \left[F^{1/3}\left(1 - \frac{2}{9d_2}\right) - \left(1 - \frac{9}{9d_1}\right)\right]\left(\frac{2}{9d_1} + F^{2/3} \cdot \frac{2}{9d_2}\right)^{-1/2}$$

unde: $a_1 = 0.196854$; $a_2 = 0.115194$; $a_3 = 0.000344$; $a_4 = 0.019527$

$|e(Y)| < 2.\ 10^{-4}$

d_1 = nr. grade de libertate al numărătorului; d_2 = nr. grade de libertate al numitorului.

Exemplu: Care este centila unei curbe de distribuție F dacă $F = 0.474$ şi numărul gradelor de libertate este 1 şi 18?

Care este centila pentru $F = 23.7$, iar numărul gradelor de libertate este egal cu 3 și 6?

```
: RUN
DISTRIBUTIA F
(INTRODUCETI 0 PT TERMINAREA PROGRAMULUI
VALOAREA LUI F? .474
GRADE DE LIBERTATE LA NUMARATOR? 1
GRADE DE LIBERTATE LA NUMITOR? 18
CENTILA = .4937
VALOAREA LUI F? 23.7
GRADE DE LIBERTATE LA NUMARATOR? 3
GRADE DE LIBERTATE LA NUMITOR? 6
CENTILA = .9984
VALOAREA LUI F? 0
END PROGRAM AT LINE 340
```

Opțiune: Este posibil să se dorească calcularea valorii părții drepte a curbei ((aria porțiunii nehașurate din figura de mai sus). Singura modificare necesară este următoarea:

```
310 PRINT „VALOAREA PARTII DREPTE =“; X
```

```
 10  PRINT „Distributia F“
 20  PRINT
 30  PRINT „(Introduceti 0 pt terminarea programului)“
 40  PRINT „Valoarea lui F =“
 50  INPUT F
 60  IF F = 0 THEN 340
 70  PRINT „Grade de libertate de la numarator“
 80  INPUT D1
 90  PRINT „Grade de libertate de la numitor“
100  INPUT D2
110  X = 1
120  IF F<1 THEN 170
130  S = D1
140  T = D2
150  Z = F
160  GOTO 200
170  S = D2
180  T = D1
190  Z = 1/F
200  J = 2/9/S
210  K = 2/9/T
219  REM — Calculeaza cu ajutorul formulei de aproximare —
220  Y = ABS((1 — K)*Z (1/3) — 1 + J)/SQR(K*Z (2/3) + J)
230  IF T<4 THEN 270
```

```
240  X = .5/(1 + Y*(.196854 + Y*(.115194 + Y*(.000344 +
     +Y*.019527)))) 4
250  X = INT(X*10000 + .5)/10000
260  GOTO 290
270  Y = Y*(1 + .08*Y 4/T 3)
280  GOTO 240
290  IF F > = 1 THEN 310
300  X = 1 − X
310  PRINT „Centila =“; 1 − X
320  PRINT
330  GOTO 40
340  END
```

5.35 — COEFICIENT DE CORELAȚIE LINIARĂ

Acest program calculează coeficientul de corelație liniară între două variabile. Presupunem că relația existentă între cele două variabile este liniară. Trebuie să indicăm coordonatele unui grup de puncte ce constituie linia de regresie.

Exemplu: Tabela de mai jos reprezintă, în centimetri, talia a 12 bărbați și a fiilor lor. Care este coeficientul de corelație între talia tatălui și cea a fiului?

tată	65	63	67	64	68	62	70	66	68	67	69	71
fiu	68	66	68	65	69	66	68	65	71	67	68	70

```
: RUN
COEFICIENT DE CORELATIE LINIARA
NUMAR DE PUNCTE ? 12
X,Y PENTRU PUNCTUL  1 ? 65,68
X,Y PENTRU PUNCTUL  2 ? 63,66
X,Y PENTRU PUNCTUL  3 ? 67,68
X,Y PENTRU PUNCTUL  4 ? 64,65
X,Y PENTRU PUNCTUL  5 ? 68,69
X,Y PENTRU PUNCTUL  6 ? 62,66
X,Y PENTRU PUNCTUL  7 ? 70,68
X,Y PENTRU PUNCTUL  8 ? 66,65
X,Y PENTRU PUNCTUL  9 ? 68,71
X,Y PENTRU PUNCTUL 10 ? 67,67
X,Y PENTRU PUNCTUL 11 ? 69,68
X,Y PENTRU PUNCTUL 12 ? 71,70
COEFICIENT DE CORELATIE LINIARA =
= .7026516450773
END PROGRAM AT LINE 220
```

```
10  PRINT „Coeficient de corelatie liniara"
20  PRINT
30  PRINT „Numarul punctelor"
40  INPUT N
50  J = 0
60  K = 0
70  L = 0
80  M = 0
90  R = 0
99  REM — Se introduc coordonatele punctelor —
100 FOR I = 1 TO N
110     PRINT „X, Y pentru punctul "; I
120     INPUT X, Y
129     REM — Se acumuleaza valorile intermediare —
130     J = J + X
140     K = K + Y
150     L = L + X^2
160     M = M + Y^2
170     R = R + X*Y
180 NEXT I
189 REM — Calculeaza si afiseaza coeficientul —
190 R2 = (N*R-J*K)/SQR((N*L-J 2)*(N*M-K^2))
200 PRINT
210 PRINT „Coeficient de corelatie liniara ="; R2
220 END
```

5.36 — REGRESIE LINIARĂ

Acest program realizează ajustarea unei drepte pe un ansamblu de puncte date, utilizînd metoda celor mai mici pătrate. Sînt afişate ecuaţia dreptei, coeficientul de determinare, coeficientul de corelaţie şi eroarea standard de estimare. După ce ajustarea a fost realizată, pot fi determinate valori ale funcţiei Y pornind de la valori date ale lui X.

Exemplu: Tabela de mai jos conţine dimensiunea, în centimetri, a taliei a 11 subiecţi, şi greutatea lor în kilograme. Să se ajusteze o dreaptă pe aceste puncte. Care va fi greutatea medie a doi subiecţi cu talia de respectiv 70 şi 72 de centimetri?

talia	71	73	64	65	61	70	65	72	63	67	64
greutatea	160	183	154	168	159	180	145	210	132	168	141

```
: RUN
REGRESIE LINIARA
NUMARUL PUNCTELOR CUNOSCUTE ? 11
X,Y ALE PUNCTULUI  1 ? 71,160
X,Y ALE PUNCTULUI  2 ? 73,183
X,Y ALE PUNCTULUI  3 ? 64,154
X,Y ALE PUNCTULUI  4 ? 65,168
X,Y ALE PUNCTULUI  5 ? 61,159
X,Y ALE PUNCTULUI  6 ? 70,180
X,Y ALE PUNCTULUI  7 ? 65,145
X,Y ALE PUNCTULUI  8 ? 72,210
X,Y ALE PUNCTULUI  9 ? 63,132
X,Y ALE PUNCTULUI 10 ? 67,168
X,Y ALE PUNCTULUI 11 ? 64,141
F(X) = -106.7916666666 + (4.047222222222*X)
COEFICIENT DE DETERMINARE (R 2) = .5562601669757
COEFICIENT DE CORELATIE = .74582851043
EROAREA STANDARD A ESTIMATIEI = 15.41348816
INTERPOLARE: (INTRODUCETI X=0 PT
TERMINARE PROGRAM)
X = ? 70
Y = 176.5138888889
X = ? 72
Y = 184.6083333334
X = ? 0
END PROGRAM AT LINE 390
```

```
 10  PRINT „Regresie liniara“
 20  PRINT
 30  PRINT „Numarul punctelor cunoscute“
 40  INPUT N
 50  J = 0
 60  K = 0
 70  L = 0
 80  M = 0
 90  R2 = 0
 99  REM — Se introduc coordonatele punctelor —
100  FOR I = 1 TO N
110     PRINT „X, Y ale punctului“; I
120     INPUT X, Y
129     REM — Acumulare puncte intermediare —
130     J = J + X
```

```
140     K = K + Y
150     L = L + X^2
160     M = M + Y^2
170     R2 = R2 + X*Y
180 NEXT I
189 REM — Calculeaza coeficientul curbei —
190 B = (N*R2 — K*J)/(N*L — J^2)
200 A = (K — B*J)/N
210 PRINT
220 PRINT „F(X) ="; A;" + („; B;"*X)"
229 REM — Calculează regresia —
230 J = B*(R2 — J*K/N)
240 M = M — K^2/N
250 K = M — J
260 PRINT
270 R2 = J/M
280 PRINT „Coeficient de determinare ="; R2
290 PRINT „Coeficient de corelatie ="; SQR(R2)
300 PRINT „Eroarea standard a estimatiei ="; SQR(K/(N—2))
310 PRINT
319 REM — Estimare coordonata Y pt punctul cu coordonata X —
320 PRINT „Interpolare: (introduceti X = 0 pt terminare
    program)"
330 PRINT „X="
340 INPUT X
349 REM — Restart sau sfirsit program —
350 IF X = 0 THEN 390
360 PRINT „Y ="; A + B*X
370 PRINT
380 GOTO 330
390 END
```

5.37 — REGRESIE LINIARĂ MULTIPLĂ

Acest program determină coeficienții unei ecuații liniare cu mai multe variabile, utilizînd metoda celor mai mici pătrate. Ecuația are forma următoare:

$$Y = C + a_1X_1 + a_2X_2 + \ldots a_nX_n$$

unde: Y — variabila dependentă; C — constantă; a_1, a_2, ..., a_n — coeficienții variabilelor independente

$$X_1,\ X_2,\ \ldots,\ X_n.$$

Trebuie indicate coordonatele X, Y ale punctelor cunoscute. După introducerea datelor și definirea ecuației se pot determina și alte valori ale variabilei dependente Y fixînd valorile variabilelor independente.

Instrucția din linia 30 limitează numărul punctelor cunoscute la intrare. Se poate modifica această limită procedînd astfel:

30 DIM X(N + 1), S(N + 1), T(N + 1), A(N + 1, N + 2)

unde: N — numărul punctelor cunoscute.

Exemplu: Tabela de mai jos indică vîrsta, talia și greutatea a opt subiecți. Luînd greutatea ca variabilă dependentă, să se ajusteze o dreaptă pe ansamblul de puncte. Să se evalueze greutatea unui copil de șapte ani a cărui talie măsoară 51 cm.

vîrsta	8	9	6	10	8	9	9	7
talia	48	49	44	59	55	61	55	50
greutatea	59	55	50	80	61	75	67	58

```
:RUN
REGRESIE LINIARA MULTIPLA
NUMARUL PUNCTELOR CUNOSCUTE? 8
NUMARUL VARIABILELOR INDEPENDENTE? 2
PUNCT 1
   VARIABILA 1? 8
   VARIABILA 2? 48
   VARIABILA DEPENDENTA? 59
PUNCT 2
   VARIABILA 1? 9
   VARIABILA 2? 49
   VARIABILA DEPENDENTA? 55
PUNCT 3
   VARIABILA 1? 6
   VARIABILA 2? 44
VARIABILA DEPENDENTA? 50
PUNCT 4
   VARIABILA 1? 10
   VARIABILA 2? 59
   VARIABILA DEPENDENTA? 80
```

```
PUNCT 5
   VARIABILA 1? 8
   VARIABILA 2? 55
   VARIABILA DEPENDENTA? 61
PUNCT 6
   VARIABILA 1? 9
   VARIABILA 2? 51
   VARIABILA DEPENDENTA? 75
PUNCT 7
   VARIABILA 1? 9
   VARIABILA 2? 55
   VARIABILA DEPENDENTA? 67
PUNCT 8
   VARIABILA 1? 7
   VARIABILA 2? 50
   VARIABILA DEPENDENTA? 58
COEFICIENTII ECUATIEI:
   CONSTANTA: -15.70212765959
VARIABILA 1: 3.680851063828
VARIABILA 2: .9432624113481
COEFICIENT DE DETERMINARE (R↑ 2)=.71569735 88726
COEFICIENT DE CORELATIE MULTIPLA=.84598898271
EROAREA STANDARD A ESTIMATIEI= 6.4288798755
INTERPOLARE: (INTRODUCETI 0 PENTRU TERMI-
   NARE PROGRAM)
VARIABILA 1? 7
VARIABILA 2? 51
VARIABILA DEPENDENTA = 58.17021276596
VARIABILA 1? 0
END PROGRAM AT LINE 810
```

```
 10  PRINT „Regresie liniara multipla"
 20  PRINT
 29  REM — Limite X(N+1), S(N+1), T(N+1), A (N+1, N+2)—
 30  DIM X(9), S(9), T(9), A(10)
 40  PRINT „Numarul punctelor cunoscute"
 50  INPUT N
 60  PRINT „Numarul variabilelor independente"
 70  INPUT V
 80  X(1)=1
 90  FOR I=1 TO N
100  PRINT „Punct"; I
```

```
110 FOR J=1 TO V
119       REM — Se introduc variabilele independente —
120       PRINT „Variabila"; J
130       INPUT X(J+1)
140       NEXT J
149       REM — Se introduc variabilele dependente —
150       PRINT „Variabile dependente"
160       INPUT X(V+2)
170       FOR K=1 TO V+1
180           FOR L=1 TO V+2
190               A(K, L)=A(K, L)+ X(K)*X(L)
200               S(K)=A(K,V+2)
210           NEXT L
220       NEXT K
230       S(V+2)=S(V+2)+X(V+2)^2
240 NEXT I
249 REM — Liniile 250—500 rezolva sistemul de ecuaţii liniare
    din matricea A —
250 FOR I=2 TO V+1
260       T(I)=A(1, I)
270 NEXT I
280 FOR I=1 TO V+1
290       J=1
300       IF A(J, I)<>0 THEN 340
305       J=J+1
310       IF J<=V+1 THEN 300
320       PRINT „Nu exista solutie unica"
330       GOTO 810
340       FOR K=1 TO V+2
350           B=A(I, K)
360           A(I, K)=A(J, K)
370           A(J, K)=B
380       NEXT K
390       Z=1/A(I, I)
400       FOR K=1 TO V+2
410           A(I, K)=Z*A(I, K)
410               A(I, K)=Z*A(I, K)
420       NEXT K
430       FOR J=1 TO V+1
440           IF J=1 THEN 490
450           Z=—A(J, I)
460           FOR K=1 TO V+2
470               A(J, K)=A(J, K)+Z*A(I, K)
```

```
480          NEXT K
490       NEXT J
500    NEXT I
520  PRINT „Coeficienţii ecuatiei:"
525  PRINT „Constanta:"; A(1, V+2)
530  FOR I=2 TO V+1
540       PRINT „Variabila („;I-1;"):"; A(I, V+2)
550  NEXT I
560  P=0
570  FOR I=2 TO V+1
580       P=P+A(I, V+2)*(S(I)-T(I)*S(1)/N)
590  NEXT I
600  R=S(V+2)-S(1)^2/N
610  Z=R-P
620  L=N-V-1
640  PRINT
650  I=P/R
660  PRINT „Coeficient de determinare ="; I
670  PRINT „Coeficient de corelatie multipla ="; SQR(I)
680  PRINT „Eroarea standard a estimatiei ="; SQR(ABS(Z/L)
690  PRINT
700  PRINT „Interpolare: (introduceti 0 pt terminare program)
710  P=A(1, V+2)
720  FOR J=1 TO V
730       PRINT „Variabila"; J
740       INPUT X
749       REM - Test de sfirsit program -
750       IF X=0 THEN 810
760       P=P+A(J+1, V+2)*X
770  NEXT J
780  PRINT „Variabila dependenta ="; P
790  PRINT
800  GOTO 710
810  END
```

5.38 — REGRESIE DE ORDINUL N

Acest program determină coeficienţii unui polinom de ordinul *n* utilizînd metoda celor mai mici pătrate. Polinomul se prezintă sub forma următoare:

$$Y = C + a_1X + a_2X^2 + \ldots + a_nX^n$$

unde: Y — variabila dependentă; C — constantă; $a_1, a_2, \ldots, a_n$ — coeficienţii variabilelor independente $X_1, X_2, \ldots, X_n$.

Programul afișează valorile coeficienților ecuațiilor, coeficientul de determinare, coeficientul de corelație și abaterea standard a estimației.

Trebuie indicate coordonatele X, Y ale punctelor cunoscute. După determinarea coeficienților polinomului se pot afla valorile Y pentru diferite valori date ale variabilelor independente X. Instrucția din linia 30 limitează gradul polinomului. Această limită poate fi modificată după cum urmează:

30 DIM A(N), R(D+1, D+2), T(D+2)

unde: D — gradul maxim al polinomului; N — 2*D+1.

Exemplu: Tabela de mai jos indică distanța de oprire (distanța corespunzătoare timpului de reacție plus distanța de frînare) a unei mașini la diferite viteze.

viteza (km/h)	32	50	60	80	97	113
distanța de oprire (m)	54	90	138	206	292	396

Să se evalueze distanța de oprire a unei mașini cu o viteză de 88,5 km/h.

```
:30 DIM A(5), R(3, 4), T(4)
:RUN
REGRESIE DE ORDINUL N
GRADUL ECUATIEI? 2
NUMARUL PUNCTELOR CUNOSCUTE? 6
X, Y PENTRU PUNCTUL 1? 32,54
X, Y PENTRU PUNCTUL 2? 50,90
X. Y PENTRU PUNCTUL 3? 64,138
X, Y PENTRU PUNCTUL 4? 80,206
X, Y PENTRU PUNCTUL 5? 97,292
X, Y PENTRU PUNCTUL 6? 113,396
          CONSTANTA =
COEFICIENT GRADUL 1 =
COEFICIENT GRADUL 2 =
COEFICIENT DE DETERMINARE (R↑2) =
COEFICIENT DE CORELATIE =
ABATEREA STANDARD A ESTIMATIEI =
INTERPOLARE: (INTRODUCETI 0 PT TERMINARE
PROGRAM)
X = 55
Y =
X = 0
```

```
10  PRINT „Regresie de ordinul n"
20  PRINT
28  REM — Gradul ecuației este limitat prin A(2D+1), R(D+1,
    D+2), T(D+2)
29  REM unde D este gradul ecuației —
30  DIM A(13), R(7, 8), T(8)
40  PRINT „Gradul ecuației"
50  INPUT D
60  PRINT „Numărul punctelor cunoscute"
70  INPUT N
80  A(1)=N
89  REM — Introduceti coordonatele punctelor —
90  FOR I=1 TO N
100       PRINT „X, Y pentru punctul"; I
110       INPUT X, Y
119       REM — Liniile 120—200 populează matricea cu siste-
              mul de ecuații —
120       FOR J=2 TO 2*D+1
130            A(J)=A(J)+X^(J-1)
140       NEXT J
150       FOR K=1 TO D+1
160            R(K, D+2)=T(K)+Y*X^(K-1)
170            T(K)=T(K)+Y*X^(K-1)
180       NEXT K
190       T(D+2)=T(D+2)+Y^2
200 NEXT I
209 REM — Liniile 210—490 rezolva sistemul de ecuatii —
210 FOR J = 1 TO D+1
220       FOR K=1 TO D+1
230            R(J, K)=A(J+K-1)
240       NEXT K
250 NEXT J
260 FOR J=1 TO D+1
270       FOR K=J TO D+1
280            IF R(K, J) <> 0 THEN 320
290       NEXT K
300       PRINT „Nu exista soluție unica"
310       GOTO 790
320       FOR I=1 TO D+2
```

```
330          S=R(J, I)
340          R(J, I)=R(K, I)
350          R(K, I)=S
360       NEXT I
370       Z=1/R(J, J)
380       FOR I=1 TO D+2
390             R(J, I)=Z*R(J, I)
400       NEXT I
410       FOR K=1 TO D+1
420          IF K=J THEN 470
430          Z=-R(K, J)
440          FOR I=1 TO D+2
450             R(K, I)=R(K, I)+Z*R(J, I)
460             NEXT I
470       NEXT K
480    NEXT J
490 PRINT
495 PRINT „          Constanta="; R(1, D+2)
499 REM — Afişeaza coeficientii ecuatiei —
500 FOR J=1 TO D
510 PRINT J; „Coeficient gradul =";R(J+1, D+2)
520 NEXT J
530 PRINT
539 REM — Calculeaza regresia —
540 P=0
550 FOR J=2 TO D+1
560       P=P+R(J, D+2)*(T(J)-A(J)*T(1)/N)
570 NEXT J
580 Q=T(D+2)-T(1)^2/N
590 Z=Q-P
600 I=N-D-1
620 PRINT
630 J=P/Q
640 PRINT „Coeficient de determinare="; J
650 PRINT „Coeficient de corelatie ="; SQR(J)
660 PRINT „Abaterea standard a estimatiei ="; SQR (Z/I)
```

```
670 PRINT
679 REM — Calculul coordonatei Y —
680 PRINT „Interpolare: (introduceti 0 pt terminare program)"
690 P=R(1, D+2)
700 PRINT „X="
710 INPUT X
720 IF X=0 THEN 790
730 FOR J=1 TO D
740       P=P+R(J+1, D+2)*X^J
750 NEXT J
760 PRINT „Y="; P
770 PRINT
780 GOTO 690
790 END
```

5.39 — REGRESIE GEOMETRICĂ

Acest program ajustează o curbă geometrică la un ansamblu de puncte utilizînd metoda celor mai mici pătrate. Programul afișează funcția curbei, coeficientul de determinare, coeficientul de corelație și abaterea standard a estimației.

Trebuie indicate coordonatele X, Y ale punctelor cunoscute. După ajustarea curbei se pot determina valori Y pentru anumite valori X fixate.

Tabela de mai jos indică presiunea unui gaz măsurat în diferite volume, în cursul unui experiment. Relația existentă între presiunea și volumul unui gaz este dată de formula

$$PV^K = C$$

unde: P — presiunea; V — volumul; C, K = constante.
Această relație se poate scrie sub următoarea formă geometric normalizată:

$$P = CV^{-K}.$$

Să se ajusteze o curbă pe ansamblul acestor date și să se evalueze presiunea unui volum de 90 cm³ din acest gaz.

56.1	60.7	73.2	88.3	120.1	187.5
57.0	51.0	32.2	30.2	19.6	10.5

```
:RUN
REGRESIE GEOMETRICA
NUMARUL PUNCTELOR CUNOSCUTE? 6
X, Y PT PUNCTUL 1? 56.1,57.0
X, Y PT PUNCTUL 2? 60.7,51.0
X, Y PT PUNCTUL 3? 73.2,39.2
X, Y PT PUNCTUL 4? 88.3,30.2
X, Y PT PUNCTUL 5? 120.1,19.6
X, Y PT PUNCTUL 6? 187.5,10.5
F(X)=15103.68991715*X↑-1.401550582441
COEFICIENT DE DETERMINARE (R↑2)=.9999988312731
COEFICIENT DE CORELATIE = .99999941564
ABATEREA STANDARD A ESTIMATIEI=7.73614568
E-04
INTERPOLARE: (INTRODUCETI 0 PT TERMINARE
PROGRAM)
X=? 90
Y = 29.37349825098
X=? 6
END PROGRAM AT LINE 410
```

```
10  PRINT „Regresie geometrica“
20  PRINT
30  PRINT „Numarul punctelor cunoscute“
40  INPUT N
50  J = 0
60  K = 0
70  L = 0
80  M = 0
90  R2 = 0
99  REM — Se introduc coordonatele punctelor —
100 FOR I=1 TO N
110       PRINT „X, Y pt punctul“; I
120       INPUT X, Y
129       REM — Se acumuleaza valorile intermediare —
130       Y=LOG(Y)
140       X=LOG(X)
150       J=J+X
160       K=K+Y
```

```
170        L=L+X^2
180        M=M+Y 2
190        R2=R2+X*Y
200  NEXT I
209  REM — Calculeaza si afiseaza coeficientii ecuatiei —
210  B=(N*R2-K*J)/(N*L-J^2)
220  A=(K-B*J)/N
230  PRINT
240  PRINT „F(X) = ";EXP(A);"*X^";B
249  REM — Calculeaza regresia —
250  J=B*(R2-J*K/N)
260  M=M-K^2/N
270  K=M-J
280  PRINT
290  R2=J/M
300  PRINT „Coeficient de determinare =";R2
310  PRINT „Coeficient de corelatie =";SQR(R2)
320  PRINT „Abaterea standard de estimatie =";SQR(K/(N-2))
330  PRINT
340  PRINT „Interpolare: (introduceti 0 pt terminare program)"
350  PRINT „X="
360  INPUT X
370  IF X=0 THEN 410
380  PRINT „Y = ";EXP(A)*X^B
390  PRINT
400  GOTO 350
410  END
```

5.40 — REGRESIE EXPONENȚIALĂ

Acest program determină coeficienții funcției unei curbe exponențiale. Funcția se prezintă sub forma:

$$f(X) = a\, e^{bX}$$

a și b fiind coeficienții calculați.

Programul afișează coeficienții funcției, coeficientul de determinare, coeficientul de corelație și abaterea standard a estimației. Trebuie indicate coordonatele X, Y ale punctelor cunoscute. După ajustarea curbei se pot determina valori Y pentru valori fixate ale lui X.

Exemplu: Tabela de mai jos indică numărul bacteriilor prezente într-o cultură, la momente diferite. Să se ajusteze o curbă exponențială pe acest ansamblu de date și să se evalueze numărul bacteriilor după 7 ore.

nr. de ore	0	1	2	3	4	5	6
nr. de bacterii	25	38	58	89	135	206	315

```
:FUN
REGRESIE EXPONENTIALA
NUMĂRUL PUNCTELOR CUNOSCUTE? 7
X, Y PENTRU PUNCTUL 1? 0,25
X, Y PENTRU PUNCTUL 2? 1,38
X, Y PENTRU PUNCTUL 3? 2,58
X, Y PENTRU PUNCTUL 4? 3,89
X, Y PENTRU PUNCTUL 5? 4,135
X, Y PENTRU PUNCTUL 6? 5,206
X, Y PENTRU PUNCTUL 7? 6,315
A = 24.96166337346
B = .4223750795699
COEFICIENT DE DETERMINARE (R↑2)=.9999935513734
COEFICIENT DE CORELATIE=.99999677868
EROAREA STANDARD A ESTIMATIEI=2.53820862 E-03
INTERPOLARE: (INTRODUCETI 0 PT TERMINARE
PROGRAM)
X=? 7
Y=480.0867130787
X=? 0
END PROGRAM AT LINE 410
```

```
10  PRINT „Regresie exponentiala"
20  PRINT
30  PRINT „Numarul punctelor cunoscute"
40  INPUT N
50  J=0
60  K=0
70  L=0
80  M=0
90  R2=0
99  REM — Se introduc coordonatele punctelor —
```

```
100  FOR I=1 TO N
110       PRINT „X, Y pentru punctul"; T
120       INPUT X, Y
129       REM — Se acumuleaza valorile intermediare —
130       Y = LOG(Y)
140       J=J+X
150       K=K+Y
160       L=L+X^2
170       M=M+Y^2
180       R2=R2+X*Y
190  NEXT I
199  REM — Se calculeaza si afiseaza coeficientii ecuaţiei —
200  B=(N*R2-K*J)/(N*L-J^2)
210  A=(K-B*J)/N
220  PRINT
230  PRINT „A="; EXP(A)
240  PRINT „B="; B
250  J=B*(R2-J*K/N)
260  M=M-K^2/N
270  K=M-J
280  PRINT
290  R2=J/M
300  PRINT „Coeficient de determinare =";R2
310  PRINT „Coeficient de corelatie =";SQR(R2)
320  PRINT „Eroarea standard de estimatie ="; SQR(K/(N-2))
330  PRINT
340  PRINT „Interpolare: (introduceti 0 pt terminare program)"
350  PRINT „X="
360  INPUT X
370  IF X=0 THEN 410
380  PRINT „Y = ";EXP(A)*EXP(B*X)
390  PRINT
400  GOTO 350
410  END
```

5.41 — FIABILITATEA UNUI SISTEM

Acest program calculează fiabilitatea unui sistem în funcţionare supus uzurii şi penelor intempestive. Trebuiesc indicate durata de funcţionare a sistemului, durata de utilizare (uzura) şi rata medie a penelor pentru fiecare componentă a sistemului.

Exemplu: Să se calculeze fiabilitatea unui ordinator care funcţionează timp de 1 000 de ore cu componente din tabelul de mai jos:

	Timp uzură (ore)	Rata penelor
Unitate centrală	15 000	0.00020
Terminal	3 000	0.00010
Disc	3 000	0.00015
Imprimantă	1 500	0.00015

```
:RUN
FIABILITATEA SISTEMELOR
(PT TERMINAREA PROGRAMULUI INTRODUCETI 0)
TIMP DE FUNCTIONARE IN ORE? 1000
NUMAR DE COMPONENTE? 4
COMPONENTA 1
   TIMP MEDIU DE UZURA? 15000
   RATA MEDIE A PENELOR? .00020
COMPONENT  2
   TIMP MEDIU DE UZURA? 3000
   RATA MEDIE A PENELOR? .00010
COMPONENT 3
   TIMP MEDIU DE UZURA? 3000
   RATA MEDIE A PENELOR .00015
COMPONENT 4
   TIMP MEDIU DE UZURA? 1500
   RATA MEDIE A PENELOR? .00015
FIABILITATE = .1353352332367
TIMP DE FUNCTIONARE IN ORE? 0
END PROGRAM AT LINE 230
```

```
10  PRINT „Fiabilitatea unui sistem“
20  PRINT
30  PRINT „(Pt terminarea programului introduceti 0)“
40  PRINT „Timp de functionare in ore“
50  INPUT T
59  REM — Test de sfirsit program —
60  IF T=0 THEN 230
70  PRINT „Numar de componente“
```

```
 80  INPUT N
 90  Z=0
 99  REM — Se introduc date pt fiecare component —
100  FOR I=1 TO N
110  PRINT „Component";I
120  PRINT „Timpul mediu de uzura"
130  INPUT W
140  PRINT „Rata medie a penelor"
150   INPUT F
159  REM — Calculeaza fiabilitatea in functie de fiecare compo-
          nent —
160       Z=Z+1/W+F
170  NEXT I
180  PRINT
189  REM — Calculeaza si afiseaza fiabilitatea sistemului —
190  Z=EXP(-Z*T)
200  PRINT „Fiabilitatea sistemului =";Z
210  PRINT
219  REM — RESTART PROGRAM —
220  GOTO 40
230  END
```

5.42 — ZILELE SĂPTĂMÎNII

Acest program este util în situaţia în care dorim să aflăm în ce zi a săptămînii va cădea o anumită dată calendaristică. Data trebuie indicată sub forma lună, zi, an.

Exemplu: Elena s-a născut pe data de 24 decembrie 1964. În ce zi a săptămînii s-a născut Elena?
Unchiul Nicu a avut o întîlnire pe data de 30 septembrie 1977. În ce zi a săptămînii a căzut această dată?

```
:RUN
ZILELE SAPTAMINII
(INTRODUCETI, 0, 0, 0 PENTRU TERMINARE PRO-
GRAM
LUNA, ZIUA, ANUL? 12, 24, 1964
LUNA, ZIUA, ANUL? 9, 30, 1977
LUNA, ZIUA, ANUL? 0, 0, 0
END PROGRAM AT LINE 360
```

```
 10  PRINT „Zilele saptaminii"
 20  PRINT
 29  REM — Cere introducerea datelor —
 30  PRINT „(Introduceti 0, 0, 0 pt terminare program)"
 40  PRINT „Luna, zi, an"
 50  INPUT M, D, Y
 59  REM — Test de sfirsit program —
 60  IF M<>0 THEN 100
 70  IF D<>0 THEN 100
 80  IF Y<>0 THEN 100
 90  GOTO 360
100  IF M>2 THEN 130
110  M=M+12
120  Y=Y-1
129  REM — Calculeaza nr zilelor —
130  N=D+2*M+INT(.6*(M+1))+Y+INT(Y/4)-INT(Y/100)
     +INT(Y/400)+2
140  N=INT((N/7-INT(N/7))*7+.5)
150  IF N>0 THEN 180
160  PRINT „Simbata"
170  GOTO 340
180  IF N>1 THEN 210
190  PRINT „Duminica"
200  GOTO 40
210  IF N>2 THEN 240
220  PRINT „Luni"
230  GOTO 340
240  IF N>3 THEN 270
250  PRINT „Marti"
260  GOTO 340
270  IF N>4 THEN 300
280  PRINT „Miercuri"
290  GOTO 340
300  IF N>5 THEN 330
310  PRINT „Joi"
320  GOTO 340
330  PRINT „Vineri"
340  PRINT
349  REM — Restart program —
350  GOTO 40
360  END
```

5.43 — NUMĂRUL DE ZILE DINTRE DOUĂ DATE

Acest program calculează numărul de zile dintre două date calendaristice. Numărul de zile este determinat ținînd cont de anii bisecți. De asemenea, programul presupune că de azi pînă mîine trece o zi, adică între 1 și 3 ale lunii trec două zile.

Pentru a fi siguri de utilizarea corectă a acestui program este necesar să ne luăm anumite precauții. În primul rînd trebuie să introducem mai întîi data cea mai veche. În al doilea rînd, datele trebuie introduse sub formă numerică (3 nu martie) și în ordinea: lună, zi, an. Cifrele vor fi separate de virgule. În al treilea rînd, anul nu va fi trunchiat (se va introduce 1976 nu 76). În sfîrșit, numărul lunii nu poate fi mai mare decît 12, iar numărul zilei nu poate depăși numărul zilelor din luna respectivă. În caz contrar se va afișa mesajul DATA IREALĂ.

Exemplu: Data de naștere a Ioanei este 30 iunie 1956. Cîte zile va avea ea la cea de-a treizecea aniversare.

```
:RUN
NR DE ZILE INTRE DOUA DATE
PRIMA DATA? 30, 6, 1956
A DOUA DATA? 28, 9, 1986
NR DE ZILE =
CONTINUATI  (1 = DA, 0 = NU)? 0
```

```
 10  PRINT „Numarul zilelor dintre doua date"
 20  PRINT
 29  REM — Se introduc datele —
 30  PRINT „Prima data"
 40  INPUT M1, D1, Y1
 50  PRINT „A doua data"
 60  INPUT M2, D2, Y2
 69  REM — Initializeaza variabilele utilizate in subrutina —
 70  M=M1
 80  D=D1
 90  Y=Y1
100  GOSUB 230
109  REM — Salveaza in N nr zilelor calculat —
110  N=A
120  M=M2
130  D=D2
140  Y=Y2
```

```
150  GOSUB 230
159  REM — Calculeaza si afiseaza diferenta —
160  N=A-N
170  PRINT „Nr de zile =";N
180  PRINT
189  REM — Restart sau sfirsit program —
190  PRINT „Continuati (1=da, 0=nu)?)"
200  INPUT X
210  IF X=1 THEN 20
219  REM — Sfirsit program —
220  GOTO 460
227  REM — Subrutina pt calculul nr de zile de la 0, 0, 0 la M, D, V
228  REM — efectueaza test de coerenta asupra datei —
229  REM — Testul de coerenta se face dupa zi si luna —
230  ON M GOTO 160, 280, 160, 340, 260, 340, 260, 340, 260,
     340, 260
239  REM — Daca acest mesaj este afisat data este incorecta —
240  PRINT „Data invalidă"
249  REM — Revenire in programul principal —
250  RETURN
259  REM — Luna are 31 de zile —
260  IF D>31 THEN 240
270  GOTO 350
279  REM — Luna este februarie; an bisect —
280  IF Y/4<>INT (Y/4) THEN 310
290  IF Y/400=INT (Y/400) THEN 320
300  IF Y/100<>INT (Y/100) THEN 320
309  REM — Nu este an bisect; luna are 28 de zile —
310  IF D>28 THEN 240
319  REM — Este an bisect; luna are 29 de zile —
320  IF D>29 THEN 240
330  GOTO 350
339  REM — Luna are 30 de zile —
340  IF D>30 THEN 240
349  REM — Nr de zile de la 1 IAN pina la 1 al fiecarei luni —
350  DATA 0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334
360  RESTORE
365  FOR H=1 TO M
370  READ A
375  NEXT H
379  REM — Calculeaza nr de zile de la 0, 0, 0 la M, D, Y
380  A=A+Y*365 +INT(Y/4)+D+1-INT(Y/100)+INT (Y/400)
```

```
390  IF INT(Y/4)<>Y/4 THEN 450
410  IF Y/400=INT(Y/400) THEN 450
420  IF Y/100=INT(Y/100) THEN 450
430  IF M>2 THEN 450
440  A=A-1
449  REM — Sfirsit subrutina —
450  RETURN
460  END
```

5.44 — CONVERSIA UNITĂŢILOR DE MĂSURĂ ENGLEZE ÎN UNITĂŢI DE MĂSURĂ EUROPENE

Acest program converteşte în unităţi europene o măsură exprimată în unităţi engleze. Conversiile posibile sînt următoare: 1. Inchi în centimetri; 2. Picioare în centimetri; 3. Picioare în metri; 4. Yarzi în metri; 5. Mile în kilometri; 6. Galoane în kilograme; 7. Uncii în grame; 8. Livre (pfunzi) în kilograme; 9. Grame (pfunzi) în kilograme; 9. Grade Fahrenheit în grade Celsius. Este necesar să se indice valoarea măsurii în unităţi engleze şi numărul conversiei.

Exemplu: Să se efectueze următoarele conversii: 8,5 mile în kilometri; 75° Fahrenheit în grade Celsius; 10 galoane în litri.

```
:RUN
CONVERSIA UNITAŢILOR ENGLEZE IN UNITATI
EUROPENE
(INTRODUCETI 0 PENTRU TERMINARE PROGRAM)
CE CONVERSIE DORITI? 5
VALOAREA DE CONVERTIT? 8.5
8.5 MILE=13.6765 KILOMETRI
CE CONVERSIE DORITI? 11
VALOAREA DE CONVERTIT? 10
10 GALOANE = 37.85 LITRI
CE CONVERSIE DORITI? 0
END PROGRAM AT LINE 540
```

```
10  PRINT „Conversie din unitati engleze in unitati metrice"
20  PRINT
29  REM — 9 TIPURI DE CONVERSIE —
30  DIM C(9)
39  REM — Ciclu pt introducerea factorilor de conversie in C —
40  FOR N=1 TO 9
```

```
 50        READ C(N)
 60  NEXT N
 69  REM — Tabela de conversie —
 70  DATA 2.540, 30.480, .3048, .9144, 1.609, 3.785, .4536, 907.2 ,
     .6214
 89  REM — Se cere numarul conversiei —
 90  PRINT „(Pt terminare program introduceti 0)"
100  PRINT „Ce conversie doriti"
110  INPUT N
119  REM — Sfirsit program ? —
120  IF N=0 THEN 540
129  REM — Conversie posibila —
130  IF N>9 THEN 100
140  PRINT „Valoarea de convertit"
150  INPUT I
159  REM — Efectueaza conversia utilizind factorul de conver-
     sie corespunzător —
160  R=I*C(N)
169  REM — Salt la secventa de afisare —
170  ON N GOTO 180, 200, 220, 240, 260, 380, 460, 480, 500
180  PRINT I; „INCHI =";R; „CENTIMETRI"
190  GOTO 520
200  PRINT I; „PICIOARE =";R; „CENTIMETRI"
210  GOTO 520
220  PRINT I; „PICIOARE =";R; „METRI"
230  GOTO 520
240  PRINT I; „YARZI =";R; „METRI"
250  GOTO 520
260  PRINT I; „MILE =";R; „KILOMETRI"
270  GOTO 520
380  PRINT I; „GALOANE =";R; „LITRI"
390  GOTO 520
460  PRINT I; „PFUNZI =";R; „KILOGRAME"
470  GOTO 520
480  PRINT I; „TONE =";R; „KILOGRAME"
490  GOTO 520
500  R=(I−22)*5/9
510  PRINT I; „GRADE FAHRENHEIT =";R; „GRADE CEL-
     SIUS"
520  PRINT
529  REM — RESTART PROGRAM? —
530  GOTO 100
540  END
```

6. ÎN CONCLUZIE, PRACTICAȚI

Ce ați învățat pe parcursul acestor 5 capitole? Să manipulați cu ușurință principalele instrucții ale limbajului BASIC. De acum înainte, veți putea citi programe BASIC oricît de complicate, le veți putea modifica, sau veți putea realiza chiar creații originale, cu condiția să nu fiți prea ambițioși, pentru început.

Studiați cu atenție programele prezentate în lucrare și, mai ales, nu ezitați să le introduceți și să le executați pe mașina cu care lucrați. Veți vedea că, după ce veți elimina erorile, pe care în mod inevitabil le veți comite, veți reuși, într-un timp relativ scurt, să puneți la punct aceste programe și să le faceți să ruleze corect.

Nu uitați niciodată că toți programatorii, chiar și cei profesioniști, comit erori. Dumneavoastră nu sînteți (încă) programatori profesioniști. Pentru a deveni, va trebui să mergeți mai departe cu studiul limbajului BASIC (și chiar al altor limbaje: PL/1, PASCAL, C). Va trebui, de asemenea, să vă puneți la punct cu metodele matemaice utilizate în grafică, să învățați reguli de programare, modaltăți de a analiza o problemă, de a o descompune în structuri, în algoritmi, lucru pe care autorul nu poate decît să vă încurajeze să îl faceți. Succes!

BIBLIOGRAFIE

KNUTH, D.E., Algoritmi fundamentali. Editura Tehnică, București, 1974.

KNUTH, D.E., Căutarea și sortarea. Editura Tehnică, București, 1976.

KNUTH, D.E., Algoritmi seminumerici, Editura Tehnică, București, 1983.

OSBORNE, A., An introduction to microcomputer, 1980.

* * * BASIC pentru FELIX M118. Manual de utilizare.

LILEN, H., Initiation, Basic, 1980.

VRACIU, G., POPA, A., Metode numerice cu aplicații în tehnica de calcul. Edit. Scrisul Românesc, Craiova, 1982.

CUPRINS

Lector : GHEORGHE FOLESCU
Tehnoredactor : CORNEL CRISTESCU

Bun de tipar 8.XII.1986. Apărut 1987.
Comanda nr. 243.
Coli de tipar 14.

Com. nr. 60 339
Combinatul Poligrafic „Casa Scînteii"
Piața Scînteii nr.1, București
Republica Socialistă România

Lei 12,50

O introducere în microinformatică şi informatică personală prin intermediul limbajului BASIC, unul dintre cele mai răspîndite şi simple limbaje de programare.
Cartea se caracterizează printr-o structură şi conţinut original, folosind o manieră de prezentare adeseori amuzantă şi însoţită de multe desene care înlocuiesc explicaţiile greoaie şi scolastice. Tonul se situează la frontiera dintre glumă şi serios, creînd astfel aparenţa de joc şi făcînd lectura nu numai instructivă, ci şi plăcută.