

A. PETRESCU : coordonator

GH. RIZESCU

F. IACOB

C. CONSTANTINESCU

T. ILIN

E. DECSOV

C. NOVĂCESCU

A. MATEKOVITS

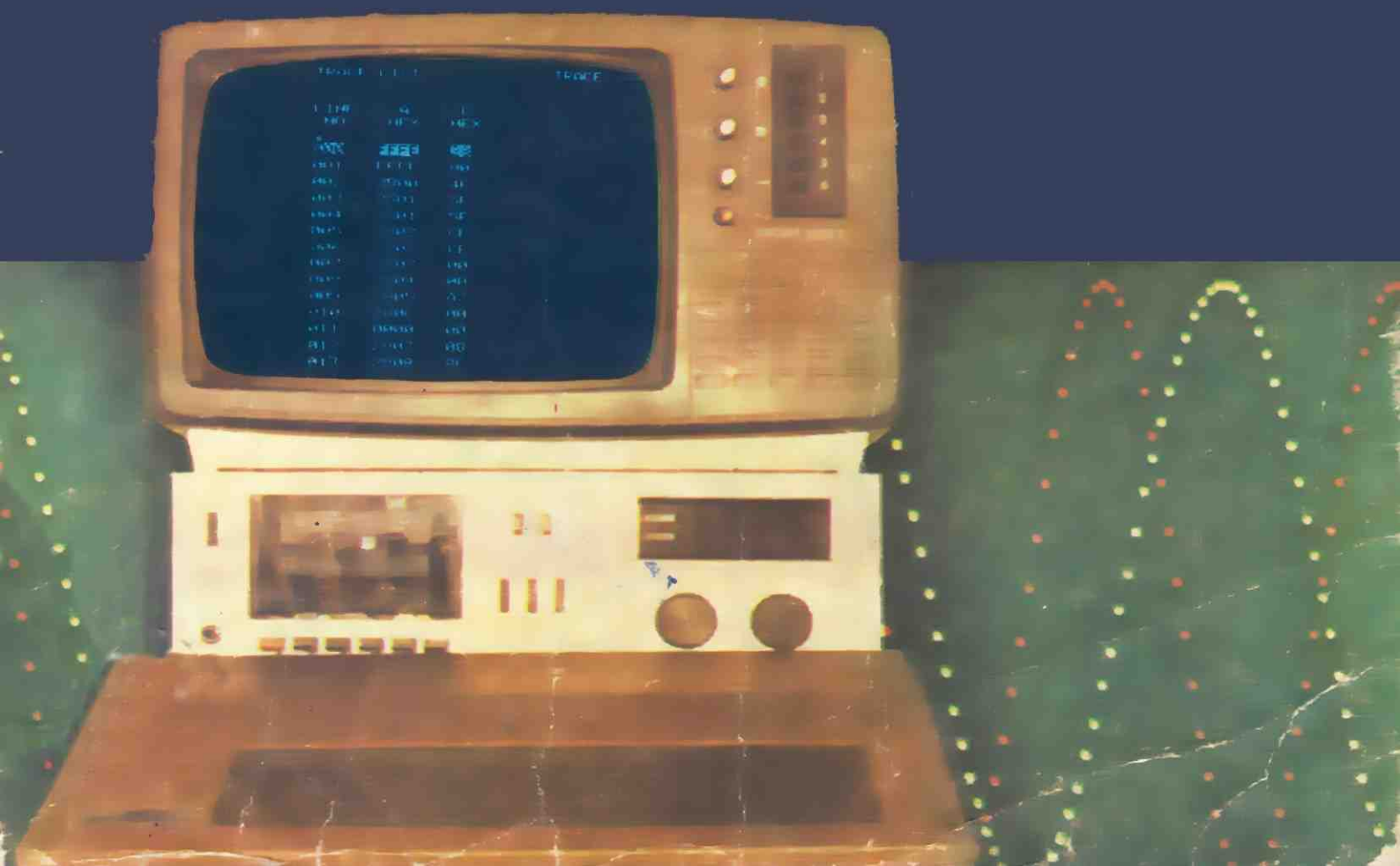
F. BAR

R. BERINDEANU

I. PETRESCU

Volumul 2

TOTUL DESPRE ... CALCULATORUL PERSONAL aMIC



Prof. dr. ing. **Adrian Petrescu**
- coordonator -
Prof. emerit **Gheorghe Rizeşcu**
Ing. asistent univ. **Francisc Iacob**
Ing. asistent univ. **Cornel Constantinescu**
Ing. **Tiberiu Ilin**
Ing. **Eduard Decsoy**
Ing. **Constantin Novăcescu**
Ing. **Agota Motekovits**
Ing. **Florian Bar**
Ing. **Radu Borindeanu**
Elev **Iacob Petrescu**

Totul despre ... calculatorul personal aMIC

Volumul 2



Editura Tehnică
Bucureşti, 1985

Colectivul de elaborare al cărții cuprinde specialiști de la Institutul Politehnic București, Liceul „Dimitrie Cantemir”, București, Întreprinderea de Memorie Electronice Timișoara, ITCI — Timișoara și „Electrotimis” Timișoara.

Contribuția autorilor este următoarea :

- A. Petrescu : coordonare, cap : 8 (p), 9 (p), 13 (p)
- Gb. Rizescu : cap : 10 (p), 12 (p)
- F. Iacob : cap : 8 (p), 12 (p)
- C. Constantinescu : cap : 9 (p), 12 (p)
- T. Ilin : cap : 8 (p), 11 (p), 13 (p)
- E. Decsoy : cap : 8 (p), 11 (p), 13 (p)
- C. Novărescu : cap : 8 (p), 14 (p)
- A. Matekovits : cap : 10 (p), 12 (p), 12 (p), 13 (p), 14 (p)
- F. Bar : cap : 8 (p), 14 (p)
- R. Berindeanu : cap : 8 (p), 14 (p)
- I. Petrescu : cap : 10 (p), 12 (p), anexa 3.

Recenzie : dr. Ing. ADRIAN DAVIDOVICIU

Redactor : Ing. PAUL ZAMFIRESCU

Culegerea și paginarea realizată de o
echipă coordonată de EDUARD GIESSER

Coperta : Arh. SILVIA MIRȚU
Desene : LAURENȚIU ILIESCU
Tehnoredactor : ELLY GORUN

Bun de tipar 10 dec. 1985 Coli tipar, 17,5
C. Z. 681.142

Tiparul a fost executat sub comanda nr. 110
la Întreprinderea Poligrafică „Banat”
Timișoara, Calea Aradului nr. 1.
Republică Socialistă România.



Cuprins^{*)}

Vol. I

<i>Prefață</i>	5
<i>Cuvînt înainte</i>	9
Capitolul 1.	
Clase de microcalculatoare personale și personal-pro- fesionale	21
Capitolul 2.	
Prezentarea generală a mi- crocalculatorului aMIC ...	31
Capitolul 3.	
Structura și funcționarea microcalculatorului aMIC	45
Capitolul 4.	
Microprocesorul Z80. In- terfețele programabile ...	80
Capitolul 5.	
Monitoarele V0.1, V0.2, Z80-V0.	138
Capitolul 6.	
Monitorul DEST	174
Capitolul 7.	
Sistemul de operare rez- dent MATE (Monitor- Asambilor-Text-Editor) ...	194
Anexa 1.	
Monitorul V01. Listing sursă	212
Anexa 2.	
Monitor-Asambilor-Text Editor (MATE). Listing sursă	247

*) În volumul I, la paginile 13-20 se află cuprinsul extins al ambelor volume.

Vol. II

Capitolul 8.

Cuplări de echipamente
periferice, interconectări și
aplicații ale microcalcula-
torului aMIC 7

Capitolul 9.

Limbajul BASIC, pentru
microcalculatorul perso-
nal aMIC. Manual practic 44

Capitolul 10.

Microcalculatorul aMIC în
matematicile elementare și
statistică 98

Capitolul 11.

Microcalculatorul aMIC în
economie și tehnică ... 114

Capitolul 12.

Microcalculatorul aMIC în
învățămînt 138

Capitolul 13.

Microcalculatorul aMIC în
grafică, jocuri, aplicații
diverse 178

Capitolul 14.

Testarea resurselor hard-
ware și a interpretorului
BASIC 221

Anexa 3.

Colecție de programe pen-
tru rezolvarea unor pro-
bleme de matematică din
materia claselor a IX-a și
a X-a 245

Cuplări de echipamente periferice, interconectări și aplicații ale microcalculatorului aMIC.

În acest capitol se vor prezenta o serie de aplicații, care urmăresc atât extinderea gamei de echipamente periferice, cât și utilizarea microcalculatorului în cadrul unor sisteme dedicate. Astfel, se descriu aplicații referitoare la :

- cuplarea la microcalculatorul aMIC a unor LED-uri și comutatoare.
- cuplarea la microcalculatorul aMIC a unui convertor numeric-analogic.
- cuplarea microcalculatorului aMIC cu microcalculatoarele FELIX M18, M118,
- cuplarea la calculatorul aMIC a unui echipament de desenare și indicare pe ecran de tip „JOYSTICK”,
- cuplarea unui convertor analog-numeric la microcalculatorul aMIC,
- utilizarea microcalculatorului aMIC pentru simularea unui circuit logic.
- cuplarea miniimprimantei MIM40 la microcalculatorul aMIC,
- cuplarea unui programator de memorii de tip EPROM la microcalculatorul aMIC,
- microcalculatorul aMIC cuplat cu terminalul DAF2010,
- interfațarea microcalculatorului aMIC cu un minirobot,
- folosirea microcalculatorului aMIC în cadrul unui echipament pentru testarea microsistemelor orientate pe magistrală.
- utilizarea microcalculatorului aMIC ca unitate pentru deservirea mașinilor unelte,
- utilizarea microcalculatorului aMIC ca sistem de înregistrare/redare a parametrilor semicontinui de proces,
- utilizarea microcalculatorului aMIC în laborator pentru prelucrarea datelor provenite din analiza cromatografică.

8.1. Cuplarea unor LED-uri și comutatoare

Utilizând interfața paralelă de care dispune microcalculatorul personal se pot conecta la acesta diferite dispozitive numerice. Cel mai simplu exemplu îl reprezintă cuplarea unor led-uri și comutatoare.

Interfața paralelă este constituită dintr-un circuit 8255 suplimentar, care are liniile porturilor A, B și C, fiecare de câte 8 biți, legate la un conector de pe carcasa microcalculatorului. La portul A s-au conectat 8 comutatoare, fiecare având posibilitatea să furnizeze independent nivelele logice 1 sau 0.

La portul C s-au conectat 8 led-uri, care vor fi stinse sau aprinse, după cum biții corespunzători ai circuitului 8255 sînt 1 sau 0. Schema este prezentată în figura 8.1.

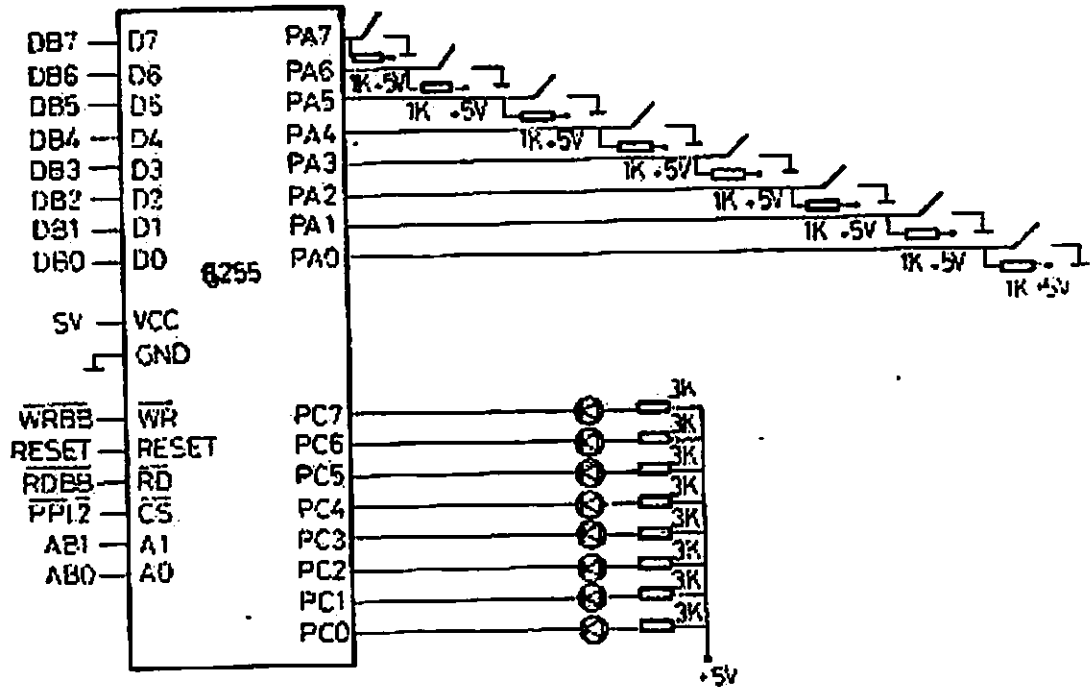


Fig. 8.1. Cuplarea unor led-uri și comutatoare.

Pentru această aplicație circuitul 8255 suplimentar a fost programat în modul 0 de lucru cu portul C pentru ieșire, iar porturile A și B pentru intrare. Adresele acestor porturi sînt :

- 40H : portul A ;
- 41H : portul B ;
- 42H : portul C ;
- 43H : portul de comandă.

În continuare se prezintă programul care realizează citirea permanentă a comutatoarelor și afișarea stării acestora la led-uri.

Program de aplicație pentru interfața paralelă

```

START: MVI  A, 92H ; încărcă în acumulator cuvîntul de comandă
        OUT  43H ; transmite în circuitul 8255
BUCLA: IN  40H ; citește portul A în acumulator
        CMA ; complementează acumulatorul
        OUT  42H ; transmite în portul C
        JMP BUCLA ; reluare.

```

8.2. Cuplarea unui convertor numeric-analogic

Microcalculatorul personal poate comanda un proces analogic simplu prin interfațarea unui convertor numeric-analogic. Valoarea numerică ce este furnizată convertorului se obține ca urmare a unor calcule sau prin prelucrarea unor date externe.

În acest exemplu s-a realizat interfașarea unui convertor numeric-analogic pe 8 biți, DAC08, la magistrala sistemului. Acest lucru a fost posibil deoarece semnalele magistralei de date, adresele și comenzile au fost legate la un conector de pe carcasă. Schema bloc a montajului este prezentată în figura 8.2.

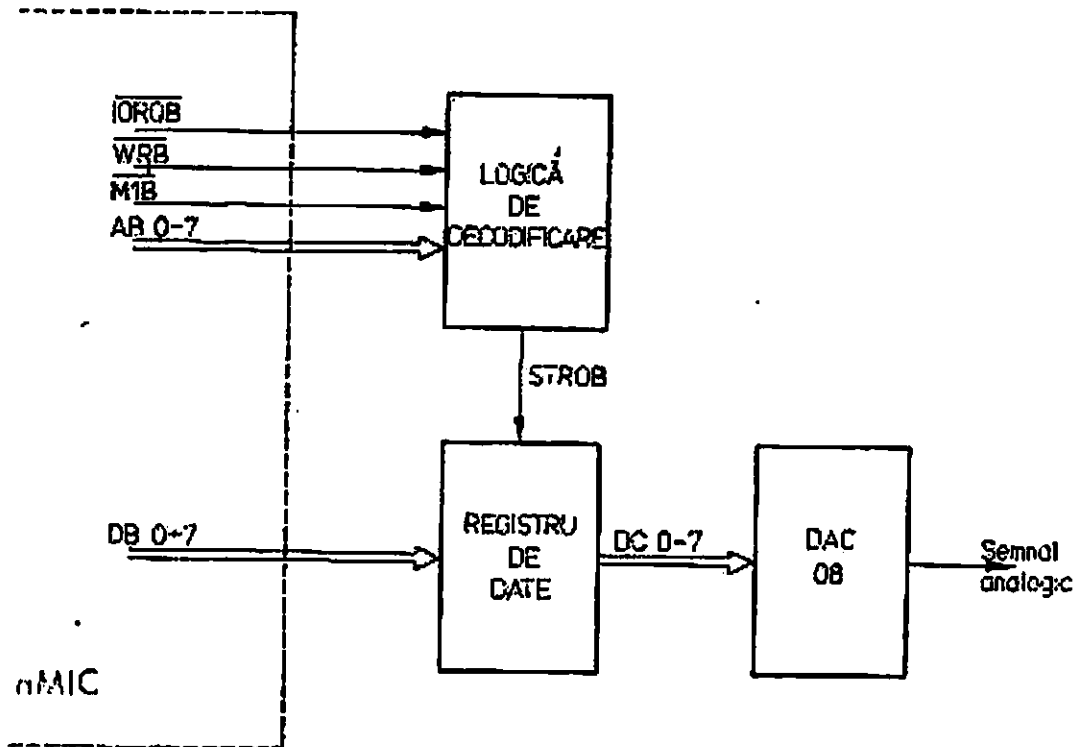


Fig. 8.2. Schema bloc pentru cuplarea unui convertor numeric analogic.

Aceasta se compune din logica de decodificare a adresei, un registru de date de 8 biți și convertorul DAC08. Schema electrică completă a interfașării convertorului se prezintă în figura 8.3.

Realizarea semnalului analogic de la ieșire costă în înscrierea în registrul de date a unor valori numerice pe 8 biți la momente de timp bine stabilite.

În continuare se prezintă programul în limbaj de asamblare care realizează la ieșirea convertorului un semnal în dinți de fierăstrău. Transmiterea datelor se face prin instrucțiunea de ieșire (OUT) cu adresa 60H.

Program de aplicații pentru DAC08.

```
START: MVI  A, 0 ; inițializarea acumulatorului cu zero
BUCLA: OUT  60H ; ieșire în registrul de date
        INR  A ; incrementează acumulatorul
        JMP  BUCLA ; salt la reluare.
```

Pentru a varia perioada semnalului analogic se poate utiliza o subrutină de așteptare /

```
START: MVI  A, 0 ; inițializarea acumulatorului cu zero
BUCLA: OUT  60H ; ieșire în registrul de date
        INR  A ; incrementează acumulatorul
        CALL WATT ; reținează perioada semnalului
        JMP  BUCLA ; salt la reluare
```

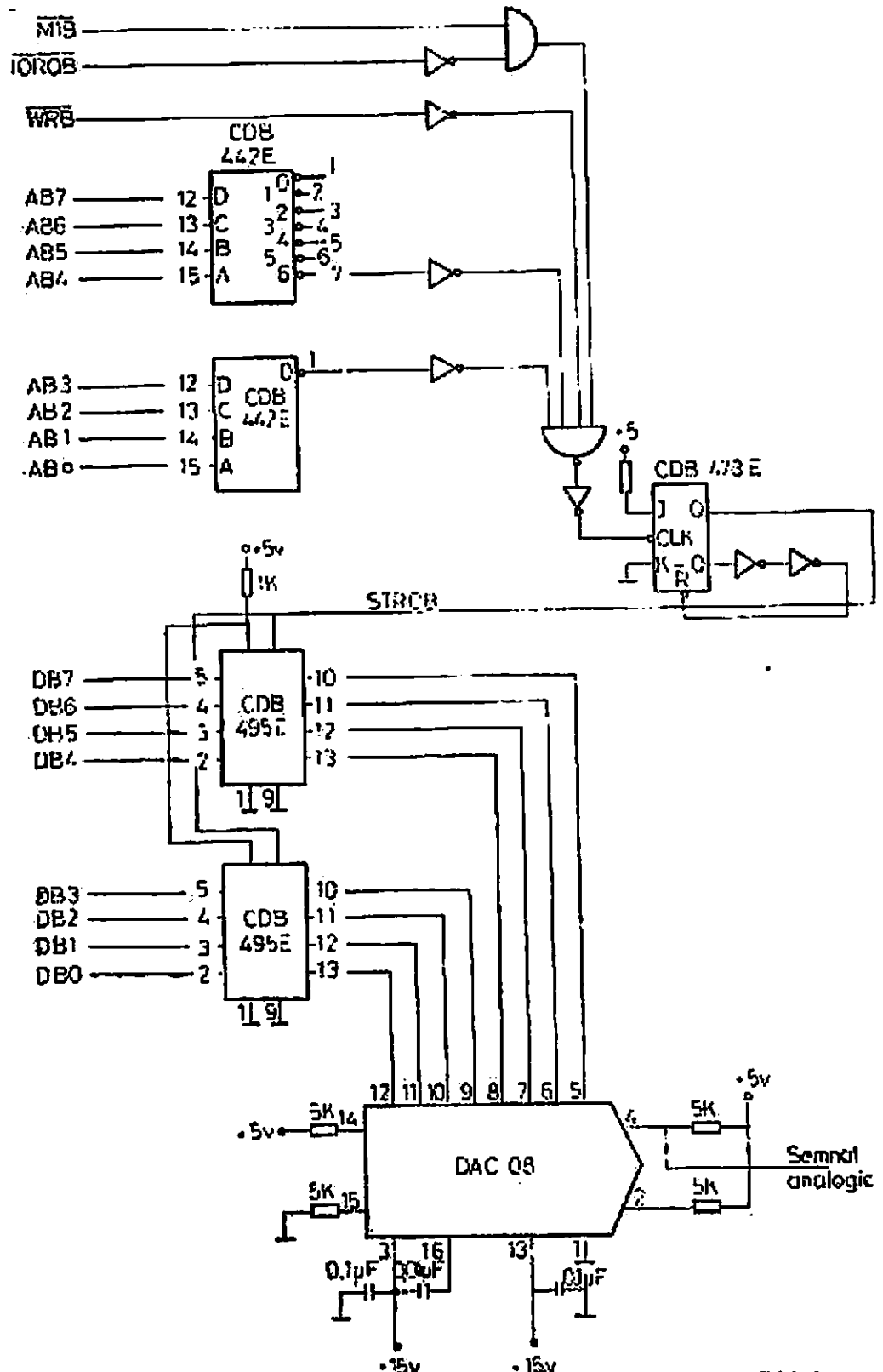



Fig. 3.3. Schema cuplării convertorului numeric-analog DAC08 la magistrala microcalculatorului „MIC”.

```

WAIT: LXI  D, 500H ; această valoare reglează perioada
      PUSH PSW   ; salvează acumulatorul
EPI:  DEX  D     ; decrementează contorul
      MOV  A, D  ; încaută registrul D în A
      ORL  E     ; SAU logic între A și E
      JNZ  EPI   ; reluare dacă diferit de zero
      POP  PSW   ; refăce acumulatorul
      RET                ; revenire în programul principal.

```

8.3. Interconectarea cu microcalculatoarele Felix M18, M118

Cuplarea microcalculatorului personal la sistemele de calcul FELIX M18 și M118 se face prin interfața serială standard RS 232.

În acest scop „aMIC” a fost dotat cu un conector de 25 de contacte la care sînt legate semnalele circuitului 8251. Un conector asemănător există și la M18/118.

Conectarea celor două sisteme se face printr-un cablu cu trei linii după schema prezentată în figura 8.4. Se utilizează conectori RACK cu 25 de contacte, de tip „tafă”.

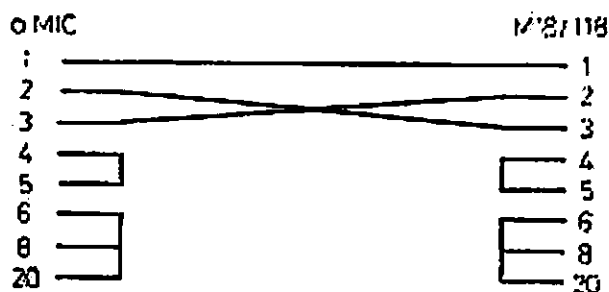


Fig. 8.4. Cablul de interconectare „aMIC” — FELIX M18/118.

Ceilați pini din conectori, neindicați în desen, sînt lăsați neconecțați.

Pentru realizarea comunicației pe fiecare din cele două sisteme rulează cîte un program. În continuare se prezintă un exemplu care transferă 100 de octeți din memoria microcalculatorului FELIX M18/118 de la adresa 6000H în memoria microcalculatorului „aMIC” la adresa 6600H. Un asemenea procedeu este deosebit de util, deoarece pe M18/118 se pot introduce și pune la punct programe complexe pentru „aMIC” utilizîndu-se resurse software evaluate disponibile pe aceste sisteme, apoi aceste programe sînt transferate la microcalculatorul personal pentru a fi rulate.

În scopul realizării transferului de date pe microcalculatorul „aMIC” se rulează un program de recepție a octeților transmiși pe linia serială.

1 Program de recepție a datelor pe linia serială

```

START: MVI  A, 0CEH ; cuvîntul de mod
      OUT  1       ; transmite la 8251
      MVI  A, 27H  ; cuvîntul de comandă

```

```

OUT 1 ; transmite la 8251
INX H, 6000H ; în H, L adresa de început
MVI C, 100 ; contorul de octeți
BUCLA : IN 1 ; bucla de citire a stării circuitului 8251
ANI 2 ; test bit 1
JZ BUCLA ; dacă 0, așteaptă
IN 0 ; citește octet
MOV M, A ; înscrie în memorie
INX H ; incrementează adresa
DCR C ; decrementează contorul
JNZ BUCLA ; dacă diferit de zero, reface
JMP 0 ; altfel intră în monitor

```

Pe microcalculatorul PELIX M18/118 rulează un program de transmisie a octeților din memorie pe linia serială.

Program de transmisie a datelor pe linia serială

```

START : MVI A, 40H ; cuvântul de comandă pentru 8251
OUT 0F7H ; realizează inițializarea circuitului
MVI A, 0CEH ; cuvântul de mod
OUT 0F7H ; transmite la 8251
MVI A, 27H ; cuvântul de comandă
OUT 0F7H ; transmite la 8251
INX H, 6000H ; în H, L adresa de început
MVI C, 100 ; contorul de octeți
BUCLA : IN 0F7H ; citește starea
ANI 4 ; test bit 2
JZ BUCLA ; dacă este zero, așteaptă
MOV A, M ; citește octet din memorie
OUT 0F6H ; transmite la 8251
INX H ; incrementează adresa
DCR C ; decrementează contorul
JNZ BUCLA ; dacă diferit de zero, reluare
RST 0 ; revenire în monitor

```

Cele două programe se execută în același timp, dar cu mențiunea că programul de recepție a datelor se lansează primul.

8.4. Cuplarea unui JOYSTICK

Joystick-ul este un dispozitiv utilizat în conjuncție cu terminalele grafice pentru specificarea manuală, de către utilizator, a coordonatelor unui punct pe ecran.

Joystick-ul constă dintr-o articulație sferică, cu o manetă fixată de partea sferică mobilă, ale cărei deplasări unghiulare sînt transformate în variații ale rezistenței electrice a doi potențiometri, corespunzînd coordonatei orizontale, respectiv verticale. Variațiile de rezistență sînt transformate apoi în variații de semnal electric, care — la rîndul lor — sînt convertite în valori numerice. Aceste valori numerice sînt preluate de sistemul de calcul și, prin metode software sau hardware (la echipamentele care cer viteză mare de lucru), sînt transformate într-un punct sau cursor grafic, afișat pe ecran. Este posibilă afișarea întregului traseu impus punctului curent, adică desenarea comandată

de la joystick. Este, de asemenea, posibilă introducerea unor funcții suplimentare, care să permită utilizarea acestui dispozitiv în diferite aplicații.

Cuplarea unui joystick la microcalculatorul „aMIC” se face prin intermediul interfeței paralele, utilizând numai patru biți ai circuitului 8255. Schema prezentată în figura 8.5 conține două circuite CDB4121E.

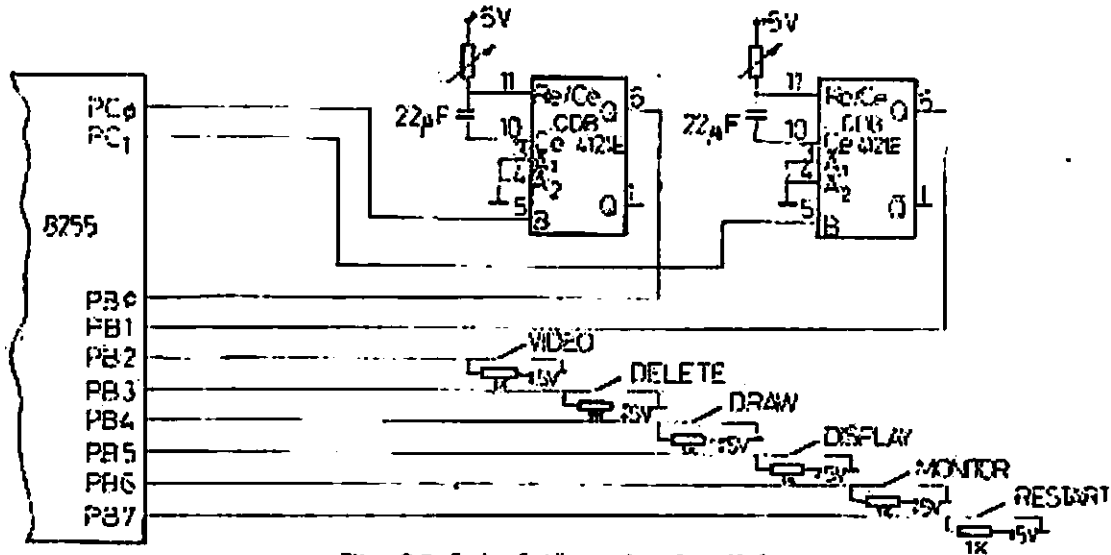
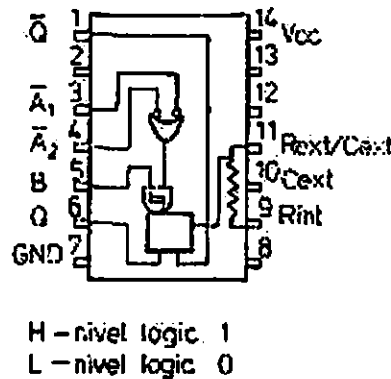


Fig. 8.5. Interfață pentru joystick.

CDB4121E este un circuit basculant monostabil având două intrări de condiționare active pe front negativ și o intrare activă pe front pozitiv. Tabela de funcționare și configurația pinilor sînt prezentate în figura 8.6. Durata impulsului obținut la ieșire se poate varia între 20 ns și 28 s utilizînd componente pasive externe. Fără componente externe, avînd intrarea Rint conectată

INTRĂRI			IEȘIRI	
\bar{A}_1	\bar{A}_2	B	Q	\bar{Q}
L	X	H	L	H
X	L	H	L	H
X	X	L	L	H
H	H	X	L	H
H	↑	H	↑	↑
↑	↑	H	↑	↑
L	X	↑	↑	↑
X	L	↑	↑	↑



H - nivel logic 1
L - nivel logic 0

Fig. 8.6. Tabela de funcționare și configurația pinilor la circuitul CDB 421E.

la V_{cc} , iar intrările C_{ext} și R_{ext}/C_{ext} lăsate neconectate, se obține un impuls cu durata specifică de 30–35 ns.

Pentru valori ale capacității externe cuprinse între 10 pF și 10 μF iar ale rezistenței externe cuprinse între 2 kΩ și 40 kΩ durata impulsului t se calculează cu formula:

$$t_w = 0,7 C_{ext} \cdot R_{ext}$$

În care capacitatea este în pF, rezistența în kΩ, iar durata impulsului se obține în ns.

Pentru determinarea valorilor potențioetrelor, deci pentru a determina poziția curentă pe ecran, se furnizează un impuls pozitiv (trecere de la 0 logic la 1 logic) pe bitul 0, respectiv 1 al portului C din circuitul 8255. După furnizarea unui asemenea front pozitiv, se așteaptă prin program o perioadă de timp care corespunde cu valoarea minimă a rezistenței potențioetruului (poziția stînga pentru orizontală, respectiv poziția sus pentru verticală), după care începe conturizarea pînă la terminarea impulsului furnizat de monostabil (trecerea din 1 logic în 0 logic a ieșirii Q a acestuia) și care este citit prin portul B în bitul 0, pentru orizontală, respectiv în bitul 1, pentru verticală. Bucla de controlare din program este în așa fel calibrată încît, cu aproximație, să se înregistreze o valoare cuprinsă între 0 și FFH, corespunzătoare celor două poziții extreme (pe orizontală și — de asemenea — și pe verticală). În figura 8.7 se prezintă semnificația valorilor celor două contoare și modul de determinare a adresei octetului pentru memoria ecran și al bitului din cadrul octetului de date. Deoarece memoria ecran este cuprinsă între adresele 4000H—5FFFH, în pozițiile cele mai semnificative A15—A18 se găsește configurația 010

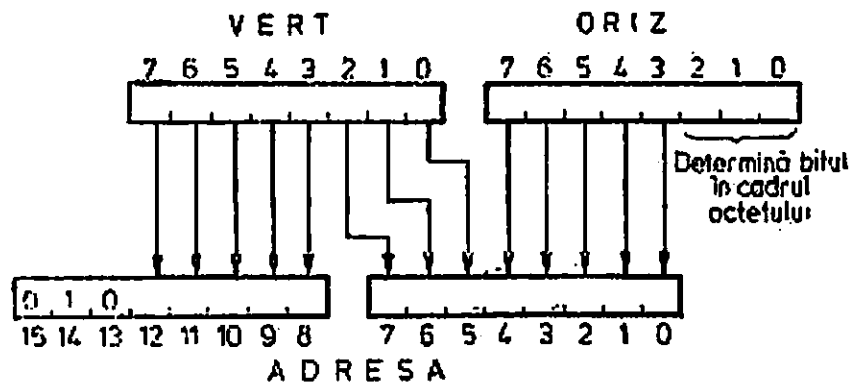


Fig. 8.7. Determinarea adresei din memoria video și octetul de date, pornind de la valorile celor două contoare VERT și ORIZ.

Ceilalți biți de care dispune interfața paralelă se pot utiliza pentru implementarea unor funcții. În exemplul considerat se utilizează încă 6 biți din portul B al circuitului 8255, avînd următoarele semnificații:

- PB7: RESTART — reiașează programul de joystick;
- PB6: MONITOR — execută revenirea în monitor (start de la adresa 0)

- PB5: DISPLAY — anulează o funcție anterioară, realizând numai afișarea punctului curent, fără să modifice imaginea de pe ecran (deci nici memoria video);
- PB4: DRAW — marchează traseul parcurs de la punctul curent pe ecran prin puncte aprinse (biți 0 în memoria video);
- PB3: DELETE — șterge contururile de pe traseul parcurs de punctul curent, adică stinge punctele aprinse. În memoria video se înscrie 1 în biții egali cu 0 de pe traseu;
- PB2: VIDEO — schimbă polaritatea punctelor de pe traseu (execută video invers). Această funcție înscrie 1 în biții egali cu 0, respectiv înscrie 0 în biții egali cu 1 din memoria video.

Programul de interfață pentru joystick este scris în limbaj de asamblare, iar organigrama sa este prezentată în figura 8.8. După lansarea programului în execuție, dacă nu se apasă nici un buton funcțional, ecranul este șters și se afișează punctul curent care se deplasează simultan cu acționarea joystick-ului.

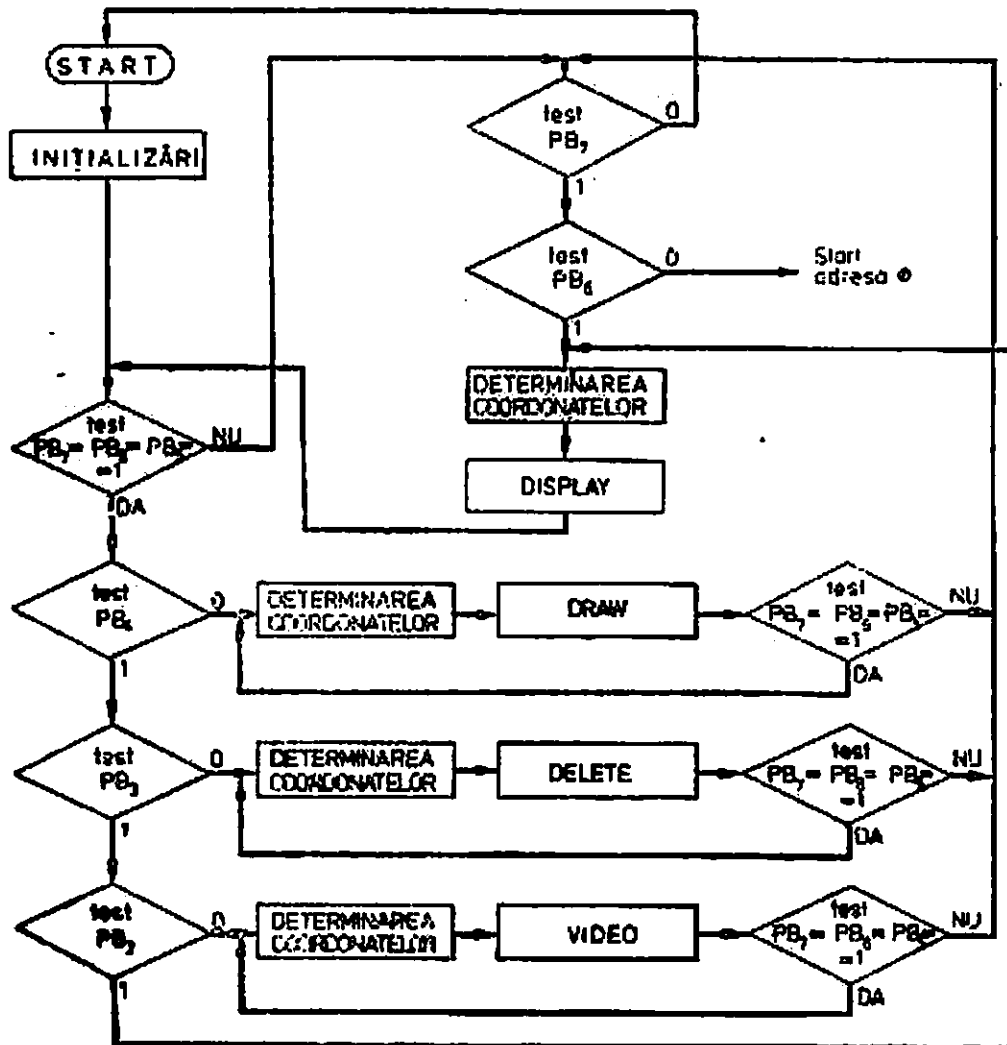


Fig. 8.8. Programul de interfață pentru joystick (organigramă)

Dacă se apasă oricare din butoanele DRAW, DELETE sau VIDEO, se execută în continuare funcția respectivă (fără ca să se mențină butonul apăsat), pînă cînd se apasă unul din butoanele RESTART, MONITOR sau DISPLAY.

Programul prezentat în acest exemplu realizează o traducere a poziției articulației joystick-ului în coordonate pe ecran (respectiv adresă și octet de date pentru memoria video). Dezavantajul acestei soluții este o sensibilitate deosebită a punctului curent (cursorului) de pe ecran față de orice mișcări ale tijei dispozitivului. Pentru eliminarea acestui neajuns se pot utiliza alte metode de interfașare a joystick-ului, ca de exemplu:

— poziția tijei (a articulației) să indice direcția dintr-un număr finit și eventual mie de direcții posibile și viteza (trepte discrete de viteză) pentru cursorul de pe ecran. Deplasarea se face din poziția curentă în direcția și cu viteza date de joystick. Poziția centrală a tijei corespunde cu poziția de repans a cursorului. Această soluție este ilustrată în figura 8.9, pentru 8 direcții posibile și două trepte de viteză:

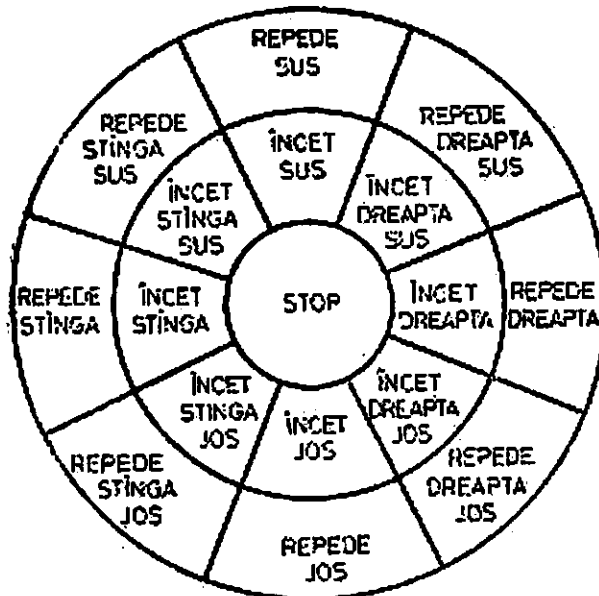


Fig. 8.9. Interpretarea pozițiilor tijei joystick-ului.

— cu ajutorul joystick-ului se marchează un punct pe ecran după care o funcție implementată pe sistem realizează unirea punctului curent cu noul punct marcat; în felul acesta se pot executa desene pe ecran cu o precizie maximă, în limitele rezoluției disponibile.

; Program de interfașă pentru joystick

```

ORG 0100H
START: LXI SP, STIVA ; inițializare indicator de stivă
        CALL CLEAR ; șterge ecranul televizorului
        MVI A, 02H ; cuvînt de comandă : mod 0
        OUT 43H ; inserie în 8255
        LXI H, 4000H ; inițializare ADP2
        SHLD ADP2
        MOV A, M ; citește octet din memoria video
        STA DAT2 ; inițializează DAT2
ET2: CALL PORTB ; citește port B și test biții 7, 6, 5
        JZ RT1 ; salt dacă PB7=PB6=PB5=1
ET4: RAL ; în CY trece PB7 complementat
        JC START ; dacă PL7=0, relansare program
        RAI ; în CY trece PB6 complementat
        JC 0 ; dacă PB6=0, revenire în monitor

```

; secvența DISPLAY pentru PB5=0

```

DISPL: CALL COORD      ; determină coordonatele curente
        LHLD ADR2      ; în H, L vechea adresă
        LDA  DAT2      ; în H, L vechiul octet
        MOV  M, A      ; reface în memoria video
        LHLD ADRI      ; în H, L noua adresă
        MOV  A, M      ; citește octetul de date
        STA  DAT2      ; salvează data
        SHLD ADRI      ; și adresa
        MOV  B, A      ; salvează în registrul B
        LDA  DAT1      ; încarcă în A noul octet de date
        ANA  B         ; aprinde punctul curent
        MOV  M, A      ; înscrie în memoria video
        JMP  ET2       ; reluare
ET1:   MOV  A, B      ; reface A cu conținutul portului B
        ANI  10H      ; testează bitul PB4
        JNZ  ET3      ; salt dacă PB4=1

```

; secvența DRAW pentru PB4=0

```

DRAW:  CALL COORD      ; determină coordonatele curente
        LHLD ADRI      ; în H, L noua adresă
        MOV  B, M      ; în B octetul de date vechi
        LDA  DAT1      ; citește noul octet de date
        ANA  B         ; aprinde punctul curent
        MOV  M, A      ; înscrie în memorie
        CALL PORTB     ; citește port B și testă biții 7, 6, 5
        JNZ  ET4       ; salt dacă cel puțin un bit este 0
        JMP  DRAW      ; altfel, reluare
ET3:   MOV  A, B      ; reface A cu conținutul portului B
        ANI  8        ; testează bitul PB3
        JNZ  ET5      ; salt dacă PB3=1

```

; secvența DELETE pentru PB3=0

```

DELETE: CALL COORD      ; determină coordonatele curente
        LHLD ADRI      ; în H, L noua adresă
        MOV  B, M      ; citește în B vechiul octet
        LDA  DAT1      ; încarcă în A noul octet
        CMA          ; complementează
        ORA  B         ; stinge punctul curent
        MOV  M, A      ; înscrie în memoria video
        CALL PORTB     ; citește port B și testează biții 7, 6, 5
        JNZ  ET4       ; salt dacă cel puțin un bit este 0
        JMP  DELETE    ; reluare
ET5:   MOV  A, B      ; reface A cu conținutul portului B
        ANI  4        ; testează bitul PB2
        JNZ  DISPL     ; salt dacă PB2=1

```

; secvența VIDEO pentru PB2=0

```

VIDEO:  CALL COORD      ; determină coordonatele curente
        LHLD ADRI      ; în H, L noua adresă
        MOV  B, M      ; în B octetul de date vechi
        LDA  ADRI      ; în A noul octet
        CMA          ; complementează octet
        XRA  B         ; complementează punctul curent
        MOV  M, A      ; înscrie în memorie
        CALL PORTB     ; citește portul B și testează biții 7, 6, 5
        JNZ  ET4       ; salt dacă cel puțin un bit este 0
        JMP  VIDEO     ; reluare

```


! subrutina COORD

! determină coordonatele punctului curent

! furnizează în ADRI adresa din memoria video

! și în DAT1 octetul de date

```
COORD: MVI D, 2           ; setează bitul 1
        CALL XY           ; determină coordonata X
        STA ORIZ         ; memorează și variabila ORIZ
        MVI D, 1         ; setează bitul 0
        CALL XY         ; determină coordonata Y
        STA VERT         ; memorează în variabila VERT
```

! secvența care determină adresa pentru memoria ecran

! și octetul de date

```
LXI H, 4000H           ; inițializare H, L
LDA ORIZ              ; citește conținutul pentru orizontală
ANI 0FBH             ; anulează biții A2-A0
RRC                  ; aduce A7-A3 în pozițiile A4-A0
RRC
RRC
MOV L, A              ; încarcă în registrul L
LDA VERT             ; citește conținutul pentru verticală
ANI 7                ; preia biții A2-A0
RRC                  ; aduce informația în pozițiile A7-A5
RRC
RRC
ORA I                ; concatenează cu informația din L
MOV I, A             ; încarcă în registrul I octetul mai puțin semnificativ de
                    ; adresă
LDA VERT             ; citește conținutul pentru verticală
ANI 0FBH            ; preia biții A7-A3
RRC                  ; aduce în pozițiile A7-A0
RRC
RRC
ORA H                ; concatenează cu informația din H
MOV H, A             ; încarcă în registrul H octetul mai semnificativ de adresă
LHLD ADRI           ; deține adresa în ADRI
LDA ORIZ            ; citește conținutul pentru orizontală
ANI 7                ; preia biții A2-A0
MOV L, A            ; încarcă în L
MVI A, 7FH         ; inițializare octet date
INR L
AD2: DCR L           ; poziționează indicatorii
     JZ AD1          ; dacă L=0, stop
     RRC             ; altfel, deplasează bit=0 (punct aprins)
     JMP AD2         ; reluare
AD1: STA DAT1        ; depune octetul de date la adresa DAT1
     RET
```

! subrutina XY

! poziționează conținutul pentru orizontală

! sau pentru verticală

! furnizează rezultatul în A (conținut între 0 și FFH)

```
XY: MVI A, 0         ; bit 0, bit 1=0 din portul C al circuitului 8255
    OUT 42H          ; trimite la 8255
    MOV A, D         ; poziționează pe 1 bitul corespunzător pentru orizontală
                    ; sau pentru verticală
    OUT 42H         ; trimite la 8255, declanșează monostabilul
    MVI C, 0FFH     ; inițializare conținut
XY1: IN 41H          ; citește port B al circuitului 8255
     ANA D           ; test bit 1/0
```

```

        JZ   XY1           ; așteaptă impulsul monostabilului
        MVI  A, 4           ; secvența se execută de patru ori
WAIT:   MVI  B, 84H        ; contor
        DCR  B
        JNZ  WAIT+2
        DCR  A
        JNZ  WAIT
; începe incrementarea contorului
XY2:   INR  C
        IN   41H           ; citește port B
        ANA  D             ; test bit 0/1
        NOP                ; calibrarea contorizării
        NOP
        NOP
        JNZ  XY2           ; așteaptă terminarea impulsului
        MOV  A, C           ; transferă rezultatul în A
        RET
; zona de variabile program
ORIZ:  DS    1             ; contorul pentru orizontală
VERT:  DS    1             ; contorul pentru verticală
ADR1:  DS    2             ; adresa nouă pentru memoria video
ADR2:  DS    2             ; adresa veche
DAT1:  DS    1             ; noul octet de date
DAT2:  DS    1             ; vechiul octet de date
        DS    10           ; stiva program
STIVA: DS    1
        END  START

```

8.5. Cuplarea unui convertor analog-numeric

Microcalculatorul personal are posibilitatea să efectueze măsurări de mărimi analogice, eventual extrase dintr-un proces, prin conectarea unui convertor analog-numeric. Interfașarea convertorului la sistem se face prin circuitul 8255 suplimentar (interfașa paralelă). Se poate executa conversia unei singure mărimi analogice sau a mai multora succesiv; în acest ultim caz, măsurile analogice sînt aduse la intrarea convertorului prin intermediul unui multiplexor analogic. Selecția intrărilor multiplexorului se face prin intermediul circuitului 8255.

Schema de principiu este prezentată în figura 8.10. Se utilizează un multiplexor analogic 16 : 1, iar selecția se realizează prin biții PC3-PC0 al portului C din circuitul 8255. Convertorul analog-numeric este pe 12 biți, avînd o intrare de comandă START conversie, conectată la bitul PC7 al circuitului 8255. Un front pozitiv pe această intrare lansează operația de conversie. Biții de date inferiori D0-D7 sînt conectați la pinii PA0-PA7, iar biții de date superiori D8-D11 la pinii PB0-PB3. De asemenea, convertorul mai dispune de un semnal pentru a anunța sfîrșitul conversiei, STOP conversie, conectat la pinul PB7.

În continuare se prezintă lista instrucțiunilor de intrare/ieșire și semnificațiile fiecăreia :

OUT 42H; biții A3-A0 selectează una din cele 16 intrări analogice

OUT 42H; bitul A7=1 lansează operația de conversie

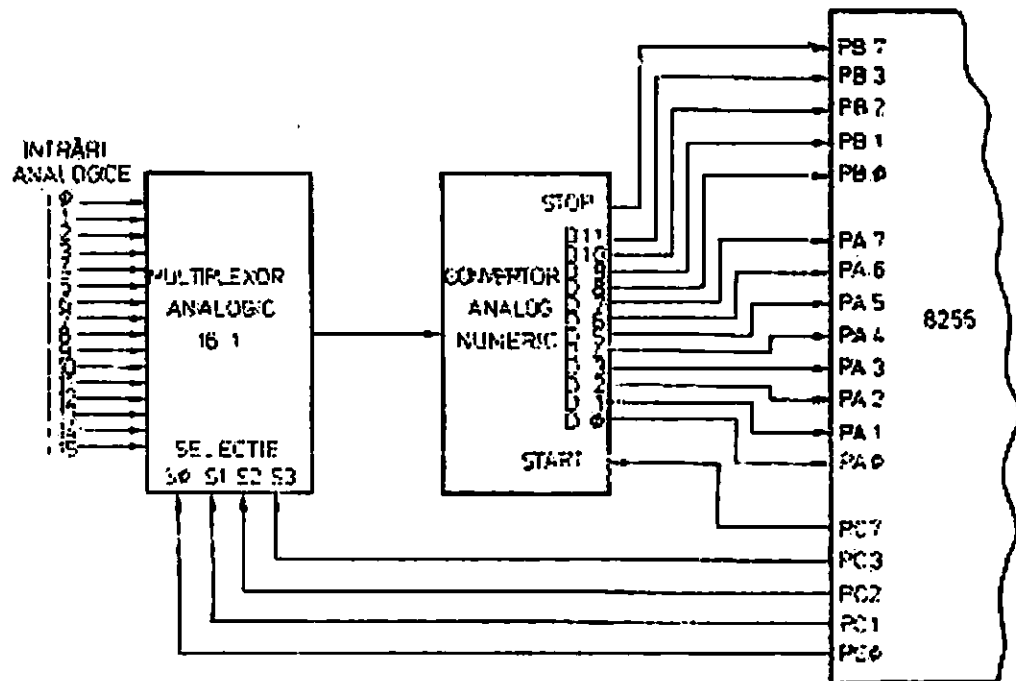


Fig. 8.10. Cuplarea unui convertor analog-numeric.

IN 41H ; citește în A7 bitul STOP conversie

IN 40H ; citește în A7-A0 biții de date D7-D0 numai după terminarea
; conversiei

IN 41H ; citește în A3-A0 biții de date D11-D8

OUT 42H ; bitul A7=0 resetează convertorul

Programul care urmează execută succesiv conversia analog-numerică a celor 16 intrări inseriind rezultatele în memorie, fiecare pe câte doi octeți începînd de la adresa LISTA.

; Program de interfațare a unui convertor analog-numeric

```

MVI A, 92H ; programare circuit 8255
OUT 43H ; trimite în portul de comandă
LXI H, LISTA ; H, L conțin adresa în zona de memorie
MVI B, 0 ; selecție intrare analogică
ET1: MOV A, B ; transferă cod de selecție în A
OUT 42H ; resetează convertorul A7=0
ORI 80H ; A7=1
OUT 42H ; start conversie
ET2: IN 41H ; citește starea
RAL ; A7=STOP trece în CY
JNC ET2 ; dacă STOP=0, așteaptă
IN 40H ; citește biții de date D7-D0
MOV M, A ; inserie în memorie
INX H ; incrementează adresa de memorie
IN 41H ; citește biții de date D11-D8
ANI 0FH ; selectează biții A3-A0
MOV M, A ; inserie în memorie

```

INC	H	; incrementează adresa de memorie
INR	B	; incrementează conținutul de selecție
ANI	0FH	; test și șterge bitii
JNZ	ETI	; relația dacă nu este gata
JMP	NEXT	; altfel execută prelucrarea datelor

8.6. Simularea unui circuit logic

Microcalculatorul „MIC” poate fi utilizat în proiectarea schemelor logice, înlocuind faza de sinteză a circuitelor și de punere la punct cu ajutorul osciloscopului, analizorului logic etc.

În acest fel se poate economisi timp și efort material. Pentru exemplificare, se consideră o interfață simplificată care conectează un echipament periferic de ieșire la un microsistem construit tot cu microprocesorul Z80 (figura 8.11).

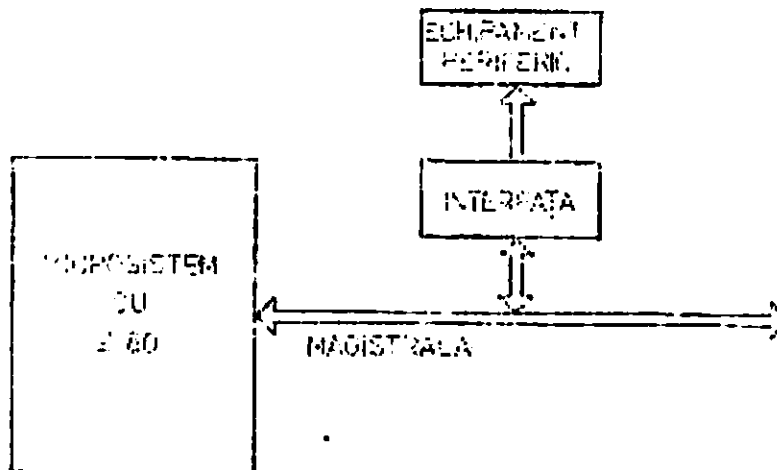


Fig. 8.11. Interfațarea unui echipament periferic la un microsistem cu Z80.

Interfața este cuplată la magistrală prin următoarele semnale (figura 8.12):

— **RESET**: activarea acestui semnal are ca efect inițializarea interfeței și abandonarea eventualei operații curente;

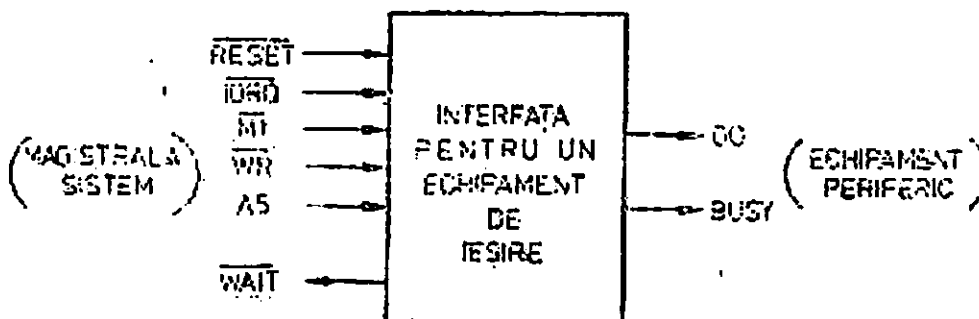


Fig. 8.12. Semnalele conectate la interfață.

În loc să se construiască fizic această schemă și să se verifice prin metode hardware, utilizând osciloscop, analizor logic, etc. ... funcționarea corectă se poate verifica prin program, utilizând microcalculatorul „aMIC”. Conectarea microcalculatorului la microsistemul cu Z80 se poate face prin interfața paralelă. În acest scop se utilizează un circuit Z80-P10 cuplat extern la magistrala „aMIC”-ului. În figura 8.14 este prezentat circuitul Z80-P10 și semnalele de cuplare la magistrală.

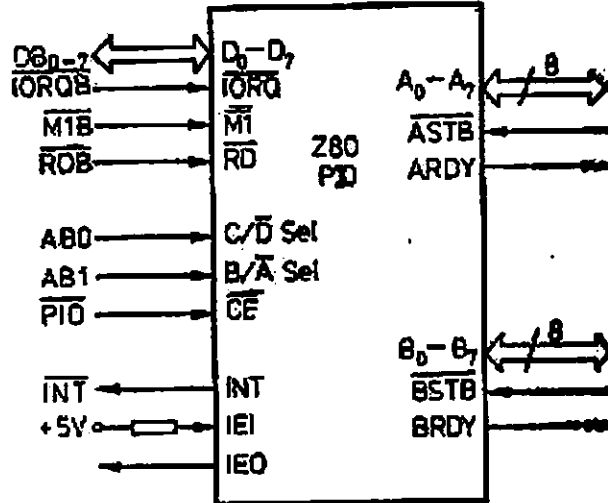


Fig. 8.14. Cuplarea circuitului Z80-P10 la magistrala microcalculatorului „aMIC”.

Astfel, în microsistemul cu Z80 interfața de testat a fost înlocuită cu microcalculatorul „aMIC” (fig. 8.15). Semnalele interfeței de cuplare la unitatea centrală de prelucrare și la echipamentul periferic de ieșire, au fost legate la liniile portului A din circuitul Z80-P10, programat în modul 3 (control bit).

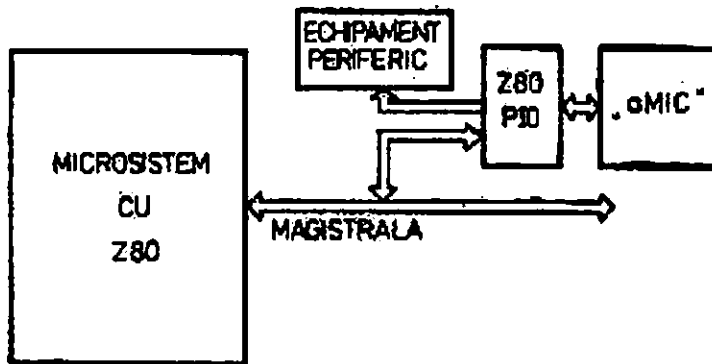


Fig. 8.15. Schema de principiu pentru simularea interfeței.

Deoarece intrarea \overline{CE} a circuitului a fost conectată la ieșirea 3 a decodificatorului (CDB442E) de adrese de I/E (semnalul PIO), iar terminalele C/D Sel și B/A Sel la liniile de adresă AB0, respectiv AB1 rezultă adresele porturilor din circuit (biții neutilizați s-au considerat 0):

- 60H: port A date;
- 61H: port A control;
- 62H: port B date;
- 63H: port B control.

Asignarea semnalelor interfeței la liniile portului A este prezentată în figura 8.16.

Programul de simulare cuprinde trei secvențe principale:

— secvența de inițializare, inclusiv programarea circuitului Z80-PIO (figura 8.17);

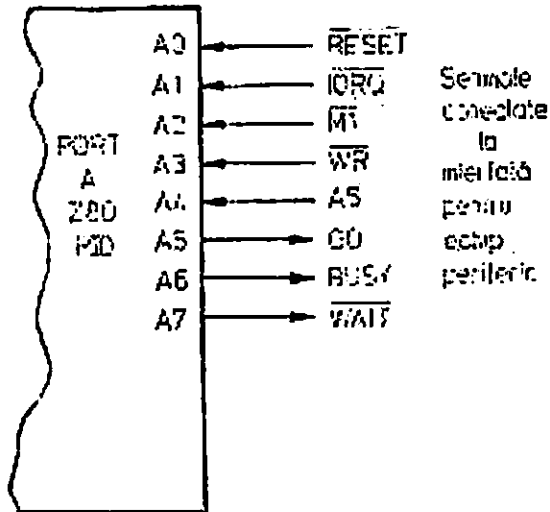


Fig. 8.16. Asignarea semnalelor interfeței la liniile portului A din Z80-PIO.

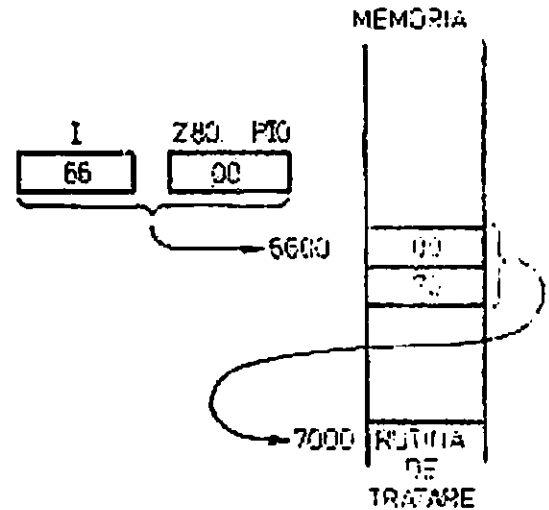


Fig. 8.18. Răspunsul la întreruperea generată de Z80-PIO.

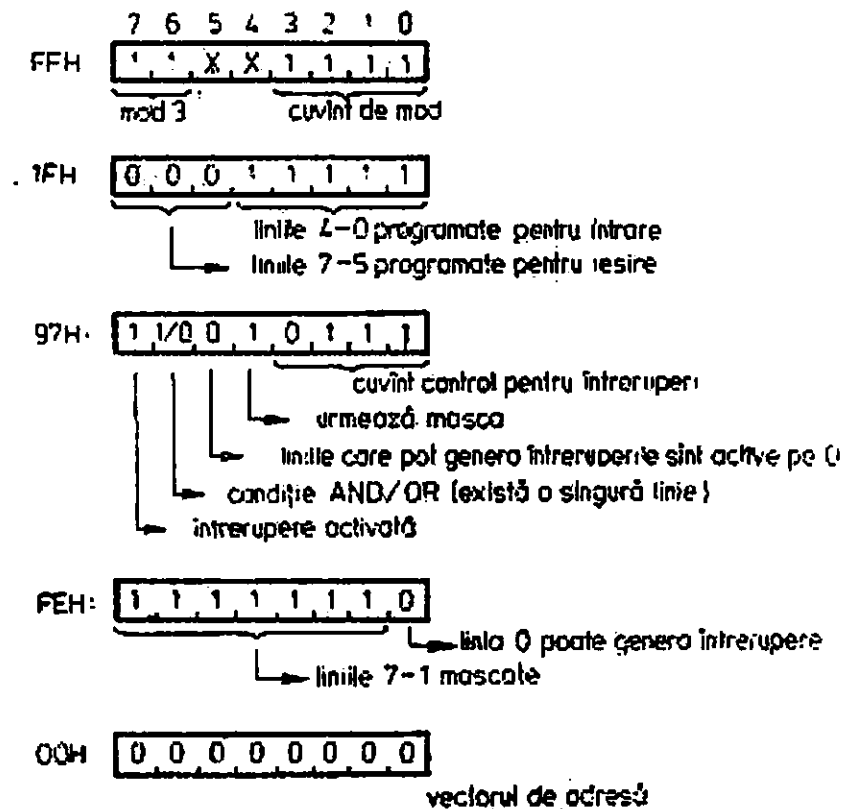


Fig. 8.17. Cuvintele de control pentru programarea circuitului Z80-PIO.

- bucla de simulare;
- subrutina de tratare a întreruperii generate de activarea semnalului RESET (tehnica de răspuns prezentată schematic în figura 8.18).

; Programarea circuitului 280-PIO

```

ORG 8000H      ;
LD  A, 0FFH    ; setează modul 3 (control bit)
OUT (61H), A   ; transmite în portul A control
LD  A, 1EH     ; liniile 7, 6, 5 programate pentru
OUT (61H), A   ; ieșire și liniile 4, 3, 2, 1, 0 pentru intrare
LD  A, 97H     ; cuvântul de control pentru întreruperi
OUT (61H), A   ;
LD  A, 0FEH    ; numai bitul 0 ( $\overline{\text{RESET}}$ ) poate genera
OUT (61H), A   ; întreruperi
LD  A, 0       ; vektorul de întreruperi
OUT (61H), A   ;

; Inițializarea sistemului de întreruperi la UCP
IM2      ; setează modul 2 de răspuns la întreruperi
LD  A, 66H    ; încarcă registrul I
LD  I, A
LD  HL, 7000H ; adresa subrutinei de tratare a întreruperii
LD  (6600H), HL ; memorează la adresa 6600H și 6601H

; Inițializare semnale de ieșire PIO
SIMUL: LD  A, 80H ; inițializare port A
      OUT (60H), A ;  $\overline{\text{WAIT}}=1$ ,  $\text{GO}=0$ ,  $\text{BUSY}=0$ 
      EI         ; activarea sistemului de întreruperi

; Bucla de simulare a interfeței
; Așteaptă selecția interfeței
BUCLA: IN  A, (60H) ; citește liniile portului A
      AND 1EH     ; selectează biții 4, 3, 2, 1
      XOR 4       ; testează semnalele A5,  $\overline{\text{WR}}$ ,  $\overline{\text{M1}}$ ,  $\overline{\text{IORQ}}$ 
      JR  NZ, BUCLA ; dacă cel puțin un bit nu este activ așteaptă

; Interfața a fost selectată,  $\text{ACC}=0$ 
IN  A, (60H) ; citește portul A
SET 5, A     ; poziționează  $\text{GO}=1$ 
OUT (60H), A ; inscrie în portul A

; Trecerea în 1 a semnalului GO declanșează monostabilul 74121
SET 6, A     ; poziționează  $\text{BUSY}=1$ 
OUT (60H), A ; inscrie în portul A

; Activarea lui BUSY, resetează semnalul GO
RES 5, A     ; poziționează  $\text{GO}=0$ 
OUT (60H), A ; inscrie în portul A

; Întrziere de 0,5 ms, timp necesar pentru terminarea operației de ieșire
LD  A, 3DH   ; constanta de întârziere pentru
DELAY: DEC A ; frecvența ceasului 2MHz
      JR  NZ, DELAY

; Lu străgă dezactivează BUSY și  $\overline{\text{WAIT}}$ 
IN  A, (60H) ; citește portul A
SET 7, A     ;  $\overline{\text{WAIT}}=1$ 
RES 6, A     ;  $\text{BUSY}=0$ 
OUT (60H), A ; inscrie în portul A

; Așteaptă terminarea accesului
STOP: IN  A, (60H) ; citește portul A
      AND 1EH     ; selectează biții 4, 3, 2, 1
      XOR 4       ; inversează M1 (semnale active pe 0)

```



```

    JH   Z, STOP   ; dacă ACC=0, așteaptă
    JH   LOOP     ; dacă ACC=1, rela secvența
; Subrutina de tratare a intreruperii generate
; de activarea semnalului RESET
    ORG  7000H
    EX   AF, AF'  ; salvare stare program
    EXX                                     ; numai dacă este necesar — în acest exemplu ; nu este necesar
    POP  HL      ; înlocuiește adresa de revenire din
    LD   HL, SIMUL ; stivă cu adresa de relansare a
    PUSH HL     ; programului de simulare
    EX   AF, AF' ; aceste două instrucțiuni nu sînt
    EXX                                     ; necesare — în general sînt utilizate pentru refacerea stării
                                             ; programului intrerupt
    RETI      ; return
    END

```

8.7. Cuplarea la microcalculator a unei miniiimprimante MIM40

Miniiimprimanta MIM40 (produsă la Electromureș Tg. Mureș) este o imprimantă paralelă cu 40 de caractere într-un rînd, fiecare coloană de caracter fiind formată din 7 puncte, fiecărui punct îi corespunde în sistemul electric de imprimare un ac de imprimare.

Formatul unui caracter este dat în figura 8.19 (7 linii \times 5 coloane).

Fiecare punct din matricea de imprimare este controlat prin software.

Din punct de vedere mecanic, transportul capului de imprimare se realizează cu un tambur dispunînd de un ghidaj elicoidal ca în fig. 8.20.

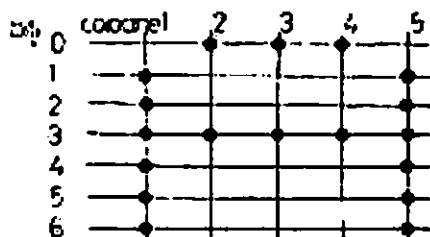


Fig. 8.19. Formatul unui caracter (exemplul A).

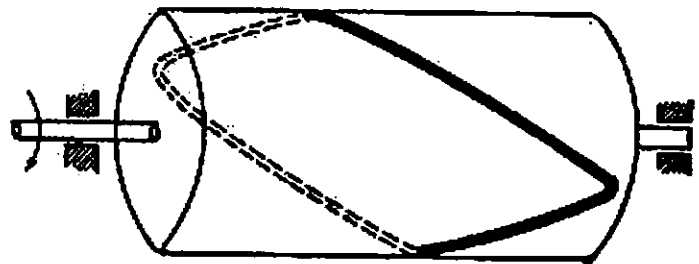


Fig. 8.20. Tambur pentru transportul capului de imprimantă.

În timpul cursei active a capului de imprimare se realizează imprimarea a 40 de caractere iar semnalul de stare a capătului de cursă activă și a cursei inverse este la nivel 1 logic.

La capătul cursei active acest semnal devine 0 logic și rămîne pe această valoare tot timpul cursei inverse. Semnalul de stare este generat printr-un sistem mecanic rigidizat cu tamburul.

Schema interfeței de cuplare a microcalculatorului aMIC cu miniiimprimanta este prezentată în figura 8.21.

Înainte de imprimarea unui caracter, din codul ASCII corespunzător se generează adresa corespunzătoare din generatorul de caractere.

Acesta conține, codificat pe cele 5 octeți, configurația tuturor caracterelor care pot fi imprimate, în ordinea codurilor ASCII.

După preluarea primului octet, care constituie prima coloană a caracterului, (având întotdeauna bitul 7 poziționat pe 1 logic: comanda motorului activă) acestu este transmis în interfața periferică programabilă și se declanșează o buclă de așteptare de 0,25 ms, timp în care solenoidii corespunzători sînt acționați executînd imprimarea coloanei.

La terminarea buclei de așteptare în portul de ieșire este înscris doar bitul de acționare a motorului poziționat pe 1 logic. Se generează o pauză de 1,25 ms necesară parcurgerii spațiului dintre două coloane, iar apoi se repetă algoritmul de imprimare coloană.

Între două caractere consecutive programul generează o pauză de 2,5 ms, timp în care se acționează în continuare motorul.

Avansul capului de imprimare se poate opri pe parcursul unei linii dar nu este posibilă întoarcerea la începutul cursei directe, deci prin imprimarea de blanuri în restul liniei. După terminarea imprimării celor 40 de caractere se generează bucla de așteptare a căderii pe zero a bitului de stare. După aceasta se generează o pauză de ordinul 130 ms acoperitoare pentru cursa inversă, iar apoi se oprește comanda motorului.

Durata pauzelor introduse în program sînt astfel alese încît imprimarea a 4 puncte alăturate să genereze un pătrat.

Micșorînd pauza între imprimarea a două coloane succesive se poate mări numărul de caractere imprimate pe o linie.

Spațiul între două linii succesive este generat automat prin forma ghidajului elicoidal de pe tambur, deci distanța între două linii imprimate este întotdeauna aceeași.

Prezentarea programului de imprimare

Înainte de lansarea subrutinei de imprimare a unei linii, programul de imprimare trebuie să asigure încărcarea unei zone tampon de date avînd lungimea de 40 de octeți, cîte un octet pentru fiecare cod ASCII care urmează să fie imprimat. După încărcarea zonei tampon, subrutina de imprimare poate fi executată iar pentru imprimarea unei pagini se repetă secvența de operații: încărcarea zonei tampon — execuție subrutină imprimare linie.

Subrutina de imprimare ia în considerare numai coduri cuprinse între 20H și 5FH pentru alte coduri imprimînd blanc. Generatorul de caractere conține cifrele zecimale, literele majuscule de la A la Z și următoarele caractere speciale: blanc, ?, " , ' , \$, ; , () , * , + , ' , - , , / , ; , = , , ?.

Subrutina de afișare a unei linii este prezentată în cele ce urmează :

```

IMPRIM :   PUSH H
           LD A, 81H
           OUT 43H      ; programarea interfeței 8255
           XOR A
           OUT 42H
           LD HL, BUFTIP ; HL-adresa zonei tampon
           LD B, 28H     ; 40 caractere pe linie
  
```

TIP 2: IN A, 42H
AND 01
LD A, 80H ; se comandă motorul
OUT 40H A
JP Z, TIP1 ; se așteaptă pînă ce bitul de stare
JP TIP2 ; devine „0”

TIP 1: LD DE, CONT4 ; pauză de început linie
CALL PAUZA

TIP 3: IN A, 42H
AND 01 ; se așteaptă pînă ce bitul de stare devine „1”
JP Z, TIP3

TIP 7: LD A, (HL)
CP 20H
JP C, TIP4
CP 00H
JP C, TIP5 ; Codurile ASCII care nu aparțin
[20, 5F] se înlocuiesc cu 20H.

TIP 4: LD A, 20H

TIP 5: INC HL.
PUSH HL. ; Salvează adresa caracterului următor
LD HL, TAB-0A0F
LD B, 00
LD E, A
ADD HL, DE
ADD HL, DE
ADD HL, DE ; Calculul adresei din generatorul de caractere a codului
ADD HL, DE ; din Acc.
ADD HL, DE ; Nr. de coloane într-un caracter
LD C, 05H ; Se încearcă 8255 cu o coloană de imprimare

TIP 6: LD A, (HL)
OUT 40H A
LD DE, CONT1 ; durata de acționare a solenoidilor
CALL PAUZA
LD A, 80H
OUT 40H A ; comandă motor
LD DE, CONT2 ; pauză între 2 coloane succesive
CALL PAUZA
INC HL
DEC C
JP NZ, TIP6 ; pauză între 2 caractere
LD DE, CONT3
CALL PAUZA
POP HL
DEC B ; încă mai sînt caractere de imprimat
JP NZ, TIP7

TIP 8: IN A, 42H
AND 01 ; se așteaptă ca bitul de stare să devină „0”
JP NZ, TIP8
LD DE, CONT5 ; pauză necesară curselor inverse
CALL PAUZA
XOR A ; oprire motor.
OUT 40H A
POP H
RET

PAUZA: DEC DE
LD A, E
OR
JP NZ, PAUZA
RET.

Configurația semnalelor la conectorul de interfață cu miniimprimanta

1. PA7	bufferat	9. PB7	17. PC7
2. PA6	bufferat	10. PB6	18. PC6
3. PA5	bufferat	11. PB5	19. PC5
4. PA4	bufferat	12. PB4	20. PC4
5. PA3	bufferat	13. PB3	21. PC3
6. PA2	bufferat	14. PB2	22. PC2
7. PA1	bufferat	15. PB1	23. PC1
8. PA0	bufferat	16. PB0	24. PC0
			25. GND.

8.8. Cuplarea microcalculatorului cu un programator de EPROM

Programatorul asigură posibilități de programare pentru circuitele EPROM de tipul Intel 2716.

Configurația circuitului este următoarea (fig. 8.22).

A7	1	24	Vcc
A6	2	23	A8
A5	3	22	A9
A4	4	21	Vpp
A3	5	20	\overline{OE}
A2	6	19	A10
A1	7	18	\overline{CE}
A0	8	17	O7
D0	9	16	D6
D1	10	15	D5
D2	11	14	D4
GND	12	13	D3

Fig. 8.22. Configurația terminalelor circuitului 2716.

Semnificația semnalelor :

D0 - D7 = date intrare/ieșire

A0 - A10 = adrese

Vpp = selecție regim de programare

\overline{OE} = validare ieșire

\overline{CE} = intrare selecție circuit.

Modurile de lucru ale circuitului sunt prezentate în figura 8.23 a, unde se pun în evidență condițiile în care se realizează fiecare regim : citire, programare, verificare.

Regimul de programare, verificare este selectat forțând la intrarea Vpp o tensiune de 25 V.

Datele de intrare trebuie să fie stabile înainte și după pulsul de programare, minimum 2 μ s. Pulsul de programare durează între 45 și 55 μ s și este livrat circuitului la intrarea \overline{CE} de nivel TTL. (fig. 8.23 b).

PIN MOD	\overline{CE}/PGM (18)	\overline{OE} (20)	V_{pp} (21)	V_{CC} (24)	OUTPUTS 19-11,13-17)
READ	V_{IL}	V_{IL}	-5	+5	D_{OUT}
STANDBY	V_{IH}	NU CONT	+5	+5	HIGH Z
PROGRAM.	PULS V_{IH} V_{IH}	V_{IH}	+25	+5	D_{IN}
PROGR. VERIFY	V_{IL}	V_{IL}	+25	+5	D_{OUT}
PROGR. INHIB	V_{IL}	V_{IH}	+25	+5	HIGH Z

c

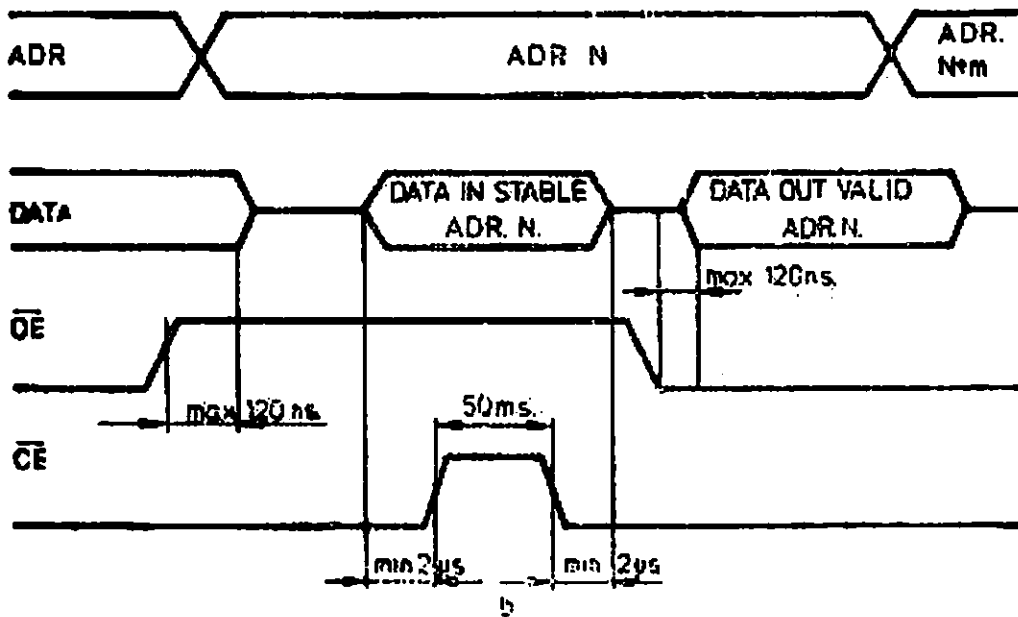


Fig. 8.23. Memoria EPROM 2716. Regimurile de lucru (a), diagramele de timp (b).

În figura 8.24 este prezentată schema bloc de interfațare a programatorului la microcalculator.

Interfațarea este realizată utilizându-se adresarea diferitelor operații ca porturi. Adică datele, adresele cit și comenzile sînt livrate programatorului prin interfața periferică programabilă, fiind memorate în registre cu încărcare paralelă (CDB495).

Descrierea funcționării programatorului (fig. 8.25)

Dispozitivul este cuplat la magistrala externă de date a microcalculatorului, necesitînd semnale de adresă (AB3, AB4) și semnale de comandă (RDB, IOREQ).

Semnalele de adrese și comenzi sînt folosite pentru decodarea celor 6 selecții necesare funcționării programatorului. Transferul de date din/înspre

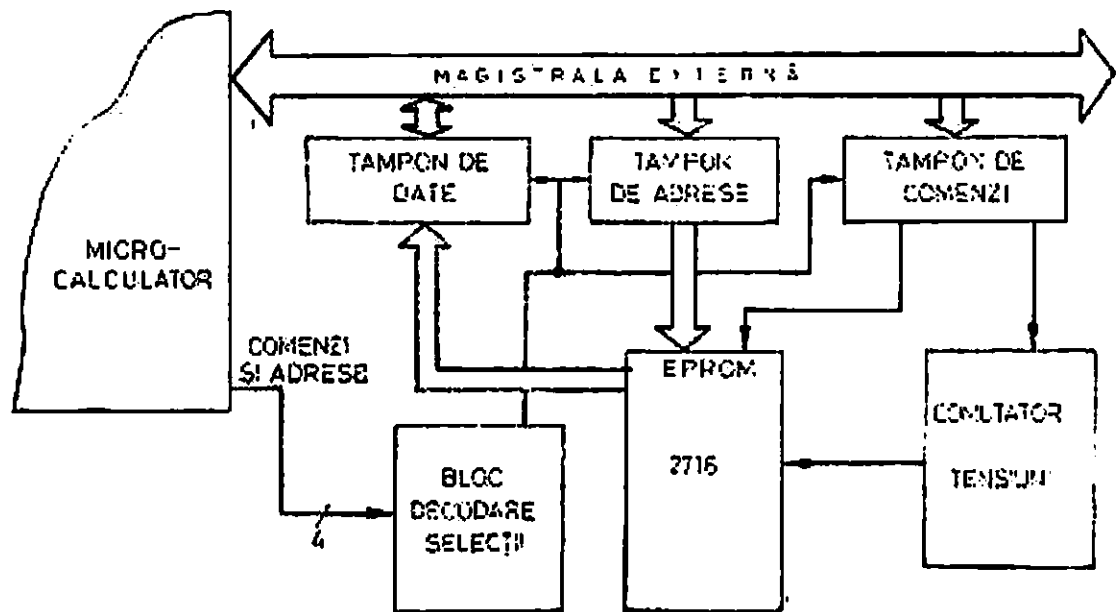


Fig. 8.24. Interfața microcalculator-programator EPROM.

programator se realizează prin instrucțiuni de intrare/ieșire. Codurile porturilor I/O sînt astfel construite încît selectează magistrala externă de date prin cuartetul superior și porturile din programator prin cuartetul inferior. Adresele sînt prezentate programatorului octet cu octet fiind încărcate în registrele tampon de adrese iar datele sînt înscrise în registrul tampon de date sau citite din EPROM prin bufferele de separare de tipul 7409.

Pentru protejarea memoriei EPROM ieșirile registrului tampon de date au fost bufferate cu circuite „trei stări” de tipul 74125.

Pe pinul de alimentare V_{cc} al circuitului EPROM se aplică tensiune de $+5\text{ V}$ programată. În timpul introducerii respectiv scoaterii circuitului 2716 din soclu tensiunea de alimentare nu ajunge la circuit.

Deschiderea liniilor de ieșire din tamponul de date spre circuitul EPROM se comandă prin înscriserea unui bit corespunzător într-un registru de comenzi.

Intrucît, în funcție de regimul de lucru, intrarea V_{pp} a circuitului 2716 este forțată pe 5 V (în regim de citire) respectiv 25 V (în regim de programare/verificare) s-a prevăzut un circuit de comutare nivele de tensiune comandat tot printr-un bit înscris în registrul de comenzi.

În timpul operării cu memoria EPROM s-a prevăzut posibilitatea de preluare a stării tensiunilor V_{pp} , respectiv V_{cc} , prin citirea pe magistrala de date a unui cuvînt de stare. În funcție de nivelul tensiunii $V_{pp} \leq 24\text{ V}$ bitul 0 din cuvîntul de stare este poziționat pe 1 sau zero logic.

Bitul 1, din cuvîntul de stare este poziționat pe 1 logic dacă EPROM-ul nu este alimentat cu tensiunea $+5\text{ V}$.

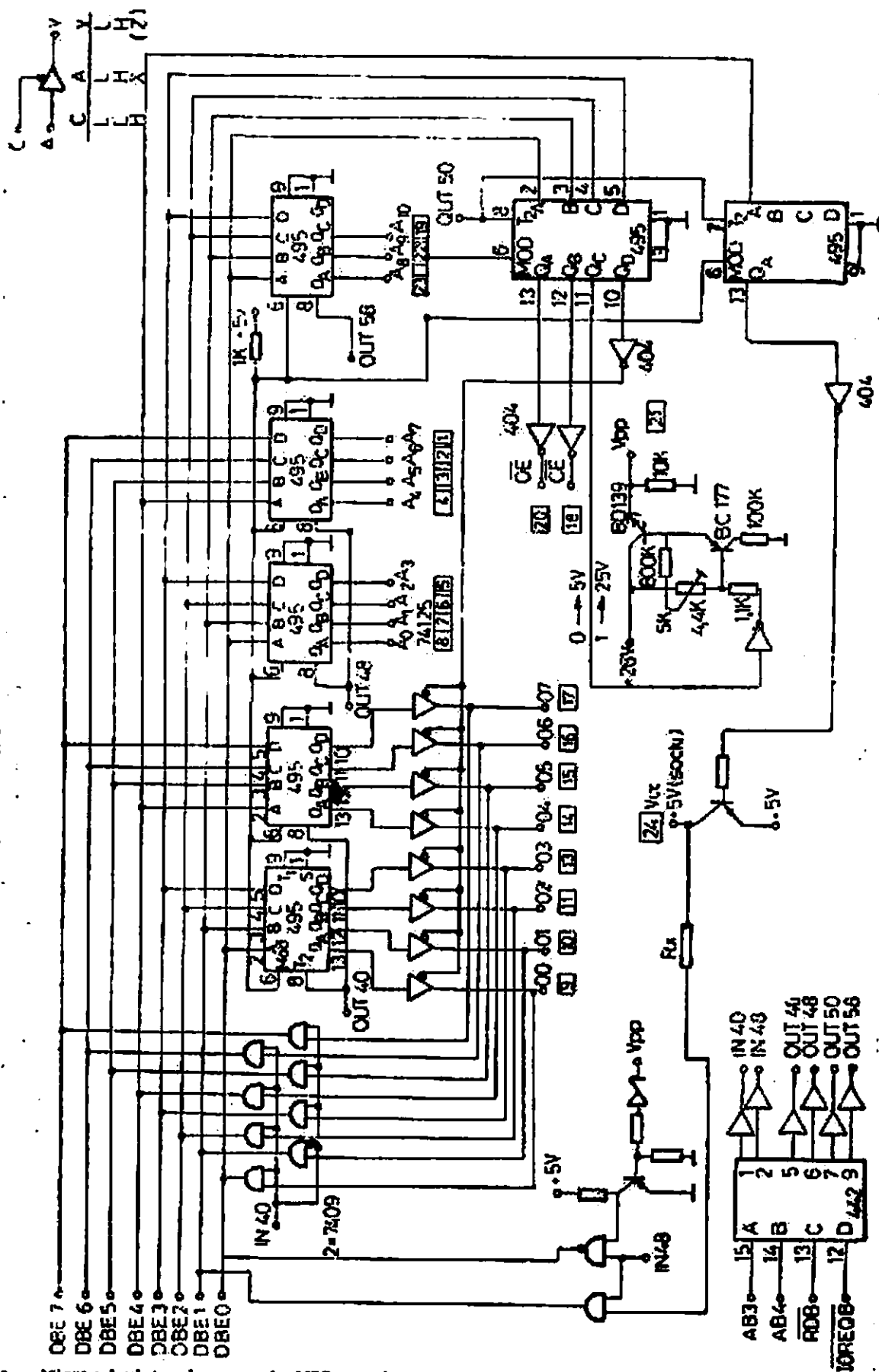


Fig. 9.25. Programator de memorii EPROM 2716.

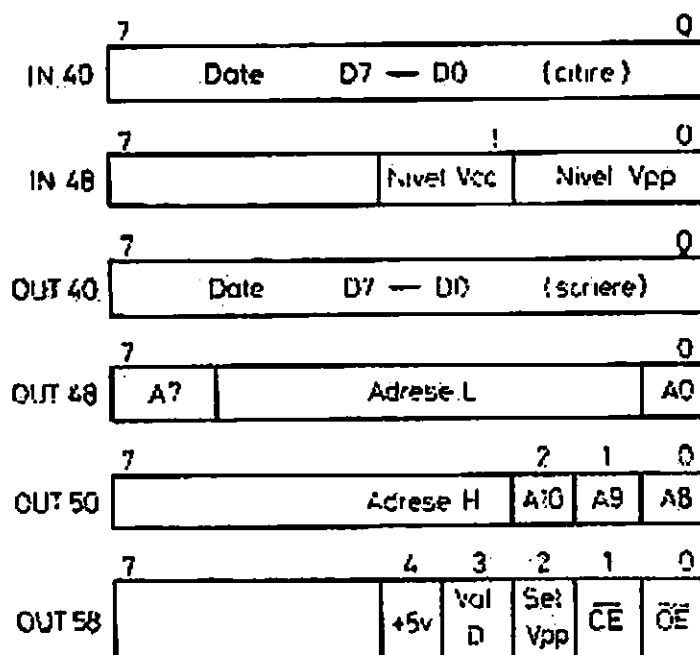


Fig. 8.26. Instrucțiuni de I/E.

Semnificația informației transferate în urma apelării unui port. reiese din figura 8-26.

Organigrama programului de scriere/verificare a EPROM-ului

Deoarece programatorul nu este prevăzut cu posibilități de RESET-are, la cuplarea tensiunii este nevoie ca programul să fie lansat înainte de a introduce cipul de memorie în soclu, pentru a poziționa corect registrul de comenzi (o configurație întâmplătoare a acestuia ar putea pune în pericol cipul de memorie EPROM). După lansare, programul execută inițializările necesare și afișarea mesajului :

```
PROGRAMATOR_2716
COMPL (D, N)
```

În funcție de răspunsul operatorului D sau N, informația din EPROM se consideră complementată sau nu. În continuare se afișează prompterul (*) și se așteaptă comenzi. Programul acceptă un număr de 7 comenzi, ale căror acțiuni vor fi descrise în continuare.

Comanda A (alimentări) verifică prezența tensiunilor de +5 V și +25 V la pinii 24 și respectiv 21 ai soclului în care se amplasează EPROM-ul. Comanda constituie un test pentru funcționarea comutatoarelor de tensiune, dînd diferite comenzi și prelînd cuvîntul de stare. Testul se execută de preferință cu soclul liber.

Comanda V (verificare) operează cu memoria EPROM în regim citire, verificînd dacă toate locațiile memoriei sînt FF. Dacă există erori de „ștergere“ ale memoriei, se va afișa adresa și octetul corespunzător și se așteaptă decizia

operatorului de a continua verificarea, cu (CR) sau de a reveni la prompter, cu orice altă tastă. La sfârșitul verificării se va afișa numărul total de erori de ștergere.

Comanda T (transfer) execută transferul zonei de memorie EPROM cuprinsă între ADR. ÎNCEPUT SURSA ȘI ADR. SFÎRȘIT SURSA în memoria RAM de la ADR. ÎNCEPUT DESTINAȚIE, complementat sau nu.

Comanda C (Comparare) realizează compararea unei zone din EPROM cu o zonă din memoria RAM. Adresele limită ale blocurilor se dau după lansarea comenzii.

Fiecărei erori depistate îi corespunde un mesaj care cuprinde tipul erorii (corectabilă sau necorectabilă, adică „1” în loc de „0” și invers), adresa, octetul din RAM și cel din EPROM. Se continuă compararea cu (CR). După terminarea comparării se afișează numărul total de erori apărute și numărul de erori necorectabile.

— format : A ADR (CR)

Comanda D (Display) permite afișarea a 8 octeți din memoria EPROM începând de la adresa menționată în comandă, adresă care se afișează la începutul rândului.

Tastarea (CR) are ca efect afișarea următorilor 8 octeți.

Comanda D (Programare) realizează cea mai importantă acțiune, aceea de înscriere a unei zone din memoria RAM în memoria EPROM. Adresele limită a blocurilor au în cazul acestei comenzi următoarea semnificație : adresele referitoare la sursă delimitază zona din EPROM care va fi înscrisă respectându-se convenția care definește memoria EPROM ca sursă iar cea RAM ca destinație. Înainte de programarea propriu-zisă se realizează compararea zonei sursă cu zona destinație. Dacă sînt depistate erori necorectabile ele sînt afișate și se așteaptă decizia operatorului. Acesta poate să le ignore și să continue compararea cu (CR). Acest test care anticipă programarea, dă posibilitatea înscrierii unei memorii EPROM care deși nu este ștersă complet, nu are diferențe necorectabile față de informația care urmează să fie înscrisă.

Înscrierea fiecărui octet este anticipată de verificarea tensiunii de 25 de volți care poate fi mai mică de 24 de volți datorită sursei, comutatorului 5—25 V sau datorită intrării Vpp defecte a cipului de EPROM. Succesiunea comenzilor și inițierile necesare pentru înscrierea corectă a unui octet sînt generate prin software.

Se verifică octetul înscris, comparîndu-se cu octetul corespunzător din RAM și dacă apare eroarea corectabilă se repetă înscrierea aceleiași locații. Dacă eroarea corectabilă se menține, ea va fi contorizată și se continuă programarea următoarelor locații. Dacă apare o eroare necorectabilă ea este afișată și se așteaptă decizia operatorului. După terminarea înscrierii întregului bloc de date se afișează numărul total de erori și numărul de erori necorectabile apărute.

Comanda S permite înscrierea unui singur octet din EPROM. Comanda cere adresa și octetul de înscriere după care îl compară cu octetul existent în EPROM. Dacă diferența este necorectabilă ea este afișată și se cere decizia operatorului. Revenirea în monitorul microcalculatorului se face cu comanda (CR).

8.9. Cuplarea cu un terminal DAF 2010

Prin intermediul interfeței seriale 8251 se realizează transferul bidirecțional asincron de date între microcalculator și un sistem DAF 2010, cu o rată de transfer de 2400 Bauds.

Informația vehiculată poate fi transferată octet cu octet, sau sub formă, de blocuri de date, informat HEX-INTELLEC, care presupune următoarea structură a înregistrării :

CB LF : NO ADR T Date Control

unde : NO este numărul de octeți de date ai înregistrării,
ADR — adresa de început a înregistrării,
T — tipul înregistrării,
Date — reprezintă maxim 16 octeți
Control este suma logică a octeților din înregistrare.

Sfârșitul de fișier este constituit de o înregistrare cu număr de octeți zero iar adresa corespunzătoare va fi adresa de salt în fișierul încărcat.

Programul de transmisie a unui fișier, la lansare așteaptă adresa de început și sfârșitul a blocului de date iar la sfârșitul transmisiei formează automat înregistrarea de sfârșit fișier.

Programul de recepție a unui fișier, așteaptă după lansare, adresa de deplasare a blocului de date față de adresa de început a fișierului (intrinsecă).

Secvența de programare a interfeței serie 8251 este prezentată în continuare :

MVI A, 0EAH
OUT 01H

MVI A, 15H
OUT 01H

Semnificația este următoarea : 2 biți de stop, control de paritate dezactivat caracter pe 7 biți, viteza de transmisie = rata de transmisie X16, activare transmisie și recepție, anularea erorilor din registrul de stare.

Subrutina de transmisie a unui caracter aflat în memorie la adresa HL este prezentată în continuare :

TCAR : MOV A, M
OUT 00H

TCI : IN 01H
ANI 01H
JZ TCI
RET

Recepția unui octet de la interfața serială este realizată de următoarea subrutină :

RCAR : IN 01H
ANI 02H
JZ RCAR

IN 00H
ANI 7FH
RET

Programul de transmisie fișier preia din memorie blocuri de 16 octeți, ornamează informația în HEX-INTELLEC și calculează suma de control realizând astfel o înregistrare. Acest ciclu este repetat până la terminarea blocului de date, completându-se fișierul cu înregistrarea nulă.

Programul de recepție fișier transferă în memorie înregistrări succesive, prelucrând secvențial fiecare octet și verificând la sfârșit suma de control.

În cazul apariției unei erori programul afișează mesajul: ER: adresă, unde adresa constituie începutul înregistrării eronate.

În momentul recepției unei înregistrări nule se face ieșirea în monitorul microcalculatorului.

În continuare se dă configurația semnalelor la conectorul de interfață cu un DAF 2010:

1. GND	4.	8. STRAP pt. DAF 9÷10 neutilizat
2. TXD	5. STRAP pt. DAF	7. GND
3. RXD		8. —
		20. —

8.10. Interfața cu un minirobot

Minirobotul de laborator M2,5 elaborat la „Electroliniș” Timișoara este un robot experimental cu aplicabilitate în industria electronică pentru manipularea de piese și componente de mici dimensiuni și greutate redusă (2,5 Nm; greutate piese 300 g, deschiderea dispozitivului de prehensiune: 7 cm;). Minirobotul dispune de 5 grade de libertate fiind prevăzut cu 5 motoare pas cu pas (20 W) acționate în buclă deschisă.

Minirobotul este realizat constructiv cu elemente mecanice din material plastic, transmisia este asigurată prin fire, posedă un panou de învățare și o tastatură cu electronică de urmărire.

Mișcările pe care le poate executa acesta sînt:

- pivotarea (rotația în jurul axului principal)
- mișcările brațului și antebrațului
- mișcările de flexie și supinație a mîinii mecanice.

Minirobotul este interfațat la microcalculator printr-o interfață periferică programabilă, avînd cele 3 porturi programate ca ieșiri.

Prin această interfață microcalculatorul asigură 8 comenzi de sens de mișcare și 8 comenzi de tact repartizate fiecărui motor pas cu pas. (1 comandă de sens mișcare și 1 comandă de tact la fiecare motor).

Comenzile microcalculatorului sînt prelucrate într-o interfață de acționare pentru motoarele pas cu pas.

Viteza de execuție a mișcărilor este controlată software, interfața de acționare intervenind la depășirea valorilor prescrise pentru semnalele de tact.

Controlul mișcării accelerate și decelerate este asigurat tot din interfața de acționare.

Programul de aplicație este scris într-un limbaj de nivel înalt, ARMBASIC, cu subrutine în limbaj mașină Z80. Între posibilitățile software remarcăm posibilitățile de elaborare a unor modele matematice de transformări de coordonate, un formalism matricial, posibilități de comandă în coordonate carteziane etc. Comenzile specifice ARMBASIC au fost utilizate și la simularea mișcărilor unui minirobot în cadrul a două programe de aplicație: un program

de învățare pas cu pas a execuției unei mișcări și un program de execuție a unei traiectorii impuse. Programul de învățare permite simularea acționării pas cu pas a motoarelor; aducând minirobotul în poziții succesive, la comenzi de operator.

Cel de-al doilea program realizează simularea mișcării mâinii minirobotului pe o traiectorie a cărei coordonate succesive sînt calculate secvențial de către microcalculator. Simularea minirobotului pe microcalculatorul aMIC este reflectată prin afișarea stilizată pe display a pozițiilor succesive ale minirobotului.

8.11. Echipament de testare pentru micro sisteme orientate pe magistrală

Structura hardware a microcalculatorului „aMIC” constituie procesorul „master” al unui sistem biprocesor de test în curs de elaborare la ITC.

Sistemul de test asigură posibilități de fabricație, service și instalare de aplicații a micro sistemelor cu circuite LSI organizate pe principiul magistralei.

Echipamentul realizează o testare funcțională dinamică pe principiul emulării.

Stimularea plachetelor sau micro sistemelor sub test este asigurată de emulator iar evaluarea răspunsurilor se face prin sondă mobilă cu analiză de semnături.

Resursele sistemului LSMINITEST sînt: memorie utilizator în procesorul master 48 Ko, memorie de emulare 8 Ko, sistem de operare și interpretor de test rezidente în EPROM, procesor master (16 Ko), monitor de depanare/testare (rezident sau încărcat în memoria externă) programe de test și module de test funcțional încărcate din memoria externă, program de conduc sonda mobilă, monitor de emulare (2 Ko).

Memoria externă este realizată cu minifloppy sau casetă audio. Sistemul de test are ca display un receptor TV obișnuit, dispune de miniprintantă MIN40, are disponibilități de transmisie/recepție serială de informații, putînd constitui un post de lucru al unei echipament de test complex, multipost, LSI TEST.

Sistemul de testare are posibilități de punere la punct și de editare a programelor de test, avînd interfațat un programator de memorii EPROM.

Emulatorul dispune de următoarele facilități:

Stimularea în timp real a plachetelor sau micro sistemelor sub test cu trei categorii de stimuli: primari, module de test funcționale și stimuli defavorabili, memorie de emulare relocabilă în spațiul de adresă al procesorului emulator, ceas de gardă, generare și tratare de întreruperi, memorie trasoare, emulare pe micro sisteme cu microprocesor Z80 sau 8080.

POD-ul emulatorului este constituit din microprocesorul 8080 sau Z80, bufferele de interfață și controlerul acestora. Intervenția pe placa sau micro-sistemul sub test se face la nivelul DIP-ului pe soclul microprocesorului emulat. Un translator de bus asigură posibilitatea testării micro sistemelor organizate pe diferite standarde de magistrală. El este compus dintr-un modul

de unitate centrală specific, un modul de memorie specific și o placă adaptoare de bus. Sonda mobilă asigură facilități de ridicare a unei semnături, discriminează nivele TTL, MOS, analogice, lucrează sub comanda procesorului master și emulatorului.

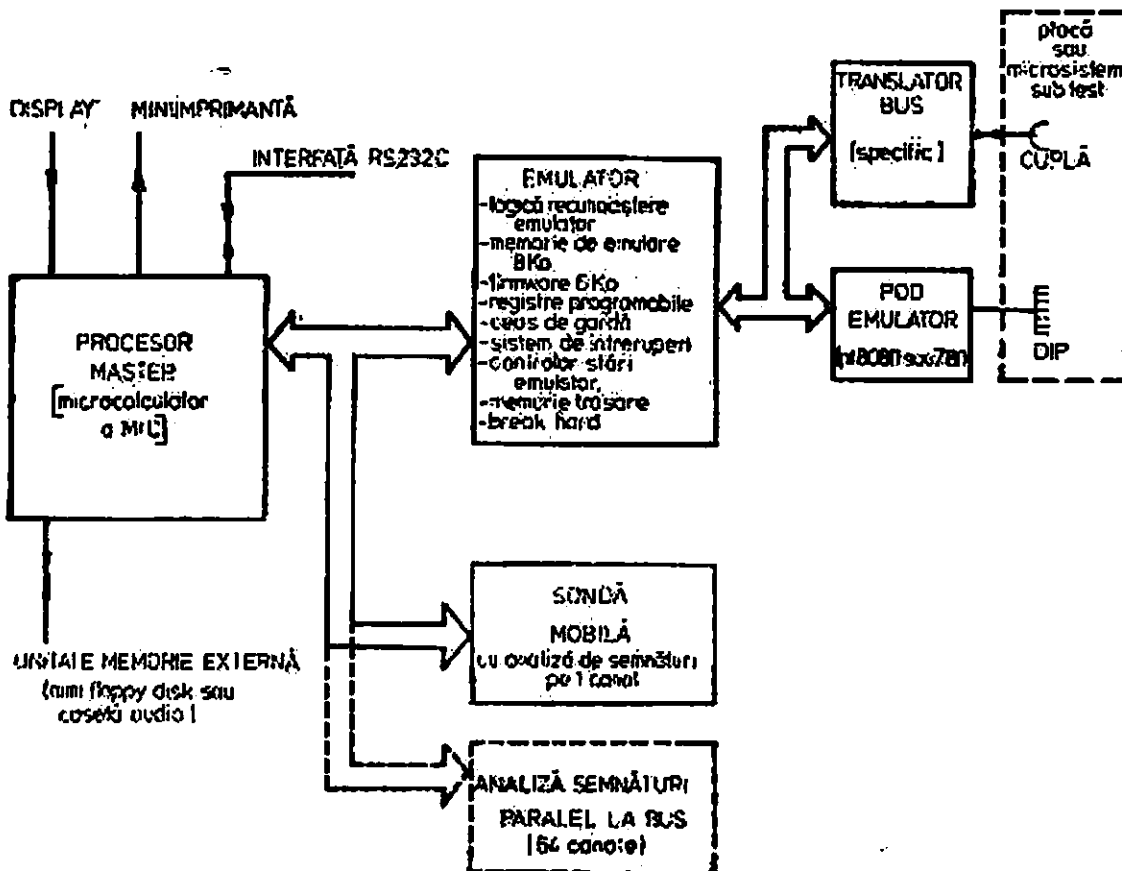


Fig. 8.27. Schema bloc a echipamentului de test LSIMINITEST.

Sonda este ghidată prin program pentru localizarea defectului pe placa sau micro sistemul sub test.

Schema bloc a echipamentului de test LSIMINITEST se prezintă în figura 8.27.

8.12. aMIC-ul în unități de deservire pentru mașini-unelte

În cadrul unei lucrări în curs de elaborare la ITC în colaborare cu o întreprindere constructoare de mașini-unelte, microcalculatorul aMIC se constituie ca o unitate de deservire a mașinilor-unelte, preluând următoarele funcțiuni:

- editare de bandă perforată;
- programare EPROM;

- miniconsolă pentru automate programabile;
- imprimare programic.

Editor de bandă perforată

Resursele hardware ale microcalculatorului sînt completate cu interfațarea unui lector (respectiv perforator) de bandă perforată. Editarea de bandă perforată presupune existența unui program de citire/scriere a benzilor, cum și un program de prelucrare a textelor în limbajul specific mașinilor-unelte.

Intrucît acest limbaj nu este încă standardizat este necesară o mare flexibilitate a programului, astfel încît să permită apelarea, modificarea, înscrierea (respectiv ștergerea) unor instrucțiuni sau linii de program, cu relocarea corespunzătoare a textului.

Programul permite și copierea benzilor perforate, prin controlul numărului de linie sau a unei anumite instrucțiuni.

Programator memorie EPROM

Dispozitivul reprezintă o variantă cu posibilități largite de programare, putînd programa circuite 2708 sau 2716. Informația se încarcă într-o zonă RAM tampon în mai multe moduri:

- prin citire de pe bandă perforată
- prin citirea unei memorii EPROM
- prin introducere manuală
- de la miniconsola pentru automat programabil.

Microconsola pentru automate programabile (AP101, AP117). Sistemul poate fi cuplat la automatul programabil, asigurîndu-i o zonă de memorie RAM în care se încarcă programul de control al procesului.

Prin software se poate prelucra ușor informația înscrisă în memoria RAM permițînd căutarea, ștergerea, modificarea, înscrierea unei instrucțiuni specifice automatului programabil. După punerea la punct a programului, acesta este înscris în memoria EPROM, constituindu-se ca program rezident pentru automatul programabil.

Sistemul permite, de asemenea, printr-un program adecvat, simularea execuției programului, fără cuplare la proces, variabilele de intrare putînd fi controlate și modificate la intervenția operatorului, iar variabilele intermediare și cele de ieșire putînd fi urmărite pe display.

Se poate testa, astfel, urmărirea și controlul procesului (în condiții defavorabile) și se poate optimiza programul de conducere a procesului.

În cazul simulării variabilelor de intrare în condițiile unui proces în buclă închisă, se verifică optimalitatea programului de control al procesului înainte de realizarea aplicației, sistemul preluînd funcții de sistem expert.

Interfațarea miniimprimantei MIM40

În cadrul sistemului de deservire a mașinilor-unelte, funcția de imprimare este utilă pentru a asigura un suport de ieșire al programelor în procesul prelucrării și modificării acestora.

Soluția de interfațare a miniimprimantei la sistem a fost prezentată în paragraful 8.7.

8.13. Sistem de înregistrare/redare a parametrilor semicontinui de proces

Acest sistem este destinat, într-o primă variantă, achiziției parametrilor de sudare cu arc electric (tensiunea arcului U_a , curentul de sudare I_a , sub forma unei mărimi nominalizate în domeniul 0–10 V și viteza trenului de sudare V_a , sub forma unui tren de impulsuri de maximum 5 kHz) cu rezoluția de 10 biți și timpul de eșantionare de 200 ms. Parametrii achiziționați pot fi memorati pe un suport extern (casetă magnetică audio sau (opțional) bandă perforată) și tipăriți off-line pe o miniimprimantă paralelă. De asemenea, conținutul unei înregistrări din proces poate fi redat sub forma unor mărimi analogice de comandă a procesului. Numărul de canale este maximum 16.

Configurația completă are următoarele componente:

- modul unitate centrală (structurat pe varianta microcalculator personal oMIC) prevăzut cu 2 interfețe paralele (una funcțională pentru lector/perforator de bandă și una de intrare pentru tastatură), interfață serie RS232C (destinată comunicației cu DAF2010 sau cu alt sistem de calcul), memorie RAM dinamică (48 Ko) și EPROM (16 Ko), programator on line pentru circuite EPROM tip 2716, 2732;

- modul interfață universală, care realizează interfețe cu modulul de afișare (6 digiți), cu imprimanta tip MIM40 (40 caractere); prin generator de caractere pe coloane rezident într-un ROM, cu unitatea de casetă (8251), cum și un ceas programabil de timp real pentru prescrierea frecvenței de eșantionare până la 100 Hz;

- setlar de interfețe specifice cu procesul, care preiau semnalul de la transducătorii de tensiune, curent, temperatură și-l încadrează în urma modulării, optoizolării și demodulării în domeniul 0–10 V cu eroare mai mică de 1%;

- periferice înglobate: modul afișare 6 digiți, imprimantă MIM40, (inclusiv modul de secționare) unitate de casetă audio, tastatură caracter, 10 cifre și funcțională;

- conectori de interfață cu lector LB50/perforator PE50, cu unitate de casetă UCM101, pentru parametrii proces.

Sistemul lucrează în regim de întrerupere, mod IM1, mascabile prin soft. Ele se înregistrează într-un registru de 8 biți, care este citit și decodificat prin program în urma generării unui semnal unic de întrerupere.

În regim de programare, operatorul poate prescrie prin dialog:

- numărul canalului;
- tipul canalului;
- tipul de eșantionare (100 Hz);
- funcția înregistrată

În figura 8.28 se prezintă schema bloc a sistemului de înregistrare/redare a parametrilor de proces.

Software-ul aplicației asigură urmărirea procesului în timp real, fiind realizat în limbaj mașină 8080 (sau 280).

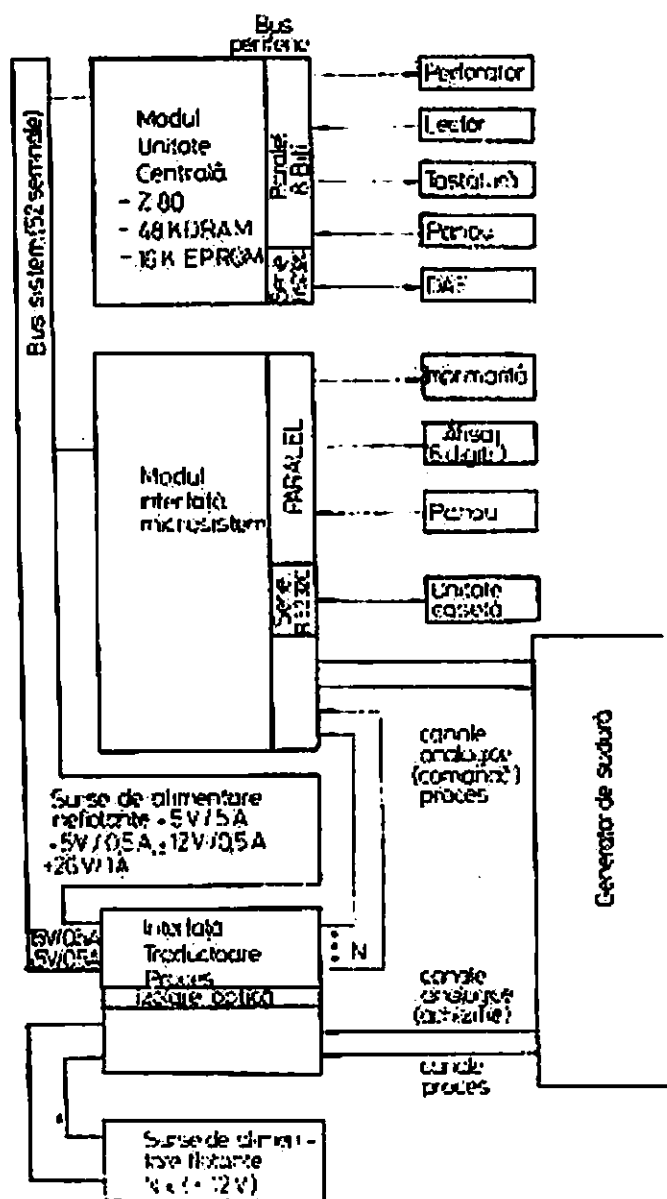


Fig. 8.20. Schemă bloc a sistemului de înregistrare/redare a parametrilor de proces.

8.14. Microcalculator (de laborator) pentru prelucrarea datelor provenite din analiza cromatografică

Pentru a evita costul ridicat al unei implementări specifice și pentru a putea destina sistemul și altor proceduri de analiză fizico-chimică, soluția aleasă la ITC Timișoara recurge la o variantă cu un anumit grad de modularitate, funcția principală de sistem fiind preluată de microcalculatorul **AMIC**, cu resurse proprii de memorie 64 Ko (48 Ko RAM și 16 Ko EPROM).

Acest modul interfațează tastatura de caractere și funcțională, un DAF sau monitor TV ca echipament extern de vizualizare și asigură o interfață de date paralelă.

Restul periferiei este tratat pe modulul de interfață care gestionează imprimanta paralelă de tip MIM10, înglobată în echipament, subsistemul de achiziție a mărimii de analiză, un ceas real de 20 ms și ceasul de eșantionare.

Pe același modul sunt rezidente și registrul de întreruperi și măștile de întreruperi.

Conversația numeric-analogică de tip interactiv se realizează pe 15 biți, plus un bit de semn, în domeniul $-100 \text{ mV} - +1 \text{ V}$, avînd perioada de eșantionare sincronizată cu frecvența rețelei, pentru rejecția zgomotelor de înaltă frecvență.

Semnalul este preluat diferențial pentru a reduce zgomotele induse prin masă.

Semnalul de analiză provenit de la cromatograf se prezintă sub forma unei succesiuni de vîrfuri (peak-uri).

Fiecare „peak” apare la un moment de timp care trebuie corect estimat, întrucît identifică tipul unui anumit component din amestecul de analiză.

Integrala „peak”-ului dă o informație cantitativă asupra componentului detectat, astfel încît, în final, se poate face o estimare exactă a compoziției procentuale a probei analizate.

Software-ul sistemului este scris în limbaj de asamblare 8080 (16 Ko program și 12 Ko zonă de date, permițînd integrarea a 350 peak-uri).

Pachetul de programe este organizat modular și cuprinde un monitor cu facilități de autodepanare, task-uri de dialog, de integrare, de corecție, de calcul al raportului (precizia de calcul: 24 biți).

Limbajul BASIC

pentru microcalculatorul personal aMIC

Manual practic

9.1. Introducere

BASIC (Beginners All-purpose Symbolic Instruction Code) reprezintă un limbaj conversațional de nivel înalt. Pentru execuția programelor de aplicație scrise în BASIC se folosesc cel puțin trei metode distincte: interpretarea, compilarea și combinarea lor. Pachetul software, care folosește metoda combinată, poartă numele de compilator/interpretor sau pseudo-compilator.

Fiecare din metodele de execuție a programelor de aplicații are avantajele și dezavantajele ei. În general, alegerea metodei este determinată de aplicație, de utilizarea particulară a calculatorului.

În timpul execuției, un interpretor analizează fiecare instrucțiune, verifică existența erorilor, apoi efectuează funcția BASIC solicitată. Pentru a executa funcția respectivă, interpretorul traduce instrucțiunea dată în codul ASCII într-un număr de instrucțiuni mașină executabile, pe care le execută în continuare. Acest proces se repetă la fiecare instrucțiune BASIC, chiar dacă instrucțiunea respectivă apare de mai multe ori în cadrul unui ciclu.

Modul interpretativ de execuție presupune existența în memorie calculatorului a întregului program BASIC-sursă, sub forma unei liste de linii marcate cu etichete numerice. Fiecare instrucțiune de ramificație impune ca interpretorul să caute într-o listă de numere de linii, pentru a găsi instrucțiunea dorită. În aceeași manieră interpretorul va căuta o anumită variabilă în cadrul unei liste date.

Metoda bazată pe compilare presupune traducerea programului sursă, scris în BASIC, într-un fișier obiect relocabil. În continuare, un program de sistem, denumit Editor de legături, va înălța toate fișierele obiect relocabile produse, cât și fișiere obiect extrase dintr-o bibliotecă, pentru a produce fișierul executabil în cod mașină. Programul obiect compilat, va fi executat de 25÷30 de ori mai repede decât se execută în maniera interpretativă același program BASIC sursă.

Compilarea și editarea de legături trebuie repetată la fiecare modificare executată în programul sursă. În cazul interpretorului se va modifica cu ajutorul unor facilități de editare încorporate, linia respectivă din programul sursă, după care se va trece direct la execuție.

Astfel, un programator poate utiliza un interpretor pentru a pune la punct un program sursă BASIC, după care poate utiliza un compilator compatibil, pentru a obține un program obiect, care poate fi executat într-un timp foarte scurt, corespunzător aplicației date.

În comparație cu un compilator, interpretorul este mult mai ieftin.

Metoda bazată pe pseudo-compilare asigură un compromis între viteză și cost. Astfel, costul unui pseudo-compilator este apropiat de cel al unui interpretor. În timp ce viteza sa este de circa cinci ori mai mare decât a acestuia din urmă. Viteza relativă a pseudo-compilatorului depinde de instrucțiunile programului, de tipul de date și de gradul în care codul rezultat necesită translatarea în timpul execuției.

Varianta de BASIC aleasă pentru aMIC are un caracter interpretativ, ceea ce o face extrem de atractivă pentru punerea la punct a programelor. Marea majoritate a aplicațiilor pentru acest calculator nu intră în categoria aplicațiilor de timp real.

În cazul în care se impune o viteză de execuție mai mare, se poate folosi metoda apelării unor subrutine, scrise în limbaj de asamblare, pentru acele zone din program care trebuie executate cu viteză mare.

Asemenea situații apar în cadrul aplicațiilor de colectare a datelor sau de conducere a unor procese. Pentru a realiza accesul la subrutine scrise în limbaj de asamblare se folosește instrucțiunea CALL, care include un nume de variabilă (un indicator la o celulă de memorie în care se află punctul de intrare în subrutina scrisă în limbaj de asamblare) și dacă este necesar, o listă de parametri care trebuie transferați subrutinei.

Pentru punerea la punct a programelor în limbaj de asamblare se poate folosi programul de sistem MATE (Monitor-Assembler-Text Editor), programul asamblat fiind stocat pe caseta magnetică. În continuare el va fi încărcat în memoria operativă a calculatorului o dată cu programul BASIC-sursă. Într-unul din paragrafele acestui capitol se dau exemple pentru ilustrarea acestei tehnici.

Calculatorul aMIC are disponibile două versiuni de interpretoare BASIC. Prima versiune reprezintă un subset al celei de-a doua. În sensul că nu posedă instrucțiuni matriceale, instrucțiuni de prelucrare grafică și instrucțiunea CALL. Prima versiune necesită un spațiu de 8 Ko în memoria PROM, în timp ce versiunea a doua necesită un spațiu de 14 Ko. Se menționează și posibilitatea încărcării interpretorului BASIC de pe caseta magnetică, în scopul reducerii numărului de circuite integrate de memorie PROM al căror cost este relativ ridicat.

Operarea sistemului. Elemente introductive privind programarea în limbajul BASIC

În vederea unei sesiuni de lucru, calculatorul se conectează la intrarea de antenă a unui televizor și la o tastatură alfanumerică*. Televizorul va fi folosit ca echipament de ieșire, în timp ce tastatura va juca rolul unui echipament de intrare.

* Amănunțele privind conectarea la un televizor comercial, cât și tastatura slot date în capitolul 3 al lucrării.

După alimentarea de la rețea a televizorului și selectarea canalului corespunzător se va alimenta și calculatorul, prin intermediul sursei proprii, și se va acționa tasta RESET. În acest moment pe ecranul televizorului va apărea un text, care indică faptul că sistemul se află sub controlul programului monitor. În continuare se pot introduce, de la tastatură, comenzi monitor (descrise în capitolul referitor la monitorul sistemului). Pentru a lansa BASIC-ul, trebuie acționate tastele B și RETURN. Pe ecran va apărea mesajul HEADY, indicând posibilitatea introducerii unor comenzi și instrucțiuni BASIC. De exemplu, dacă se dorește calculul valorii unei expresii aritmetice: $30 - 2 \cdot 3^2 + 4 : 2 \cdot 3$, se va introduce de la tastatură următoarea instrucțiune:

```
10 PRINT 30 - 2 * 3 ^ 2 + 4 / 2 * 3
```

Se observă că instrucțiunea începe cu o etichetă numerică (10). La sfârșitul introducerii instrucțiunii trebuie acționată tasta RETURN. În continuare linia (instrucțiunea) introdusă este depusă în memoria calculatorului, fără a se executa. Se constată că operațiile de înmulțire, împărțire și ridicare la putere sînt descrise prin simbolurile *, / și ^.

În memoria calculatorului pot fi introduse de la tastatură succesiv mai multe instrucțiuni, avînd numere de linie diferite. Un grup de instrucțiuni depuse în memorie poartă numele de program.

Pentru a executa o instrucțiune (un program) depusă în memoria calculatorului, trebuie dată comanda RUN (urmată de RETURN). Astfel, introducînd instrucțiunea PRINT, și apoi comanda RUN, calculatorul va afișa pe ecran rezultatul expresiei (18), prioritatea în execuția operatorilor fiind cea obișnuită (întîi ridicări la putere, apoi înmulțiri și împărțiri și în final adunări și scăderi). După tipărirea rezultatelor, pe ecran se va afișa un mesaj de eroare cu numărul 1, indicînd faptul că programul executat nu s-a terminat cu o instrucțiune de oprire a execuției (STOP sau END). Pentru a evita această eroare, se va introduce după PRINT o nouă instrucțiune: 20 END.

Executînd din nou programul (format din două instrucțiuni: 10 PRINT ... și 20 END) și folosind tot comanda RUN, nu se va mai obține mesajul de eroare.

În instrucțiunea PRINT pot apărea mai multe expresii (după necesități) separate prin virgulă. De exemplu instrucțiunea:

```
10 PRINT 1+7, 3, 5, 5*2+3, 19/4
```

Va tipări cinci valori, cîte două pe un rînd separate cu spații. De notat că ecranul televizorului este împărțit în 32 de rînduri a 30 de caractere fiecare. În cazul folosirii virgulei în instrucțiunea PRINT valorile sînt afișate începînd din poziția 1 și apoi din poziția 15 a unui rînd, realizîndu-se o tabulare automată. Pentru a nu se lăsa spațiu (tabula) între valorile tipărite, se folosește separatorul ; între expresii.

Astfel, 10 PRINT 8; 15/2-1; 2+7; 19 va tipări cele patru valori 8, 6.5, 128 și 19 pe același rînd (fără tabulare).

Pentru a memora o valoare (un număr) care va fi folosită ulterior în program, se va utiliza o variabilă literală, căreia i se va atribui valoarea respectivă.

Limbajul BASIC dispune în acest sens de instrucțiunea de atribuire **LET**, cu ajutorul căreia se stochează în memoria calculatorului mărimi ce pot fi utilizate ulterior. Astfel, folosind variabila **X**:

```

5 LET X=15          30 PRINT X
10 PRINT X          40 STOP
20 PRINT X/2, X+2-X+1

```

valoarea tipărită de instrucțiunile 10 și 30 este aceeași (15). Instrucțiunea 20 utilizează valoarea variabilei **X**, fără a o modifica.

9.2. Elementele limbajului BASIC

În continuare vor fi prezentate câteva noțiuni necesare pentru descrierea unei probleme în limbajul BASIC. Cuvintele folosite în limbajul BASIC formează vocabularul limbajului. Ele se scriu după reguli precise, date de sintaxa limbajului.

Caracterele întrebuințate în BASIC pentru alcătuirea cuvintelor sînt:

- literele mari ale alfabetului: A, B, C, ..., Z
- cifrele: 0, 1, ..., 9
- caractere speciale: + - * / = " > < . , † () ; \$

9.2.1. Constante. Constantele sînt de două tipuri: constante numerice și constante șir sau texte. În continuare, în lipsa menționării explicite, prin constantă ne vom referi la constante numerice. Constantele șir reprezintă orice șir de caractere introdus între ghilimele. De exemplu: "STUDENTII HARNICI", "INTRODUCEȚI DATELE", "127.5", etc.

Constantele numerice utilizate în BASIC sînt reale. Ele pot avea, de exemplu, următoarele exprimări:

```
25    123.45    +4    -0.30    +.7321
```

De asemenea, la fiecare număr de mai sus, se poate adăuga un exponent, utilizînd litera **E**. Exponentul este un număr întreg. El indică puterea lui 10, cu care se înmulțește numărul.

Astfel, următoarele constante sînt corecte:

```
25E-1=2.5    123.45E12    -0.30E-15
```

Intern, constantele sînt reprezentate în formatul cu virgulă mobilă, pe patru octeți. Primii trei octeți conțin mantisa, subunitară și normalizată, iar octetul patru conține exponentul:

M			S	E
0	23	24		31

unde:

M este mantisa normalizată $0.5 \leq M < 1$

S este un bit care reprezintă semnul mantisei (1 pentru negativ).

E este exponentul în complement față de doi. **E** reprezintă puterea lui doi, cu care se înmulțește mantisa.

Conform cu această reprezentare internă, cel mai mic număr manipulat va fi (în modul) : $2.71051E-20$ iar cel mai mare : $9.22337E18$. Din reprezentare se observă că sînt păstrate aproximativ 7 cifre semnificative.

9.2.2. Variabile. Variabilele utilizate în BASIC pot fi, ca și constantele, de două tipuri : variabile numerice (pe care le vom numi, pe scurt, variabile) și variabile șir.

Variabilele șir reprezintă șiruri de caractere alfanumerice. Numele unei variabile șir este format dintr-o literă urmată de caracterul $\$$.

Exemple : A \$, B \$

Se pot utiliza și tablouri de variabile șir, tablourile fiind formate din mulți multe variabile șir, cu aceeași lungime și același nume.

Numărul și lungimea variabilelor șir ce formează un tablou, trebuie declarate într-o instrucțiune DIM, înainte de utilizarea tabloului. Variabilele șir ce formează un tablou, vor fi specificate prin utilizarea indicilor. De exemplu, dacă A \$ este un tablou de zece variabile șir, a câte 50 de elemente (caractere) fiecare, atunci vom referi variabilele șir prin : A \$ (1), A \$ (2), ... A \$ (10). Iar tabloul A \$ va fi declarat într-o instrucțiune :

DIM A \$ (10, 50).

În unele aplicații, este necesară utilizarea unor porțiuni (subșiruri) dintr-un șir, desemnat de o variabilă șir. Pentru specificarea subșirurilor unei variabile șir, se folosește notația : $(n_1 \text{ TO } n_2)$ asociată numelui variabilei șir ; n_1 este indicele primului caracter al subșirului, iar n_2 este indicele ultimului caracter al subșirului din variabila șir. De exemplu, fie B \$ = "123ABC", subșirul B \$ (2 TO 5) = "23AB".

Indicii n_1 și n_2 pot fi constante, variabile sau expresii, valoarea lor fiind cuprinsă între 1 și dimensiunea (lungimea) șirului. În cazul absenței indicelui n_1 , se consideră că subșirul începe cu primul caracter al șirului, de exemplu : B \$ (TO 4) = "123A".

În cazul absenței indicelui n_2 se consideră că subșirul se termină cu ultimul caracter al șirului, de exemplu : B \$ (3 TO) = "3ABC". Pot lipsi ambii indici, caz în care subșirul este identic cu șirul (variabila șir) dat. Exemplu : B \$ (TO) = B \$ = "123ABC".

Se pot utiliza subșiruri ale variabilelor șir ce formează un tablou de variabile șir.

De exemplu, pentru tabloul A \$ (10, 50) format din zece șiruri de câte 50 de caractere, se poate specifica subșirul A \$ (3, 12 TO 43), format din elementele 12 la 43, ale variabilei șir A \$ (3).

De remarcat că, în cazul unui tablou de variabile șir, trebuie selectată variabila șir din tabloul la care ne referim și apoi specificat un subșir din această variabilă.

Pentru subșiruri formate dintr-un singur caracter, se poate folosi un singur indice, cel al caracterului selectat.

Astfel : B \$ (4) = B \$ (4 TO 4) = "A" ; B \$ (6) = "C", pentru variabila B \$ utilizată mai sus, iar în cazul tabloului A \$, elementul (caracterul) al treilea din șirul șapte va fi specificat prin A \$ (7, 3) sau A \$ (7, 3 TO 3).

Variabilele numerice utilizate în BASIC sînt reale. Variabilele pot fi simple sau indexate. Variabilele simple sînt reprezentate fie printr-o literă, fie o literă și o cifră.

Exemple: A, V1, Q7, A9

Variabilele indexate reprezintă elementele unui tablou (vector sau matrice). Identificatorul (numele) tabloului trebuie să fie compus dintr-o singură literă. Litera identificator de tablou poate coincide cu numele unei variabile simple (fără a produce conflict).

Tablourile, înainte de utilizare, trebuie declarate în instrucțiunea DIM (sau o instrucțiune MAT). Indicele poate fi o constantă, o variabilă sau o expresie și trebuie să aibă valoarea:

$1 \leq \text{indice} \leq \text{dimensiunea declarată în DIM}$

Observație:

Dacă, în urma evaluării expresiei indice, nu se obține o valoare întreagă, se va reține partea întreagă a valorii obținute.

Exemple: E(1,5); V(ABS(R)); A(I-3, J-K)

9.2.3. Operatori. Operatori aritmetici:

↑ ridicare la putere,
*, / înmulțire, împărțire,
-, + scădere, adunare.

Operatorii au fost scriși în ordinea priorității în evaluare (↑ are prioritatea cea mai mare). Când se dorește schimbarea priorității în evaluare, sau când există dubii, este bine să se utilizeze parantezele; operațiile din interiorul parantezelor vor fi executate înaintea celor din exterior.

Operatori relaționali

Operatorii relaționali sînt utilizați în instrucțiunea IF pentru a determina relația dintre valorile a două expresii:

operator	semnificație	Operator	semnificație
=	egalitate	>= sau =>	mai mare sau egal,
>	mai mare]	<= sau =<	mai mic sau egal,
<	mai mic	<> sau ><	neegalitate.

9.2.4. Funcții. În alcătuirea expresiilor pot fi utilizate următoarele funcții matematice:

SIN(x) — sinus de x, unde x este un unghi exprimat în radiani;
 COS(x) — cosinus de x, unde x este exprimat în radiani;
 TAN(x) — tangentă de x, unde x este exprimat în radiani;
 ATN(x) — arctangentă de x. Rezultatul este exprimat în radiani:

$$(-\pi/2 < \text{ATN}(x) < \pi/2);$$

LOG(x) — logaritm natural din x;
 EXP(x) — calculează e^x ;

Observație :

Constantele π și e sînt definite intern, în interpretor, și pot fi utilizate în expresii prin simbolurile PI și respectiv EE.

SQR(x) — calculează rădăcina pătrată din x.

ABS(x) — calculează valoarea absolută a lui x.

INT(x) — calculează cel mai mare întreg $\leq x$.

RND(x) — calculează un număr aleator în intervalul (0, 1). Valoarea argumentului x nu are importanță în calcul.

SGN(x) — returnează :

1, dacă $x > 0$

0, dacă $x = 0$

-1, dacă $x < 0$.

În toate funcțiile descrise, argumentul x poate fi în general o expresie oarecare.

Argumentele funcțiilor LOG și SQR trebuie să fie numere pozitive, în caz contrar sistemul va răspunde printr-un mesaj de eroare.

În situația cînd, în calculul unei funcții (expresii), rezultatul depășește scala numerelor reprezentabile în calculator ($2.71051E-20$, $9.22337E18$), se afișează la consolă un mesaj de eroare (UNDERFLOW sau OVERFLOW IN LINE NN), execuția programului continuînd. Ca rezultat al funcției (expresiei) se ia cea mai mică, respectiv cea mai mare valoare reprezentabilă (corespunzător depășirii). De pildă, în cazul funcției EXP(x), valoarea argumentului trebuie să fie cuprinsă aproximativ între $-45 < x < 44$, pentru ca rezultatul e^x să poată fi reprezentat intern.

Funcția INT calculează cel mai mare întreg, mai mic, sau egal cu argumentul.

Astfel: $INT(7.25) = 7$

$INT(-7.25) = -8$

$INT(-1) = -1$

INT poate fi utilizată pentru a rotunji un număr la cel mai apropiat întreg.

Astfel, $INT(x+0.5)$ va calcula întregul cel mai apropiat de x.

Pe lângă funcțiile descrise mai sînt disponibile: funcții pentru lucrul cu șiruri de caractere, precum și funcții speciale de intrare/ieșire.

Astfel, pentru lucrul cu șiruri de caractere, se pot utiliza funcțiile :

VAL(șir) — calculează valoarea numerică a șirului, tratat ca o expresie aritmetică. De exemplu: $VAL("127.5-B") = 7.5$, dacă variabila B are valoarea 120.

LEN(șir) — calculează lungimea șirului specificat. De exemplu :

$LEN("ABC123") = 6$, sau dacă $A\$ = "B79"$, atunci $LEN(A\$) = 3$

STR\$(expresie) — calculează valoarea expresiei, iar rezultatul formează un șir de caractere, ca și cum ar fi tipărit cu instrucțiunea PRINT.

De exemplu STR(127.5-B) = "7.5"$, dacă variabila B are valoarea 120.

CHR\$(expresie) — calculează (determină) caracterul care are codul ASCII egal cu valoarea expresiei. De exemplu $\text{CHR}(65) = "A"$

INKEY\$ — citește un caracter de la tastatură (în cazul în care a fost acționată o tastă) și întoarce codul ASCII al caracterului. În cazul când nu s-a acționat nici o tastă rezultatul este șirul nul ("").

Funcțiile VAL și LEN pot fi folosite în orice expresie (aritmetică), însă pe prima poziție în cadrul expresiei. Funcțiile STR\$, CHR\$ și INKEY\$ pot fi folosite numai în expresii de tip șir.

Pentru executarea unei operații de intrare/ieșire, pe un port specificat, se pot folosi funcțiile :

GET(x) — citește un octet de la portul numărul x, $0 \leq x \leq 255$.
Valoarea funcției va fi un număr întreg, $0 \leq \text{GET}(x) \leq 255$.
Funcția GET poate fi folosită în orice expresie.

PUT(x) — poate apărea numai în membrul stâng al unei instrucțiuni de atribuire. Execuția funcției constă în transmiterea la portul cu numărul x, a valorii expresiei din membrul drept al instrucțiunii de atribuire.

Exemple :

10 PUT(127) = A + 17

Expresia din membrul drept este evaluată, convertită în întreg, iar cei mai puțin semnificativi opt biți sunt trimiși la portul 127.

Instrucțiunea : 15 PUT (127) = GET (255), va citi un octet de la portul 255 și-l va transmite la portul 127.

9.2.5. **Expresii.** Expresiile utilizate în instrucțiunile BASIC sînt de două tipuri : expresii aritmetice (le vom numi, pe scurt, expresii) și expresii șir (sau șiruri).

Expresiile șir pot conține ca operanzi : constante șir, variabile șir sau subșiruri precum și funcțiile ce au ca valoare șiruri de caractere : STR\$, CHR\$, și INKEY\$. Ca operator, în formarea expresiilor șir, poate fi utilizat operatorul de concatenare, notat +. De exemplu : "123" + "ABC" = "123ABC", sau $\text{CHR}(65) + \text{STR}(12 + 7) + "CASA" = "A12CASA"$.

Expresiile aritmetice pot fi compuse din constante, variabile simple sau indexate, funcții, legate între ele prin operatori aritmetici etc.

Exemple : 100

VAL ("70.5+B")

(5-x^4 + 3) * (Z/(k-1) - A/LOG(Z+1))

Ordinea de execuție a operațiilor.

— În expresiile fără paranteze succesiunea este următoarea :

1. calculul valorilor funcțiilor,
2. ridicarea la putere,
3. înmulțiri și împărțiri (cu prioritate egală),
4. adunări și scăderi (cu prioritate egală).

— Dacă două sau mai multe operații de aceeași prioritate apar într-o expresie, ele se efectuează în ordine, de la stînga la dreapta, cu excepția ridicării la putere, unde execuția este de la dreapta spre stînga.

Exemplu :

$$3 \uparrow 2 \uparrow 3 = 3 \uparrow 8 = 6561$$

— Dacă într-o expresie apar paranteze, se vor executa întâi operațiile cuprinse între parantezele interioare.

Observații :

1. Singurele paranteze admise sînt parantezele rotunde.
2. Operațiile aritmetice trebuie scrise explicit.

Exemplu :

54B se va scrie : $5 * A * B$.

9.2.6. Instrucțiuni și comenzi. Elementele principale ale limbajului BASIC sînt instrucțiunile și comenzile limbajului. Sintactic, instrucțiunile se deosebesc de comenzi prin faptul că orice instrucțiune este etichetată, adică orice instrucțiune începe cu un număr întreg (pe care-l vom numi număr de linie), cuprins între 0 și 32767.

Numerele de linie au un dublu rol :

— determină ordinea de execuție a instrucțiunilor (instrucțiunile pot fi introduse în orice ordine, însă vor fi executate în ordinea crescătoare a numerelor de linie);

— sînt utilizate în instrucțiunile de transfer, pentru referirea instrucțiunilor.

Comenzile și instrucțiunile vor fi prezentate detaliat în capitolele următoare.

9.2.7. Exerciții

1. Dacă dispuneți de un tabel cu logaritmul, încercați să verificați următoarea regulă : a ridică 10 la o putere este echivalent cu inversul logaritmului din acea, număr. Introduceți :

```
10 PRINT 10 ↑ 0.3020
20 END
RUN
```

și urmăriți în tabelu de logaritmi, inversul logaritmului din 0.3020. De ce cele două rezultate nu sînt exact egale ?

Observație : În continuare, instrucțiunea END de la sfîrșitul programului și comanda RUN ce lansează în execuție programul, nu vor mai fi specificate explicit, considerîndu-se subînțeles.

2. S-a văzut că numerele sînt reprezentate în formatul cu virgulă mobilă, format care permite reprezentarea unei game largi de valori numerice reale, precizia reprezentării (6—7 cifre semnificative) nedepășind de valoare. Executați însă următoarea instrucțiune :

```
10 PRINT 1E8 + 1 — 1E8, 1E8 — 1E8 + 1 "
```

Observați că pentru calculator $1E8$ și $1E8 + 1$ sînt valori egale.

3. Pentru obținerea logaritmilor zecimali (cei găsiți în tabelele de logaritmi), împărțiți logaritmul natural prin $\text{LOG}(10)$. Astfel, pentru a calcula $\log 2$ se folosește instrucțiunea :

```
10 PRINT LOG(2)/LOG(10)
```

obținind rezultatul 0.301029. Încercați să faceți înmulțiri și împărțiri cu logaritmi servindu-vă de calculator ca de o tabelă de logaritmi. Pentru test, puteți folosi exercițiul 1, care calculează inversul logaritmului.

4. EXP și LOG sînt funcții inverse una alteia. Apelîndu-le succesiv unui număr, îl vor lăsa neschimbat.

Exemplu : $\text{LOG}(\text{EXP}(2)) = \text{EXP}(\text{LOG}(2)) = 2$.

Acest lucru este valabil și pentru funcțiile TAN și ATN. Puteți utiliza acest exercițiu pentru a testa precizia de calcul a funcțiilor respective de către calculator.

5. Se știe că π radiani reprezintă 180° . Pentru a transforma o mărime exprimată în grade, în radiani, o împărțim prin 180 și înmulțim cu π , astfel :

```
10 PRINT TAN((45/180 * PI)
```

calculează $\tan 45^\circ (=1)$. Pentru a transforma radianii în grade, trebuie să împărțim prin π și să înmulțim cu 180 .

6. Cum utilizați funcțiile RND și INT pentru a obține un număr aleator între 1 și 6, care ar putea reprezenta aruncarea unui zar ? (Răspuns : $\text{INT}(\text{RND}(1) * 6 + 1)$).

7. Pentru a testa precizia cu care este memorat π de către calculator, executați instrucțiunea : `10 PRINT PI, PI-3, PI-3.1, PI-3.14, PI-3.141.`

8. Funcția INT rotunjește la întreg prin lipsă.

Pentru a rotunji la întregul cel mai apropiat se adună 0.5 .

Exemple :

$\text{INT}(2.9 + 0.5) = 3$

$\text{INT}(5.4 + 0.5) = 5$

$\text{INT}(-2.9 + 0.5) = -3$

$\text{INT}(-5.4 + 0.5) = -5$

Comparați cele de mai sus cu rezultatele obținute în cazul că nu adunați 0.5 .

9. De ce numele unei variabile trebuie să înceapă cu o literă ?

10. Reamintim câteva reguli de calcul cu puteri

$A \uparrow 0 = 1$

$A \uparrow (-B) = 1 / A \uparrow B$

$A \uparrow (1/B) = \sqrt[B]{A}$ rădăcina de ordin B din A

$A \uparrow (B+C) = A \uparrow B * A \uparrow C$

$A \uparrow (B * C) = (A \uparrow B) \uparrow C$

unde A și B sînt numere întregi pozitive.

Testați aceste reguli utilizînd calculatorul pentru calculul diferitelor expresii conținînd \uparrow . De exemplu :

```
10 PRINT 3 \ (2 + 0), 3 \ 2 = 3 \ 0
```

```
20 PRINT 4 \ (-1), 1/4
```

11. Valoarea lui e, baza logaritmilor naturali, este 2.71828. Ea este memorată intern cu numele EE. Verificați dacă :

$\text{EXP}(X) = \text{EE} \uparrow X$

pentru diverse valori ale lui X.

12. Executați următoarele instrucțiuni :

```
10 LET A$ = "ATN(1) * 4"
```

```
20 PRINT A$ ; "=" ; VAL(A$)
```

(Rezultatul tipărit este π)

Repetăți execuția schimbând instrucțiunea 10.

De exemplu :

```
5 LET X=9
10 LET A$="X+3+2 * X"
```

Reexecutați pentru expresii din ce în ce mai complicate atribuite variabilei și A\$.

13. Unele versiuni ale limbajului BASIC utilizează următoarele funcții pentru lucrul cu șiruri: LEFT\$, RIGHT\$, MID\$ și TL\$, cu următoarele semnificații:

- LEFT\$(A\$, N) — calculează subșirul lui A\$ compus din primele N caractere.
 RIGHT\$(A\$, N) — calculează subșirul lui A\$ compus din ultimele caractere începând cu al N-lea.
 MID\$(A\$, N1, N2) — calculează subșirul lui A\$ compus din N2 caractere începând cu al N1-lea caracter din A\$.
 TL\$(A\$) — calculează subșirul lui A\$ format din toate caracterele lui A\$ cu excepția primului.

Încercați să descrieți funcțiile de mai sus cu ajutorul decupării cu T.O.

14. Executați programul :

```
10 A$=CHR$(INT (RND (0)*10+48))
20 PRINT A$
30 GOTO 10
```

(Instrucțiunea GOTO 10 indică repetarea execuției de la linia 10).

Programul tipărește o cifră (între 0 și 9), deoarece 48 este codul cifrei 0 la care se adună un număr aleator între 0 și 9, dând codul ASCII al unei cifre (între 48 și 57). Puteți da altă soluție pentru generarea unui număr întreg între 0 și 9?

9.3. Comenzile și modul de utilizare al interpretorului BASIC

9.3.1. Lansarea în execuție a interpretorului BASIC. Interpretorul BASIC este depus în memoria cu conținut permanent (PROM), începând de la adresa 800H (hexazecimal). De aceea poate fi lansat în execuție cu ajutorul comenzii G0800, a monitorului sistemului, ca orice program utilizator. Pentru simplitate, s-a prevăzut între comenzile monitorului, o comandă specială pentru a lansa în execuție interpretorul BASIC, anume comanda B*.

Când interpretorul BASIC intră în execuție, va tipări pe display mesajul READY, indicând faptul că se așteaptă instrucțiuni sau comenzi de la utilizator. În continuare, utilizatorul va introduce un program de la tastatură sau va cili un program de pe casetă magnetică. Programul astfel introdus poate fi executat, listat pe display sau modificat (cu ajutorul unor facilități de editare) după dorința utilizatorului. După execuție, programul poate fi salvat pe casetă și șters din memorie, pentru a se introduce un nou program.

La terminarea unei sesiuni de lucru cu interpretorul BASIC, se va ieși de sub controlul interpretorului acționând comutatorul (RESET) de la consolă, lansându-se în execuție monitorul sistemului.

Există două situații speciale de lansare în execuție a interpretorului BASIC.

* Comanda B este disponibilă numai în versiunea V0.1.

În prima situație, se utilizează subrutine în limbaj mașină, apelate dintr-un program BASIC. În acest caz, înainte de lansarea în execuție, a interpretorului, se va introduce în memorie codul pentru subrutinele apelate, cu ajutorul comenzilor monitorului. Apoi se lansează interpretorul BASIC de la adresa 815H (deci cu comanda G0815).

În cea de-a doua situație lansarea interpretorului are loc de la adresa 829H, fără inițializări. Lansarea de la această adresă (cu G0829) se folosește când în memoria utilizată de interpretor pentru păstrarea programelor, se află un program care nu trebuie șters (de exemplu când s-a ieșit din BASIC în Monitor și se dorește să se intre iarăși sub controlul interpretorului, fără ștergerea programului BASIC introdus în sesiunea anterioară).

9.3.2. Editarea programului. Există facilități de corectare a unei linii în cursul introducerii ei de la consolă (tastatură), sau de editare a programului deja introdus (ștergerea unei linii sau înlocuirea ei cu altă linie).

Astfel :

- ștergerea ultimului caracter introdus se realizează acționând DEL
- ștergerea liniei în curs de introducere se face acționând simultan CTRL și Y
- ștergerea unei instrucțiuni din program se face prin tipărirea numărului ei de linie și RETURN.

O înlocuirea se poate face prin tipărirea noii instrucțiuni cu același număr de linie cu vechea instrucțiune.

Observație : Orice linie introdusă de la consolă (instrucțiune, comandă sau linie de date) va fi luată în considerare de sistem la acționarea lui RETURN.

9.3.3. Listarea și salvarea pe casetă a unui program. Un program BASIC aflat în memorie poate fi listat la display sau salvat (stocat) pe casetă magnetică. Aceste operații pot fi executate indiferent dacă programul a fost executat sau nu. Pentru listarea programului se folosește comanda :

LIST N1,N2

unde : N1 și N2 sînt numere de linie.

Execuția comenzii constă în listarea programului existent în memorie în ordinea crescătoare a numerelor de linie. Parametrii N1 și N2 sînt opționali. În cazul în care se specifică un singur număr de linie se vor lista instrucțiunile ce au numărul de linie mai mare sau egal cu numărul specificat în comanda LIST. Când se specifică ambii parametri, se vor lista instrucțiunile care au numărul de linie cuprins între N1 și N2 inclusiv. Execuția comenzii poate fi oprită de la consolă, acționând CTRL.

Pentru salvarea pe casetă a unui program se folosește comanda SAVE. Dacă programul folosește subrutine în limbaj mașină, la comanda SAVE se salvează automat și subrutinele apelate. Dacă comanda SAVE este executată după ce programul a fost executat, atunci se salvează pe casetă și valorile variabilelor utilizate în execuția programului.

Deoarece identificarea programelor de pe casetă se face manual de către utilizator, este bine ca înainte de salvarea unui program să se înregistreze

pe casetă un text vorbit (titlu), referitor la programul ce urmează, necesar identificării. Apoi se conectează casetofonul pe înregistrare la calculator și se lansează în execuție comanda SAVE.

9.3.4. Citirea unui program de pe casetă. Pentru citirea unui program de pe casetă se folosește comanda LOAD. Înainte de a executa comanda LOAD, trebuie să se depisteze (sonor) începutul programului (după titlul înregistrat la salvarea programului) și apoi să se conecteze casetofonul la calculator, pe redare. Programul este citit de pe casetă în memoria calculatorului împreună cu eventualele subrutine în limbaj mașină și cu variabilele, exact în starea în care a fost salvat, vechiul program din memorie fiind șters.

Pentru o citire cât mai fiabilă este bine ca volumul redării să fie reglat corespunzător.

9.3.5 Execuția unui program. Pentru a lansa în execuție un program BASIC aflat în memorie se pot folosi comenzile :

RUN nr. linie

sau

GOTO nr. linie

execuția programului începând de la linia cu numărul specificat în comandă. Pentru comanda RUN, numărul de linie poate să lipsească, caz în care execuția începe cu prima instrucțiune din program. Deosebirea dintre cele două comenzi constă în faptul, că RUN inițializează (șterge) variabilele înainte de lansarea în execuție, pe când GOTO păstrează valorile variabilelor obținute într-o execuție anterioară. De exemplu, dacă se citește un program de pe casetă împreună cu valorile variabilelor (eventual tablouri), atunci lansarea sa în execuție se va face cu comanda GOTO n1, n1 fiind numărul de linie al instrucțiunii aflată după instrucțiunea DIM în program, pentru a se putea folosi valorile variabilelor tablou salvate pe casetă. Dacă nu se dorește utilizarea valorilor variabilelor, obținute într-o execuție anterioară, programul poate fi lansat în execuție cu comanda RUN.

Observație : Pentru a nu se salva valorile variabilelor pe casetă atunci când nu sînt necesare, trebuie să se știe faptul că tabela de variabile este ștersă în următoarele cazuri :

- la introducerea unei noi linii în program,
- la ștergerea unei linii din program,
- la execuția comenzii SCRATCH (însă în acest caz este șters și programul).

Execuția unui program este oprită la :

- depistarea unei erori în program,
- execuția unei instrucțiuni STOP sau END
- întreruperea de la consolă, acționînd caracterul CTRL.
- întreruperea de la consolă, acționînd RESET, caz în care se intră sub controlul Monitorului.

La terminarea execuției, sistemul tipărește READY. După eventuale corectări lașerări de instrucțiuni, programul poate fi executat din nou.

9.3.6. Ștergerea unui program din memorie. Înainte de introducerea unui nou program de la tastatură, vechiul program din memorie trebuie șters, altfel liniile lui vor interfera cu liniile noului program. Pentru ștergerea programului din memorie (inclusiv a eventualelor subrutine în limbaj mașină) se folosește comanda: SCRATH (sint suficiente primele trei caractere: SCR).

9.3.7. Exerciții

1. S-a văzut că variabilele utilizate de un program în timpul execuției sînt șterse din memorie la introducerea sau ștergerea unei instrucțiuni a programului și la comanda SCR. În toate aceste cazuri, este alternat însă și programul o dată cu ștergerea variabilelor (desigur, există posibilitatea de a șterge o linie și a o reintroduce cu același conținut).

O posibilitate de a șterge variabilele fără a altera programul, este prin execuția comenzii RUN n1, unde n1 este numărul de linie al instrucțiunii END sau STOP din program.

Acest lucru este valabil deoarece, la începutul execuției programului, prin comanda RUN se șterg variabilele utilizate de program (într-o execuție anterioară), iar apoi se va executa instrucțiunea END sau STOP. Ștergerea variabilelor este utilă, în cazul salvării pe casetă a programelor ce conțin variabile (eventual tablouri), ale căror valori nu sînt necesare la relucărarea de pe casetă. Testați acest lucru, pentru programe care conțin tablouri suficient de mari, și veți observa diferența de timp la salvare și încărcare pe casetă.

2. Executați următorul program :

```
10 LET X=7
20 PRINT X
30 END
```

Folosind comanda RUN. Executați acum programul începînd de la linia 20 cu comenzile GOT020 și RUN20. Observați că rezultatul este diferit (în primul caz este 7 în al doilea 0). Acest lucru se datorează faptului că, comanda RUN șterge variabilele rămase dintr-o execuție anterioară (lucru care s-a mai spus), iar interpretorul BASIC consideră variabilele nedefinite ca avînd valoarea zero.

3. Pentru a testa faptul că comanda SAVE salvează împreună cu programul și variabilele utilizate, se poate introduce un program, de exemplu cel de la exercițiul 2, și după execuție se va salva pe casetă. Pentru salvare se vor executa următoarele acțiuni :

- poziționarea benzii în zona în care se dorește să se înregistreze programul,
- utilizînd microfonul, se va da un nume programului ; acest lucru nu este absolut necesar, dar este util pentru regăsirea ulterioară a programului,
- se conectează casetofonul (prin cablul corespunzător) cu calculatorul,
- se introduce comanda SAVE (fără a acționa tasta RETURN),
- se comută casetofonul pe înregistrare,
- se acționează RETURN,
- se urmărește apariția pe ecranul televizorului a mesajului de așteptare de operație (READY). La apariția mesajului, opriți casetofonul. Pe durata înregistrării se va auzi un zgomot specific, în difuzorul asociat calculatorului, iar imaginea pe televizor va fi formată din dungi variabile (în primele 10 secunde din înregistrare, se depune un preambul necesar sincronizării, iar apoi programul BASIC și variabilele).

În continuare, se șterge programul din memorie cu ajutorul comenzii SCR (se poate testa, că memoria nu conține nici un program, introducînd comanda LIST).

În continuare, se încarcă programul de pe casetă în memorie executînd următoarele acțiuni :

- se reabobinează banda pentru a o aduce la începutul fișierului (zonei) cu programul. Pentru depistarea începutului programului se va face uz de numele înregistrat înainte de programul,
- se conectează casetofonul la calculator, pe redare,
- se reglează volumul redării suficient de mare (3/4), iar tonalitatea la valoare înaltă (dacă casetofonul are reglaj de tonalitate),
- se introduce comanda LOAD (fără a acționa tasta RETURN).

— se pornește casetofonul.

— se urmărește dacă beculuțel de la consolă s-a aprins și apoi se acționează RETURN.

— se urmărește apariția pe ecranul televizorului a mesajului de sfârșit de încărcare.

Astfel, în cazul în care operația a decurs fără erori, vor fi afișate două numere în hexazecimal care reprezintă : adresa de început a zonei de memorie salvată pe casetă și respectiv — lungimea ei (numărul de octeți). Apoi este tipărit mesajul READY, de către interpretor. În cazul în care s-au depistat erori la încărcare, monitorul va tipări ? și va prelua controlul. În acest caz, încărcarea programului de pe casetă trebuie reluată, însă este indicat a se intra în BASIC din monitor cu comanda G0829 (fără inițializări), pentru a se lista (cu LIST) programul încărcat și a se constata amploarea erorii de încărcare. Uneori eroarea poate fi corectată editând o anumită linie, fără a mai încărea încă odată programul de pe casetă.

După încărcarea programului, se execută comanda GOTO20, și se observă că rezultatul este 7, deci împreună cu programul a fost salvată și variabila X.

4. Introduceți un program BASIC și apoi opriți alimentarea de la rețea a calculatorului. Realignând calculatorul și lansând interpretorul BASIC veți constata că programul s-a pierdut. Deel, în cazul programelor mari, este indicat salvarea lor pe casetă, chiar în faza de punere la punct, pentru a nu risca reintroducerea lor în întregime.

9.4. Instrucțiunile limbajului BASIC

9.4.1. Exemplu de program. Înainte de a începe prezentarea instrucțiunilor limbajului BASIC, se va da un exemplu de program scris în BASIC, pentru rezolvarea sistemului de două ecuații lineare cu două necunoscute.

$$ax + by = c$$

$$dx + ey = f$$

Sistemul are soluție unică dacă $ae - bd \neq 0$, caz în care :

$$x = \frac{ce - bf}{ae - bd} \quad \text{și} \quad y = \frac{af - cd}{ae - bd}$$

Dacă $ae - bd = 0$, atunci, fie nu există nici o soluție (sistem imposibil), fie există o infinitate de soluții (sistem nedeterminat). Însă, nu există soluție unică. Programul este următorul :

```

10 REM "PROGRAM CE REZOLVA UN SISTEM"
15 REM "DE DOUA ECUATII LINEARE"
20 PRINT "INTRODUCETI COEFICIENTII A, B, C, D"
30 INPUT A, B, C, D
40 LET G=A*B-B*D
50 IF G=0 THEN 120
60 PRINT "INTRODUCETI TERMENII LIBERI C, F"
70 INPUT C, F
80 LET X=(C*B-B*F)/G
90 LET Y=(A*F-C*D)/G
100 PRINT X, Y
110 STOP
120 PRINT "SISTEMUL NU ARE SOLUTIE UNICA"
130 END

```

Privind acest program se observă în primul rând că, în scrierea sa, sînt folosite numai litere mari. În al doilea rând, se constată că fiecare linie a programului începe cu un număr ; el este numărul de linie discutat în capitoul precedent. În al treilea rând, se observă că fiecare instrucțiune începe, după

numărul de linie, cu un cuvânt care determină tipul instrucțiunii. Astfel, în exemplul dat sînt folosite șapte instrucțiuni (REM, PRINT, INPUT, LET, IF, STOP și END).

O altă observație, care nu este evidentă din program, constă în aceea că, în BASIC spațiile nu au nici o semnificație, cu excepția celor din textele cuprinse între semnele apostrof (ca în liniile 10,20,60).

În continuare vor fi prezentate, pe rînd, instrucțiunile limbajului BASIC:

9.4.2. Comentarea unui program. Primele două instrucțiuni, din exemplul de mai sus, nu au nici un rol în rezolvarea sistemului de ecuații. Ele sînt scrise pentru a introduce comentarii într-un program BASIC.

Formatul instrucțiunii este :

nr. linie REM comentariu

Instrucțiunea REM poate apărea oriunde se dorește introducerea unui comentariu, în cadrul unui program. *Deoarece sistemul elimină spațiile din orice șir de caractere inclus între semnele apostrof (") este bine ca un comentariu să se includă între aceste semne.*

Exemplu :

130 REM "COMENTARIILE OCUPA MEMORIE UTILA"

9.4.3. Terminarea unui program. Pentru a opri execuția unui program, pot fi utilizate instrucțiunile END sau STOP.

Formatul instrucțiunii STOP este :

nr. linie STOP

La întâlnirea instrucțiunii STOP în program, sistemul va scrie mesajul :

STOP AT . NN]

unde NN este numărul de linie al instrucțiunii STOP care a produs oprirea execuției. Într-un program pot fi utilizate mai multe instrucțiuni STOP, în funcție de necesități. În exemplul din paragraful 9.4.1. instrucțiunea STOP are numărul de linie 110.

Formatul instrucțiunii END este,

nr. linie END

Spre deosebire de instrucțiunea STOP, într-un program trebuie să existe o singură instrucțiune END și să fie ultima instrucțiune din program (să aibă numărul de linie cel mai mare).

9.4.4. Instrucțiunea de atribuire (LET). Formatul instrucțiunii LET este următorul :

nr. linie variabilă = expresie

sau

nr. linie LET variabilă = expresie

unde :

— variabila poate fi o variabilă numerică sau variabilă șir, simplă sau indexată, sau funcția PUT ;

— expresia din membrul drept poate fi aritmetică sau expresie șir.

Execuția instrucțiunii constă în evaluarea expresiei din membrul drept și atribuirea valorii obținute, variabilei din membrul stâng.

Example :

```
5 H=H+1
10 A$="AP(ND)179"
20 C$=INKEY$
30 LET A=4.17+C
```

Observație :

O variabilă utilizată într-o expresie, fără a fi fost definită în prealabil, primește automat valoarea zero. Astfel, în instrucțiunea 30 de mai sus, dacă variabila C nu a fost definită, valoarea lui A după execuția instrucțiunii 30 va fi 4.17.

```
50 X=X+Y+3.5
60 LET W7=((W-X)/4.3)*SQR(Z(I)-A)/B
90 AS(7 TO 10)=CHR$(65)+„123“
```

Dacă expresia șir atribuită unei variabile șir, este mai lungă decât dimensiunea variabilei (stabilită prin DIM sau printr-o atribuire anterioară), atunci se atribuie variabilei șir doar prima parte din expresia șir (egală cu lungimea variabilei), restul expresiei șir ignorându-se. Dacă lungimea șirului din membrul drept este mai mică decât lungimea variabilei șir, restul de caractere se completează cu spații.

Example :

```
10 A$="12345678"
30 A$ (2 TO 5)="ABCDEFG"
```

În acest caz, A \$ va deveni "1ABCD678", deci, nu fost păstrate doar primele patru caractere din șirul "ABCDEFG". Dacă în locul instrucțiunii 20 de mai sus, se folosea instrucțiunea :

```
20 A$ (2 TO 5)="AB"
```

valoarea variabilei A \$ ar fi devenit "1AB"678"

```
30 B$=STR$(A+B-17)+„MEDIA“
40 J(1,INT(K+10))=COS(EXP(K+1))
```

9.4.5. Utilizarea variabilelor indexate (DIM)

Formatul instrucțiunii este :

nr. linie DIM tablou 1 (dimens), tablou 2 (dimens, dimens) ...
unde :

- tablou 1, tablou 2, sînt identificatori (nume) de tablou (vectori, matrici sau variabile șir)
- dimens reprezintă expresii ale căror valori definesc dimensiunile tablourilor ; este necesară îndeplinirea condiției

$$1 \leq \text{dimens} \leq 254$$

Tablourile numerice pot avea una sau două dimensiuni (vectori sau matrici)

Exemplu :

```
110 DIM A (50), B (M, N)
```

Instrucțiunea 110 declară două tablouri A și B, A fiind un vector cu 50 de elemente, iar B o matrice cu $M \times N$ elemente (M, N cunoscute).

Execuția instrucțiunii constă în alocarea memoriei pentru tablourile declarate și inițializarea tuturor elementelor ale fiecărui tablou numeric cu zero.

Tablourile de variabile șir trebuie declarate în instrucțiunea DIM, indicându-se numărul de variabile și lungimea variabilelor.

Exemplu :

```
10 DIM AS (15, 20), BS (50)
```

Instrucțiunea 10 declară un tablou A \$ format din 15 variabile șir a câte 20 de caractere fiecare și o variabilă șir B \$ cu lungimea de 50 de caractere. De remarcat faptul că, variabilele șir (exemplu B \$) nu trebuie declarate într-o instrucțiune DIM, ele putând fi utilizate direct în instrucțiuni de atribuire sau de intrare/ieșire. Tablourile șir trebuie declarate într-o instrucțiune DIM înainte de utilizare. Folosirea lor se face prin specificarea unei anumite variabile șir componente, și nu a întregului tablou. De exemplu, instrucțiunea :

```
20 A (2) = "1234"
```

este incorectă pentru că, A \$ a fost declarat tablou în instrucțiunea 10.

Corect ar fi :

```
20 AS (2) = "1234"
```

instrucțiunea în urma căreia, variabila a doua din tabloul A \$ primește valoarea "1234". Tot corectă este și instrucțiunea :

```
20 AS (2,10 TO 13) = "1234"
```

În cazul tablourilor (variabilelor) șir, execuția instrucțiunii DIM constă în alocarea de memorie, câte un octet pentru fiecare element (caracter), inițializat cu codul spațiului (A0H).

Tablourile numerice și cele șir pot fi declarate intercalat, în aceeași instrucțiune DIM.

Exemplu :

```
10 DIM A (10, 10), BS (100), C (10)
```

9.4.6. Exercițiul

Introduceți următoarea secvență de instrucțiuni :

```
10 LET AS = "X*+*Y"
```

```
20 LET AS (4) = CHR$ (162)
```

```
20 LET AS (2) = CHR$ (162)
```

```
40 PRINT AS
```

(162 este codul ghilimelelor)

De remarcat că secvența de mai sus nu este echivalentă cu :

```
10 LET A$="X" "+" "Y"
```

deoarece, în acest caz, membrul drept al instrucțiunii de atribuire este tratat ca o expresie de tip șir, evaluată în "XY". O altă modalitate corectă de a introduce ghilimele într-un șir este introducerea succesivă. Astfel instrucțiunea :

```
10 LET A$="X" "+" "Y"
```

este echivalentă cu secvența de mai sus.

2. Fie secvența de instrucțiuni :

```
10 LET A$="12345678"          40 PRINT A$
20 PRINT A$                  50 LET A$="1234"
30 LET A$="ABC"              60 PRINT A$
```

După execuția se va obține eroarea 28 în linia 50. Acest lucru este consecința faptului că în linia 30 are loc o redimensionare a șirului A\$, la lungimea de 3 caractere (de la 8 cite avea inițial). Pentru a evita această redimensionare, linia 30 se poate scrie 30 LET A\$(10)="ABC", șirul A\$ rămânând de lungime 8, prin completarea șirului "ABC" cu 5 spații. Deci, trebuie reținut faptul că în gestiunea șirurilor nu se folosesc doi indicatori de lungime : unul păstrând lungimea maximă, iar celălalt lungimea efectivă a șirului (sau lungimea maximă să fie pentru toate șirurile o constantă, exemplu 256 caractere).

3. Introducere programul :

```
5 DIM B$(8, 8)                40 PRINT B$(2)
10 LET B$(2)="12345678"      50 LET B$(2)="12345"
20 PRINT B$(2)              60 PRINT B$(2)
30 LET B$(2)="MPQ"          70 END
```

Execuția programului nu se va obține eroare, ca în cazul exercițiului precedent. Deci, în cazul tablourilor de șiruri, nu are loc redimensionarea unui șir component la o valoare mai mică, ci se păstrează dimensiunea (8) declarată în DIM, prin completarea automată cu spații. Pentru a nu se completa cu spații (păstrându-se totuși dimensiunea de 8 caractere) se poate utiliza specificarea cu TO. Astfel instrucțiunea 30 devine : 30 LET B\$(2,1 TO 3)="MPQ".

În acest caz, șirul tipărit de instrucțiunea 40 va fi : "MPQ45678".

4. În cazul utilizării unor tablouri mari se poate obține la execuție mesajul de eroare MEMORY FULL, indicând faptul că nu mai există memorie suficientă pentru alocarea tabloului. Trebuie să se aibă în vedere faptul că, pentru fiecare element al unei variabile tablou numerice, se stochează 4 octeți. Astfel, instrucțiunea 10 DIM A (11,11), B (16,16) va ocupa două tablouri numerice, primul ocupând aproximativ 0,5 Ko și al doilea 1 Ko. Ce acțiuni se pot indica la apariția acestui mesaj de eroare ?

9.4.7. **Instrucțiuni de intrare/ieșire.** Instrucțiunile de intrare/ieșire permit să se introducă date (valori ale variabilelor) pentru program și să tipărească rezultate sau mesaje către utilizator. În execuția acestor instrucțiuni sînt folosite în mod obișnuit tastatura și display-ul, însă pot fi utilizate oricare din perifericele sistemului, dacă se au în vedere facilitățile mai deosebite de intrare/ieșire materializate prin funcțiile GET și PUT sau subrutine în limbaj mașină, apelate din programul BASIC, cu instrucțiunea CALL.

Instrucțiunea INPUT.

Format :

nr. linie INPUT listă de variabile

Lista de variabile poate conține variabile simple și variabile indexate sau variabile șir, separate prin virgule.

Instrucțiunea INPUT este folosită pentru atribuirea de valori variabilelor din listă, valorile fiind introduse de la tastatură (consolă) în timpul execuției programului.

Atunci când se execută instrucțiunea INPUT, sistemul va tipări la consolă(:) indicând că așteaptă date de la consolă. Se vor introduce constantele dorite separate prin virgule, linia introdusă va fi luată în considerare de sistem la acționarea tastei RETURN. Dacă s-au introdus mai puține date decât variabile în instrucțiunea INPUT, sistemul va tipări din nou (:) și introducerea datelor va continua pînă ce toate variabilele au primit valori.

Dacă se introduc mai multe constante decât sînt necesare, constantele suplimentare se ignoră. În cazul în care se dorește oprirea execuției programului, în timp ce sistemul așteaptă date de la consolă, se acționează tasta CTRL și C simultan, iar sistemul va tipări READY.

Exemplu :

```
50 INPUT U, A, B, V (2).
```

La execuția instrucțiunii :50 sistemul va tipări (:) la consolă și va aștepta introducerea a patru constante. Dacă se introduce doar trei numere, sistemul va tipări iarăși (:) indicînd că așteaptă date. Astfel, la consolă va apărea :

```
: 4, 17.5, 9E10
: 10.3
```

Dacă este comisă o eroare în timpul introducerii de date, sistemul va tipări un mesaj de eroare, iar ultima linie de date va trebui reintrodusă. Astfel, dacă în exemplul de mai sus s-ar fi introdus :

```
: 4, 17M5, 9E10, 10.3
```

sistemul ar fi răspuns :

```
INPUT ERROR, TRY AGAIN
```

```
: (așteaptă noi date)
```

Introducerea datelor se repetă corect astfel :

```
: 4, 17.5, 9E10, 10.3
```

În cazul folosirii variabilelor șir, trebuie remarcat că, din linia cu constante șir corespunzătoare, introdusă de la consolă, sînt eliminate spațiile meineluse între ghilimele.

Exemple :

```
10 INPUT A$, B, C$ (2TO5);
```

```
(1 "SINT GATA", 73, ABCD)
```

Variabila A \$ a primit valoarea "SINT GATA", variabila B valoarea 73 iar variabila C\$ valoarea ABCD, pentru elementele 2 la 5. Textul SINT GATA a fost introdus între ghilimele, pentru a se păstra spațiul dintre cuvinte.

Instrucțiunile READ și DATA. Instrucțiunea READ are același efect cu INPUT, cu deosebirea că datele nu sînt introduse de la consolă, ci sînt citite dintr-un bloc de date, definit cu instrucțiunea DATA.

Formatul instrucțiunilor :

nr. linie READ listă de variabile

nr. linie DATA listă de constante.

Pentru a putea ține evidența constantelor citite în instrucțiunile DATA, interpretorul folosește un indicator la constanta ce urmează a fi citită din blocul de constante, care apar în instrucțiunile DATA, din program. În cazul în care nu se pot inițializa toate variabilele din instrucțiunea READ, din cauza epuizării constantelor din instrucțiunile DATA, sistemul va da un mesaj de eroare.

O instrucțiune DATA nu este asociată unei instrucțiuni READ, ci toate instrucțiunile DATA sînt tratate ca și cum ar forma un bloc de date.

Instrucțiunea DATA poate apărea oriunde, în cadrul programului.

Exemplu :

```
150 READ X, Y, Z
200 READ A
250 FOR I=1 TO 10
255 READ B (I)
260 NEXT I
. . . . .
400 DATA 4.2, 7.5, 25, -1, .1, .01, .001, -.1
450 DATA 2, 1, 7, 9, 1E7, 3.5
```

Primele trei constante sînt citite pentru X, Y și Z.

Valoarea -1 va fi atribuită lui A. Următoarele zece valori de la .1 la 3.5 vor fi atribuite elementelor vectorului B.

Instrucțiunea RESTORE

Format :

nr. linie RESTORE

Este folosită pentru a inițializa indicatorul din blocul DATA, pe prima instrucțiune DATA din program. Astfel, se permite reutilizarea datelor.

Exemplu :

```
20 FOR K=0 TO 10
30 READ B(K)
40 NEXT K
50 RESTORE
60 READ X, Y, Z
70 RESTORE
. . . . .
200 READ
500 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

Elementele vectorului B vor primi valorile 1,2,...., 10; variabilele X, Y, Z vor primi valorile 1, 2, 3, iar instrucțiunea 200READ va citi începînd cu 1.

În instrucțiunea READ pot fi folosite și variabile șir.

Exemplu :

```
10 READ AS, B, CS
. . . . .
100 DATA "ABC 123", 17.5, AB123
```

Variabila A\$ primește valoarea "ABC123", iar C\$ valoarea AB123.

Instrucțiunea PRINT

Formatul instrucțiunii este :

nr. linie PRINT listă

sau

nr. linie PRINT

lista poate conține : constante, variabile simple sau indexate, expresii sau șiruri.

Instrucțiunea PRINT este folosită pentru tipărirea valorilor curente ale expresiilor care apar în listă. În cazul când lista lipsește se va trece la începutul liniei următoare pe display.

Numerele reale sînt tipărite în următorul format :

- dacă numărul este negativ se tipărește semnul -
- dacă valoarea absolută a numărului N este cuprinsă între :

$$0.1 \leq N \leq 9999999,$$

atunci numărul este tipărit fie ca număr întreg (dacă valoarea sa este întreagă), fie ca număr zecimal, folosindu-se punctul ca separator al părții întregi de partea zecimală.

- dacă valoarea numărului nu se încadrează în intervalul de mai sus, atunci numărul este tipărit în notație științifică (exponențială) astfel :

$$[-] X.XXXXX E [-] YY$$

unde : [-] - indică partea opțională a reprezentării, dacă numărul este negativ se tipărește - ;

X - reprezintă o cifră a numărului ;

E - indică reprezentarea exponențială ;

Y - este cifră a exponentului.

Valoarea numărului va fi : $[-] X.XXXXX \times 10^{[-]YY}$

În funcție de separatorul folosit, între elementele din listă, se va face spațierea între valorile tipărite.

{Ca separatori pot fi folosiți "." sau " ; "}

Linia pe care se tipărește este împărțită în zone. Lungimea unei zone este de 15 caractere. Pentru un display cu lungimea liniei de 30 caractere vor exista două zone începînd cu pozițiile 0 și 15.

Exemple :

```
10 LET X=5
50 PRINT X, (X*2)+6, X*2
60 PRINT X+4, X-25, X-100
80 END
```


La display va apărea :

0...	15.....	30 ← poziții
5	1.000000E 06	
10	625	
-20	-95	

Când valoarea de tipărit depășește lungimea unei zone, elementul următor este tipărit la începutul primei zone libere.

```
10 LET X=25
20 PRINT "SQR(X)=", SQR(X)
30 END
```

Se va tipări :

0...	15...	30 ← poziții
SQR(X)=	5	

Folosirea ca separator a lui „;” conduce la tipărirea fără salt la următoarea zonă.

Dacă lista din instrucțiunea PRINT nu se termină cu „,” sau „;” atunci, la sfârșitul execuției instrucțiunii (după tipărirea tuturor elementelor din listă), se va trece la începutul liniei următoare. Terminarea listei din instrucțiunea PRINT, cu unul din separatorii „,” sau „;” este utilă în cazul tipăririi mai multor valori într-un ciclu, pentru a se face economie de spațiu de afișare.

Exemplu : Programul

```
10 FOR I=1 TO 10
20 PRINT I
30 NEXT I
40 END
```

Va tipări fiecare valoare la început de linie

```
1
2
3
. . .
10
```

Dacă se folosește separatorul „,”

```
10 FOR I=1 TO 10
20 PRINT I,
30 NEXT I
40 END
```

atunci se vor tipări cele două valori pe linie. Pentru a se putea tipări oriunde pe suprafața display-ului formată din 32 de linii a 30 de caractere (coloane) fiecare, se poate folosi în instrucțiunea PRINT funcția : AT(linie, coloană).

Exemplu :

```
10 LET AS="A="
20 PRINT AT (16,15); AS; 100
```

va tipări la mijlocul ecranului textul A=100.

Funcția AT este utilă în afișarea de valori sau texte, pe un desen realizat cu instrucțiunile grafice, sau pentru tipărirea într-o anumită zonă a ecranului, indiferent de poziția curentă. În esență urmează să se tipărească. Poziția (1, 1) este în colțul din stânga sus al ecranului, iar (32, 30) în dreapta, jos.

9.4.8. Exerciții

1. Instrucțiunea INPUT este deseori precedată de instrucțiunea PRINT, de exemplu:

```
10 PRINT "INTRODUCETI DATELE"
20 PRINT "X=";
30 INPUT X
. . .
```

Valoarea lui X va fi afișată pe același rând cu mesajul X=, deoarece instrucțiunea 20 PRINT se termină cu ;.

Similar, pentru introducerea de texte:

```
10 PRINT "DORITI O NOUĂ EXECUȚIE?";
20 INPUT R$
. . .
```

Răspunsul introdus fiind DA sau NU, în urma cărora în program se va executa o decizie.

2. Executați următorul program:

```
10 INPUT A$
20 PRINT A$; "="; VAL(A$)
30 GOTO 10
```

Introduceți instrucțiuni PRINT suplimentare astfel încât calculatorul să precizeze ce intenționează să facă și să ceară politicos datele de intrare. De remarcat că expresia introdusă spre evaluare (în A\$) poate conține orice fel de prelucrări numerice inclusiv variabile, a căror valoare trebuie introdusă în prealabil. De asemenea, șirul A\$ trebuie să fie alocat într-o instrucțiune DIM, iar în instrucțiunea 10 să se utilizeze notația (TO) pentru a împiedica realocarea

```
5 DIM A$ (10)
10 INPUT A$ (TO)
. . .
```

3. Cum se poate opri execuția programului din exercițiul precedent? (CTRL/C în INPUT sau CTRL între instrucțiunile, sau . . . RESET). Dacă execuția s-a oprit între instrucțiunile, cum se poate relua fără alterarea variabilelor?

4. Scrieți un program care calculează suma curentă a numerelor introduse.

Indicație: Se utilizează variabilele T pentru păstrarea totalului curent și V pentru numărul introdus. T se inițializează cu valoarea 0. Se citește un număr în V, se adună la T și se tipărește T. Se repetă cu introducerea noului număr.

5. Pentru a simula mișcarea unui simbol grafic* într-o anumită direcție se va afișa și se va șterge simbolul respectiv, progresivul cu este un pas în direcția respectivă. Se va utiliza în acest scop instrucțiunea PRINT cu funcția AT.

```
10 LET X=1
20 PRINT AT (5, X); "C";
30 PRINT AT (5, X); " ";
40 LET X=X+1
50 GOTO 20
```

Programul de mai sus are dezavantajul că, la terminarea liniei când X devine >30, se continuă incrementarea sa. Pentru a evita acest lucru se introduce instrucțiunea:

```
45 IF X>30 THEN 10
```

* Caracterul grafic este notat cu „C”

care, reîncepe execuția de la linia 10 dacă X a devenit mai mare decât 30. Un alt dezavantaj este timpul scurt în care simbolul apare pe ecran, datorită intervalului de timp scurt între afișare și ștergere. Pentru a mări această durată se introduce între liniile 20 și 30 un ciclu FOR fictiv :

```
22 FOR F=1 TO 10
25 NEXT F
```

Acest lucru însă, are dezavantajul că micșorează viteza globală de execuție a programului. Se recomandă următoarea soluție :

```
10 LET X=1
20 LET P=X
30 LET X=X+1
40 PRINT AT (5, P); " " : AT (5, X); "C" ;
50 IF X=20 THEN 10
60 GOTO 20
```

Programul folosește două variabile. X păstrează poziția curentă a caracterului, iar P poziția anterioară, care va fi ștersă. Simbolul apărând mereu pe ecran, crește impresia de viteză. De remarcat că, în aceeași instrucțiune PRINT poate apărea de două ori specificația AT pentru poziționare. O altă soluție constă în scrierea caracterului precedat de spațiu, PRINT AT (5, X); "C", astfel :

```
10 X=1
20 PRINT AT (5, X); "C" ;
30 X=X+1
40 IF X>20 THEN 10
50 GO TO 20
```

4) Imbunătățire adusă programului se referă la utilizarea instrucțiunii FOR pentru ciclare

```
10 FOR X=1 TO 20
20 PRINT AT (5, X); "C" ;
30 NEXT X
40 GOTO 10
```

Pentru mișcarea unui pătrățel (1/4 dintr-un caracter), pot fi utilizate similar instrucțiunile PLOT și UNPLOT. În programele de mai sus, în poziția 30 a liniei, va rămâne caracterul fără a fi șters. Pentru a-l șterge, se introduce (în ultimul exemplu) : 35 PRINT AT (5, 30); " " ;

Cum puteți descrie mișcarea simultană a unui multor caractere (figuri) ?

6. Funcția STR\$ este foarte utilă, dar deseori neglijată. Așa cum s-a arătat, ea este utilizată pentru conversia unei valori numerice într-un șir de caractere, identic cu cel obținut prin tipărirea sa cu PRINT. Încecați următorul program :

```
10 PRINT 2, STR$ (2)
20 PRINT 1/3, STR$ (1/3)
30 PRINT 0E15, STR$ (0E15)
```

Se observă că, valorile afișate sînt identice. Utilizarea principală a funcției STR\$ este în formatarea rezultatelor afișate. Se vor indica câteva exemple.

a) Așezarea punctului zecimal în poziția dorită. Executați programul :

```
10 LET A=RND (0)*10000
20 PRINT A
30 LET A=A*10
40 GO TO 20
```

Se obține următoarea secvență :

```
0.152587
1.52587
15.2587
152.587,
1525.87
15258.7
152587
```

Valorile sînt mai ușor de comparat dacă le putem alinia pe verticală la nivelul punctului zecimal, astfel :

```
0.152587
1.52587
15.2587
152.587
1525.87
15258.7
152587
```

Pentru aceasta se modifică programul de mai sus folosind funcția STR\$.

```

5 I=6
10 A=RND (0)*10000
20 N=LEN (STR$ (INT (A)))
22 PRINT AT (I, 15-N) : A
30 A=A*10
35 I=I+1
40 GO TO 20

```

Variabila I este utilizată pentru a indica linia în care se tipărește cu funcția AT în instrucțiunea 22 PRINT.

Expresia LEN (STR\$ (INT (A))) calculează lungimea părții întregi a numărului. Astfel, dacă $A=15,2587$, atunci STR (INT (A))="15"$, și cu lungimea 2, deci tipărirea se va începe din coloana $15-2=13$, încl. Iată numerele vor avea punctul în coloana 15 (exceptând cele în notatie științifică). Execuția programului va fi oprită de la tastatură, acționând CTRL.

b) Tipărirea cu un număr dorit de cifre la partea zecimală. Se vor tipări, de exemplu, numerele cu trei cifre după punctul zecimal.

```

5 I=0
10 DIM B$(15)
15 A=RND (0)*100
20 M=LEN (STR$ (A))
25 N=LEN (STR$ (INT (A)))
30 B$(0)=STR$ (A)+"0000"
35 IF M <> N THEN 45
40 B$(N+1)="."
45 PRINT AT (I, 15-C) : B$(0:N+1)
50 A=A*10
55 I=I+1
60 GO TO 20

```

Variabila I este utilizată în același scop ca la exemplul precedent (indică numărul liniei). M conține lungimea totală a numărului, iar N numărul de cifre al părții întregi (ca și în exemplul precedent). În variabila B\$ se „assemblează” numărul urmat de eventuale zerouri. Dacă numărul este întreg (test realizat în instrucțiunea 35 IF), se adaugă și punctul zecimal (în locul unuia din cele patru zerouri).

c) Economisirea spațiului de memorie și definierea de funcții. Descrie este utilă manipularea numerelor tratate ca șiruri de caractere, în loc de a le prelucra numeric, iar în final, evaluarea lor cu funcția VAL. Astfel, se pot stoca numerele sub formă de șiruri, cu ajutorul funcției STR\$:

```
LET AS=STR$ (1024)
```

apoi, se revine la valoarea numerică cu VAL : PRINT VAL (AS).

În acest caz însă, memorarea lui 1024 cu șir în variabila AS, consumă mai multă memorie decât păstrarea lui ca număr. Dacă se aplică funcția VAL unei expresii ca "ATN (X)*4", variabila X fiind definită în prealabil, evaluarea va fi corectă, deci, o expresie care poate conține variabile, se definește o singură dată în program și poate fi evaluată pentru diverse valori ale variabilelor, cu ajutorul lui VAL (similar cu instrucțiunea DEFTN din unele versiuni BASIC). De exemplu, pentru generarea unui număr aleator între 1 și 10 se va defini la începutul programului șirul :

```
AS="INT (RND (0)*10+1)"
```

și de fiecare dată (când se dorește un număr, se va obține prin utilizarea funcției VAL (AS), $N=VAL (AS)$).

9.4.9. Instrucțiuni de control (transfer necondiționat, condiționat și ciclare)

În mod normal, execuția instrucțiunilor unui program este secvențială. Totuși este necesar uneori repetarea de mai multe ori a execuției unor instrucțiuni (fără să fie scrise repetat) sau, în funcție de valoarea unor date, să nu se execute instrucțiunea următoare, ci să se treacă la execuția alteia, dintr-o altă zonă a programului.

Instrucțiunile care modifică execuția secvențială a programului sînt instrucțiunile de control. În BASIC ele sînt următoarele :

— instrucțiunea GOTO, care realizează transferul necondiționat (independent de date) ;

- instrucțiunile ON și IF pentru transfer condiționat;
- instrucțiunea de ciclare FOR.

Instrucțiuni de transfer necondiționat. Pentru transferul necondiționat se folosește instrucțiunea GOTO.

Format: nr. linie GOTO n

Execuția instrucțiunii constă în transferul controlului la instrucțiunea cu numărul de linie n. Saltul se poate face la orice instrucțiune din program.

Exemplu :

```

10 LET A=SQR(X↑2+Y↑2)
.
.
50 GOTO 100
.
.
70 GOTO 10
.
.
100 PRINT A

```

Instrucțiuni de transfer condiționat. Pentru realizarea transferului condiționat sînt disponibile două instrucțiuni: instrucțiunea IF și instrucțiunea ON.

Formatul instrucțiunii IF este:

nr. linie IF exp1 relație exp2 THEN n1

unde: — exp1 și exp2 sînt expresii numerice sau șiruri,
 — n1 este un număr de linie,
 — relație poate fi:

egalitate	=	
mai mare	>	
mai mic	<	
mai mare sau egal	>= sau =>	
mai mic sau egal	<= sau =<	
diferit	<> sau ><	

Dacă relația dintre exp1 și exp2 este adevărată se va trece la instrucțiunea cu numărul de linie n1, dacă nu, execuția va continua cu instrucțiunea care urmează după IF.

Exemplu :

```

100 IF X+Y=SQR(X) THEN 75

```

În cazul comparației șirurilor de caractere, ambele expresii trebuie să fie de tip șir. Șirurile sînt egale dacă au aceeași lungime și conțin aceleași caractere.

Exemplu :

```

10 LET A$="ABC"
20 IF A$="ABC" THEN 100

```

După execuția instrucțiunii 20 se va sări la execuția instrucțiunii 100.

Un șir se consideră mai mic decît altul, dacă are lungimea mai mică sau, în cazul lungimilor egale, dacă în ordonarea lexicografică, primul șir precede pe al doilea (cifrele și semnele speciale preced literele — conform cu codurile ASCII).

Exemplu : "ABC" este mai mic decît "BCD".

Următoarea secvență de program, testează dacă de la consolă s-a acționat caracterul "A", dacă da se continuă execuția, dacă nu, tipărește caracterul citit (sau șirul nul, în cazul când nu s-a acționat nici o tastă) și oprește execuția :

```
10 LET A$=INKEY$
20 IF A$="A" THEN 50
30 PRINT A$
40 STOP
50 . . .
```

Programul următor scrie la mijlocul ecranului textul ABC și îl șterge la acționarea alternativă a unor taste :

```
10 IF INKEY$="" THEN 10
20 PRINT AT (16, 15) "ABC"
30 IF INKEY$="" THEN 30
40 PRINT AT (16, 15) " "
50 GOTO 10
```

Formatul instrucțiunii ON este :

nr. linie ON expresie GOTO listă

sau

nr. linie ON expresie GOSUB listă

unde listă reprezintă un șir de numere de linie.

Fie N partea întreagă a valorii expresiei specificate în instrucțiunea ON. Execuția instrucțiunii constă în saltul la instrucțiunea cu al N-lea număr de linie din listă. În cazul lui ON-GOSUB, fiecare număr de linie, din listă reprezintă startul unei subrutine din program. Dacă N nu se încadrează între :

$1 \leq N \leq$ numărul de elemente din listă,

atunci execuția continuă cu instrucțiunea care urmează după ON.

Exemple :

```
10 ON M-5 GOTO 100, 500, 90, 75, 650
```

M-5 trebuie să aibă o valoare între 1 și 5 altfel, instrucțiunea nu are nici un efect.

```
20 ON X GOSUB 1000, 2000, 3000
```

În funcție de valoarea lui X se execută una din cele trei subrutine care încep la liniile 1000, 2000 sau 3000.

Instrucțiuni de ciclare. Instrucțiunile de ciclare sînt folosite pentru execuția repetată a unor instrucțiuni din program (numite cicluri sau bucle în program).

Pentru acest lucru se puteau utiliza instrucțiunile de transfer. Introducerea unor instrucțiuni speciale de ciclare s-a făcut în scopul simplificării muncii programatorului.

În BASIC, pentru realizarea ciclurilor, se utilizează instrucțiunile FOR și NEXT.

Formatul instrucțiunilor :

nr. linie FOR variabilă = exp1 TO exp2 STEP exp3

...

(instrucțiuni ce formează bucla)

...

nr. linie NEXT variabilă

unde: — variabilă reprezintă o variabilă simplă care este folosită pentru controlul numărului de repetări ale buclei. Bucla constă din instrucțiunile care se găsesc între instrucțiunea FOR și instrucțiunea NEXT, cu aceeași variabilă.

— exp1, exp2 și exp3 sînt expresii.

Semnificația acestor expresii este următoarea :

exp1 — este valoarea inițială care se atribuie variabilei din instrucțiunea FOR,

exp2 — este valoarea finală a variabilei,

exp3 — este incrementul (pasul) care se adună la valoarea variabilei FOR, de fiecare dată, cînd se execută instrucțiunile din buclă.

Observație :

Este posibil ca exp3 să nu se specifice, instrucțiunea FOR avînd forma :

nr. linie FOR variabilă = exp1 TO exp2

În acest caz valoarea pasului (exp3) se consideră 1. Bucla se execută atîta timp cît valoarea variabilei FOR este mai mică sau egală cu exp2.

Execuția instrucțiunii NEXT constă în adunarea valorii pasului (exp3) la valoarea variabilei FOR și testul dacă noua valoare a variabilei nu depășește valoarea limită (exp2). Dacă valoarea limită este depășită execuția buclei ia sfîrșit, următoarea instrucțiune executată fiind cea de după NEXT.

În interiorul unei bucle FOR pot exista alte bucle cu condiția ca buclele să nu se intersecteze.

Exemplu corect

```
10 FOR X=
  50 FOR Y=
    100 FOR Z=
      .
      .
    170 NEXT Z
  200 NEXT Y
  700 NEXT X
```

Exemplu incorect

```
10 FOR X=
  70 FOR Y=
  120 NEXT X
  300 NEXT Y
```

Sînt permise maximum trei bucle FOR incluse una în alta.

Exemplu :

```
100 FOR I=1 TO 3
  120 FOR J=1 TO 20 STEP 1
  130 READ B (I, J)
  140 NEXT J
  150 NEXT I
```

Instrucțiunea 130 READ se va executa de $20 \times 3 = 60$ de ori.

9.4.10. Exerciții

1. Programul următor afișează setul complet de caractere.

```
10 FOR N=0 TO 127          30 NEXT N
20 PRINT CHR$(N)          40 STOP
```

Să se modifice programul, utilizând funcția AT în PRINT și afișând numai cifre, pentru a se obține un ceas electronic. (Temporizarea se poate realiza cu un ciclu FOR).

2. Pentru calculul sumei a N numere reale se poate utiliza programul :

```
10 PRINT "N=" ;
15 INPUT N                35 S=S+A
20 C=1                    40 C=C+1
25 S=0                    50 IF C<=N THEN 30
30 INPUT A                60 PRINT S
```

Să se descrie același algoritm de calcul folosind instrucțiunile de ciclare FOR/NEXT cu variabila contor C.

3. Următorul program afișează numărul cu valoarea maximă dintr-un șir de 5 numere introduse :

```
10 INPUT M                50 M=A
20 FOR I=2 TO 5          60 NEXT I
30 INPUT A                70 PRINT "MAXIMUL=" ; M
40 IF M>=A THEN 60
```

Programul poate fi utilizat pentru aflarea maximumului dintr-un șir de 10 sau 100 de valori dacă modificăm limita 5 din instrucțiunea 20 FOR. Ce trebuie modificat în program pentru a calcula minimumul.

4. Să se modifice programul de calcul al sumei (din exercițiul 2) astfel încât să calculeze produsul numerelor nenule din cele N numere (pentru simplificarea se va folosi ciclul FOR/NEXT).

5. Pentru calculul factorialului dintr-un număr natural dat poate fi utilizat programul :

```
10 PRINT "INTRODUCETI N=" ;      40 P=P*I
20 INPUT N                        50 NEXT I
25 P=1                             60 PRINT "N!=" ; P
30 FOR I=2 TO N
```

Să se găsească o altă soluție pentru această problemă.

6. Executați programul :

```
10 FOR N=10 TO 1 STEP-1
20 PRINT N
30 NEXT N
```

Transformați acest program în unul ce nu conține ciclul FOR-NEXT (procedând invers exercițiului 2).

7. Fie dat un vector A de zece elemente. Programul de mai jos, ordonează crescător elementele acestui vector, deci, în final va fi satisfăcută relația $A(1) \leq A(2) \leq \dots \leq A(10)$.

```
10 FOR I=1 TO 10
15 READ A(I)
20 NEXT I
25 FOR I=1 TO 9
30 FOR J=1 TO 10-I
```



```

40 IF A (J) <= A (J+1) THEN 70
50 T=A (J)
55 A (J)=A (J+1)
60 A (J+1)=T
70 NEXT J
80 NEXT I
85 REM "AFISAREA REZULTATU-
LUI"
90 FOR I=1 TO 10
100 PRINT A (I).
110 NEXT I
120 END

```

De remarcat că, aceeași variabilă I poate apărea în mai multe cicluri FOR-NEXT. Dacă conflict, dacă ciclurile respective nu se intersectează (nu au instrucțiuni comune). De ce a fost necesară utilizarea variabilei T în instrucțiunile 50 și 60 ?

Cum poate fi accelerat acest algoritm ? Să se indice un alt algoritm (program) pentru aceeași problemă ? Ciclurile (10—20) pentru citire și (90—110) pentru tipărire pot fi înlocuite prin câte o singură instrucțiune matriceală (vezi capitolul : Instrucțiuni de calcul cu matrice). Să se modifice programul astfel încât să ordoneze un vector A cu N elemente, $2 \leq N \leq 254$.

8. Similar cu exercițiul 7, se prezintă un program ce ordonează crescător (sortează alfabetic) o listă de nume, care sînt depuse într-o matrice de șiruri de caractere. Programul se deosebește de cel precedent prin faptul că utilizează instrucțiunea IF pentru ciclare, iar ordonarea se face descrescător, însă, afișarea făcîndu-se începînd cu ultimul șir către primul (maxim), rezultatele vor fi afișate în ordine alfabetică (crescătoare).

```

10 REM "SORTARE ALFABETICĂ"
20 DIM WS (5, 10)
30 LET U=0
40 LET G=5
50 FOR A=1 TO 5
60 INPUT WS (A)
70 PRINT WS (A)
80 NEXT A
85 PRINT
90 LET Z=1
100 B=Z+1
110 IF B>G THEN 190
120 IF WS (B)>WS (Z) THEN 150
130 Z=Z+1
140 GOTO 100
150 Q=WS (Z)
160 WS (Z)=WS (B)
170 WS (B)=Q
180 GOTO 130
190 PRINT WS (G)
200 G=G-1
210 IF G>0 THEN 100
220 END

```

Programul folosește tabloul WS (5, 10) pentru păstrarea a 5 șiruri de maximum 10 caractere fiecare. Pentru extinderea la un număr mai mare de șiruri, trebuie modificată instrucțiunile : 20 DIM, 40 LET și 50 FOR, scriîndu-se în locul cifrei 5 un număr dorit (maximum 254). Se observă că, afișarea rezultatelor se face rînd pe rînd, imediat după obținerea, pe ultima poziție în WS, a șirului mai mic. Cum poate fi modificat programul pentru a afișa rezultatul, din WS (5, 10), după ordonarea sa integrală (ca în exercițiul precedent) ?

9.4.11. Utilizarea subrutinelor. Cînd este necesară efectuarea de mai multe ori într-un program a aceluiași calcul (instrucțiunii) pentru date eventual diferite, se poate folosi conceptul de subrutină. Utilizarea subrutinelor conduce la o diminuare a dimensiunii programului, deoarece prelucrările se descriu o singură dată (în cadrul subrutinei) și pot fi executate de cîte ori este nevoie, prin apelarea subrutinei, eventual cu alte date. Folosirea funcțiilor (SIN, COS,

LOG, GET, PUT etc.) în alcătuirea expresiilor, constituie un exemplu în acest sens, deoarece funcțiile sînt definite o singură dată, în interpretor, și sînt apelate cu diferite argumente de către programator.

Pentru definirea unor subrutine mai puternice (ce descriu prelucrări complexe asupra datelor), în limbajul BASIC sînt disponibile instrucțiunile GOSUB, RETURN și CALL care vor fi descrise în continuare.

Instrucțiunile GOSUB și RETURN. Aceste instrucțiuni permit utilizarea subrutinelor în BASIC.

Atunci cînd o succesiune dată de instrucțiuni, apare de mai multe ori într-un program, în locuri diferite, ea poate fi scrisă ca subrutină.

GOSUB

Format :

nr. linie GOSUB n1

Execuția instrucțiunii constă în transferul controlului la instrucțiunea cu numărul n1.

Instrucțiunea n1 poate fi oricare dintre instrucțiunile ce alcătuiesc o subrutină (în general prima).

RETURN

Format :

nr. linie RETURN

Execuția instrucțiunii constă în transferul controlului la instrucțiunea ce urmează după instrucțiunea GOSUB, care a apelat subrutina.

Exemplu :

100 LET X=5	500 Y=3*X
110 GOSUB 500	510 LET Z=1.2*EXP(Y)
120 X=7	520 LET Y=SQR(Z+2)
130 GOSUB 500	530 IF Y<100 THEN 550
140 X=11	540 RETURN
150 GOSUB 500	550 PRINT X, Y
160 STOP	560 RETURN

Variabilele în BASIC sînt globale, deci cele utilizate într-o subrutină pot fi apelate și în altă parte a programului.

Apelarea subrutinelor scrise în limbaj mașină. Pentru a apela, dintr-un program scris în limbaj BASIC, subrutine scrise în limbaj mașină, se utilizează instrucțiunea CALL.

Formatul instrucțiunii este următorul :

nr. linie CALL (M, P1, P2, ..., PN)

unde :

- M este numărul subrutinei (de la 0 la 254)
- P1, ..., PN sînt parametrii (constante, variabile sau expresii).

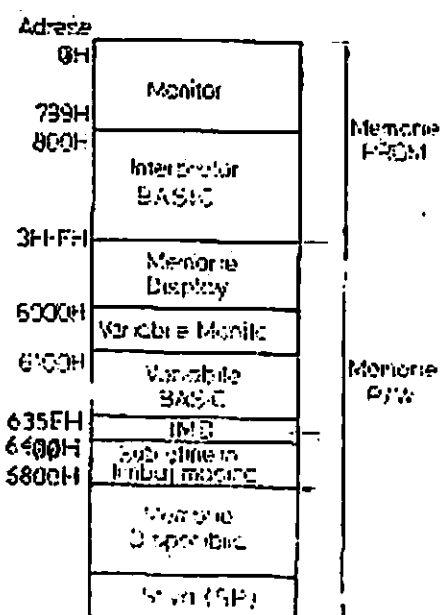


Fig. 9.1. Harta ocupării memoriei.

În ordinea: adresă de întoarcere în programul BASIC, adresa P1, adresa P2, ..., adresa PN.

Deci subrutinele în limbaj mașină, în cazul în care folosesc parametri (primesc și/sau returnează), vor trebui să facă uz de instrucțiunii POP, pentru obținerea adreselor parametrilor și să se termine cu instrucțiunea RET.

Observații:

1. Subrutinele în limbaj mașină trebuie să descarce din stivă adresele tuturor parametrilor înainte de terminare (RET), pentru ca adresa luată din stivă, la execuția instrucțiunii RET, să fie cea depusă de interpretor. Interpretorul nu face nici un control în acest sens.

2. Parametrii au valori reale, reprezentate pe patru octeți în formatul cu virgulă mobilă. În subrutina scrisă în limbaj mașină, trebuia să se țină seama de această reprezentare, pentru ca prelucrarea să fie corectă.

3. În cazul când parametrii sînt nume de tablou (vector sau matrice) elementele tabloului sînt depuse în memorie liniar, pe coloane (în cazul matricilor), fiecare element ocupînd patru octeți. La subrutină se transmite adresa primului element de pe prima coloană. Știînd acest lucru, se poate avea acces în cadrul subrutinei la oricare element al tabloului.

4. Parametrii variabile simple, nedefinite în prealabil (care vor primi valori rezultate din subrutină), nu trebuie să fie intercalați cu parametrii expresii.

Exemplu:

În continuare se arată cum trebuie actualizat indicatorul la memoria disponibilă (IMD)* și definită tabela de legături, pentru utilizarea a trei subrutine cu numerele 1, 2 și 4.

* IMD este inițializat cu valoarea 6800H, deci indiferent dacă se utilizează subrutine, se rezervă o zonă de 1Kb (400H) pentru subrutine.

Legătura dintre numărul subrutinei și adresa din memorie, unde se află subrutina, se stabilește prin intermediul unei table care se află la o adresă prestabilită (fig. 9.1, harta ocupării memoriei). O intrare în tabelă conține trei octeți. În primul octet se află numărul rutinei iar în următorii doi adresa. Sfirșitul tablei este detectat prin înțilnirea unei intrări cu numărul 255. În cazul cînd nu se folosesc subrutine în limbaj mașină, apelate dintr-un program BASIC, tabela se reduce la un singur octet cu valoarea 255. În acest caz, inițializarea tablei (octetului) se face automat, de către interpretorul BASIC, la lansarea sa în execuție, prin comenzile B sau G0800 (vezi paragraful 9.3.1).

Transferul parametrilor se asigură prin referință. Adresele parametrilor (pe doi octeți) sînt depuse în stivă, indicată de registrul SP.

	ORG	635EH	: adresa lui IMD *
IMD :	DW	MDISP	: actualizare IMD
TSUB :	DB	1	: intrare pentru
	DW	SUB1	: subrutina nr. 1
	DB	2	
	DW	SUB2	: adresa subrutinei 2
	DB	4	: intrare pentru
	DW	SUB3	: subrutina nr. 4
	DB	255	: sfârșit tabelă
SUB1 :	.	.	: cod pentru
	.	.	: subrutina nr. 1
	RET	.	
SUB2 :	.	.	: cod pentru
	.	.	
	RET	.	: subrutina nr. 2
SUB3 :	.	.	: cod pentru
	.	.	
	.	.	: subrutina nr. 4
	RET	.	
MDISP	EQU	\$: început memorie
	END	.	: disponibilită

Codul rezultat în urma asamblării programului de mai sus, scris în limbaj de asamblare, va trebui să fie introdus la adresa specificată, cu ajutorul comentariilor MONITOR, înainte de lansarea în execuție a interpretorului BASIC. După introducerea subrutinelor în limbaj mașină interpretorul BASIC se lansează în execuție cu comanda GØ315 (pentru a nu se face inițializarea implicită a lui IMD, ca în cazul absenței subrutinelor).

9.4.12. Exerciții

1. Unele aplicații necesită accesul în memoria calculatorului pentru scrierea sau citirea valorii unui octet de la o anumită adresă. După cum se știe unitatea de memorie adresabilă este octetul, iar procesorul poate adresa 65536 octeți (64 Ko). Pentru realizarea acestor funcții, unele versiuni ale limbajului BASIC conțin instrucțiuni speciale, de exemplu: PEEK A — funcție a cărei valoare este conținutul octetului de la adresa A, cu $0 \leq A \leq 65535$; iar POKE A, V este o instrucțiune ce înscrie în octetul de la adresa A valoarea V, cu $0 \leq V \leq 255$. Se vor simula aceste două instrucțiuni utilizând două subrutine în limbaj de asamblare apelate cu CALL din BASIC. Astfel, pentru PEEK A, se definește subrutina numărul 1, cu parametrii A și B; A va conține adresa iar în B se va depune valoarea octetului de la adresa A. Pentru POKE A, V se definește a doua subrutină cu parametrii A și V care va înscrie la adresa A valoarea V ($0 \leq V \leq 255$).

Subrutinele se vor apela, dintr-un program scris în BASIC, prin CALL (1, A, B) pentru PEEK A și CALL (2, A, V) pentru POKE A, V.

Subrutinele în limbaj de asamblare sînt precedate de tabela de legătură :

	ORG	635FH	: adresa IMD
IMD :	DW	SFSUB	: actualizare IMD
	DB	1	: tabela de legătură
	DW	PEEK	: adresa subrutinei PEEK
	DB	2	: intrare pentru a
	DW	POKE	: 2-a subrutină
	DB	255	: sfârșitul tabelii
PEEK :	POP	B	: în reg. B, C adresa celui
			: de-al 2-lea parametru

* În cazul când subrutinele nu depășesc 1Ko, se poate lăsa valoarea implicită pentru IMD (6300H), începîndu-se cu ORG 6400H și definirea tabelii TSUB

	POP	D	: în D, E adresa variabilei : A (primul parametru)
	PUSH	B	: salvare reg. B, C
	CALL	ADRDE	: în reg. D, E adresa de la care citim
	POP	B	: în reg. B, C adresa la care depunem : valoarea citită
; valoarea va trebui memorată în formatul cu virgulă mobilă			
	LDAX	D	: în reg. A octetul citit
	MVI	L, 8	: normalizare
NORM :	ORA	A	
	JZ	NULFL	: valoarea este zero
	JM	GATA	
	BAL		
	DCR	L	
	JMP	NORM	
NULFL :	MVI	L, 40H	: exponentul lui zero
GATA :	STAX	B	: scrierea rezultatului în
	INX	B	: virgulă mobilă
	XRA	A	
	STAX	B	
	INX	B	
	STAX	B	
	MOV	A, L	: exponentul
	INX	B	
	STAX	B	
	RET		
POKE :			: subrutină care scrie valoarea celui de-al : doilea parametru în adresa dată de primul
	POP	B	: în B, C adresa valorii
	LDAX	B	: conversie în întreg
	ORA	A	
	JZ	CONT	
	MOV	D, A	: salvare în reg. D
	INX	B	
	INX	B	
	INX	B	
	LDAX	B	: cifre exponent
	ANI	0FH	
	MOV	E, A	
	MOV	A, D	
	RLC		
	DCR	E	
	JNZ	1-2	
; în reg. A se află valoarea $0 \leq V \leq 255$, convertită în întreg			
	POP	D	: adresa primului parametru
	MOV	B, A	: salvare valoare
	CALL	ADRDE	
	MOV	A, B	: depunerea valorii la
	STAX	D	: adresa din reg. D, E
	RET		
ADRDE :			: subrutină ce primește în : D, E adresa unui număr în : virgulă mobilă și întoarce în : D, E valoarea sa (ca număr întreg)
	LDAX	D	: în reg. A primul octet al
	ORA	A	: mântisei. Este 0 ?
	JZ	ZERO	: numărul este zero

	MOV	H, A	
	INX	D	
	LDAX	D	
	MOV	L, A	: în (H, L) primii 2 octeți ai : numărului
	INX	D	
	INX	D	
	LDAX	D	: în reg. A exponentul
	ANI	1FH	: se păstrează ultimii
	MOV	C, A	: 5 biți, salvați în reg. C
	LXI	D, 0	: D, E=0
CONV:	DAD	H	
	XCHG		
	ADC	HL, HL	: HL=HL+HL+CY (Z80)
	DCR	C	
	XCHG		
	RZ		
	JMP	CONV	
ZERO:	LXI	D, 0	
	HET		
SESUB	EQU	\$	
	END		

Codul obiect rezultat în urma asamblării subrutinelor, se va depune în memoria calculatorului (cu comanda Substitute (S) a monitorului). Apoi, se lansează în execuție interpretorul BASIC, cu comanda GO800. În continuare, se poate introduce un program BASIC care utilizează subrutinele în limbaj mașină:

```

10 PRINT "INTRODUCETI ADRESA A=" ;
20 INPUT A
25 PRINT "INTRODUCETI VALOAREA V=" ;
30 INPUT V
40 REM "IN OCTETUL DE LA ADRESA A"
45 REM "VOM DEPUNE VALOAREA V"
50 CALL (2, A, V)
60 REM "PENTRU TEST VOM CITI"
65 REM "VALOAREA SCRISA SI O VOM AFISA"
70 CALL (1, A, C)
80 PRINT C
90 GOTO 10

```

Programul următor realizează aceeași funcție ca și comanda Fill a monitorului

```

10 INPUT A1, A2, V
20 FOR A=A1 TO A2
30 CALL (2, A, V)
40 NEXT A
50 END

```

Pentru ca programele să funcționeze corect, trebuie ca adresele introduse să corespundă zonei de memorie R/W și să nu altereze programul BASIC (să nu se autodistrugă).

În cazul utilizării subrutinelor în limbaj mașină, comanda de ștergere a programului : SCR, va șterge și subrutinele în limbaj mașină. Pentru a nu introduce din nou subrutinele mașină de la tastatură, ele pot fi salvate pe casetă, cu comanda SAVE, imediat după ce au fost introduse și s-a intrat în BASIC. În

acest caz, ele vor putea fi încărcate de pe casetă cu comanda LOAD înainte de introducerea unui program BASIC, de la tastatură, care le utilizează.

2. Calculatorul dispune de un difuzor care este utilizat ca „martor sonor” la apăsarea unei taste. Dacă se asociază tastelor sunete corespunzătoare notelor muzicale, se va transforma calculatorul într-un instrument muzical.

Difuzorul este conectat prin portul de ieşire 22H (31 zecimal) pe bitul 3. Pentru a obţine frecvenţa unei note dorite, se va transmite pe acest port şiruri de 1 şi 0 de durate corespunzătoare. Pentru a se transmite o valoare pe un port de ieşire poate fi utilizată funcţia PUT (nr. port) din BASIC, de exemplu :

```
10 PUT (34)=0
20 PUT (34)=8 (1 în bitul 3)
30 GOTO 10
```

Pentru a obţine frecvenţe mai mari, trebuie să se utilizeze o subrutină în limbaj de asamblare. Astfel, se va defini o subrutină în limbaj de asamblare care va primi ca parametrii : frecvenţa notei dorite şi durata şi va genera un sunet corespunzător. Subrutina se va apela cu instrucţiunea CALL (I, F, D), după ce, în F s-a încărcat frecvenţa iar în variabila D durata. Valorile lui F şi D depind de modul în care s-a scris subrutina de generare şi de frecvenţa impulsurilor de ceas ale calculatorului.

Se va folosi următoarea subrutină în limbaj maşină :

```

                ORG      635EH
                DW       MDISP      ; adresa memoriei libere
                DB       1          ; Tabela de legătură
                DW       SUNET      ; adresa subrutinei
                DB       255        ; sfârşitul tabelului
SUNET:         POP      H          ; adresa celui de al doilea parametru (P2)
                CALL    CVINT      ; conversie în întreg
                MOV     E, A        ; valoarea P2 (durata) o păstrăm în reg. A
                POP     H          ; adresa P1 (frecvenţa)
                CALL    CVINT      ; conversie în întreg
                MOV     B, A        ; salvare P1 în reg. B
LOOP1:         MOV     D, B        ; în D frecvenţa
LOOP2:         MVI    A, 8         ; transmite 1 pe bitul 3
                OUT    22H        ; al portului 22H
                DCR    D          ;
                JNZ    LOOP2
                MOV     D, B        ; în D frecvenţa
LOOP3:         XRA    A          ; transmite 0 pe bitul 3
                OUT    22H        ; al portului 22H
                DCR    D          ;
                JNZ    LOOP3
                DCR    E          ; decrementează durata
                JNZ    LOOP1
                RET

CVINT:         ; subrutină ce primeşte în reg. H, I, adresa unui număr reprezentat
                ; în formatul virgulă mobilă pe 4 octeţi şi întoarce
                ; valoarea sa (întregă) în reg. A (se presupune că valoarea
                ; numărului este cuprinsă între 0 şi 255).
                MOV     A, M        ; primul octet al numărului
                ORA    A          ; este 0?
                RZ
                MOV     D, A        ; salvarea primului octet
                INX    H          ; în reg. D
```

	INX	H	
	INX	H	
	MOV	A, M	: exponențial
	ANI	0FH	: păstrează ultimii 4 biți
	MOV	C, A	
	MOV	A, D	: conversie în întreg
	RLC		: (aliniere la dreapta)
	DCR	C	
	JNZ	S-2	
	RET		
NDISP:	EQU	5	
	END		

După introducerea codului subrutinei cu comanda Substituto a Monitorului, se va lansa în execuție interpretorul BASIC cu comanda GOSUB. Subrutina se va apela, așa cum s-a arătat, cu instrucțiunea CALL din BASIC (de exemplu CALL (1, F, D)).

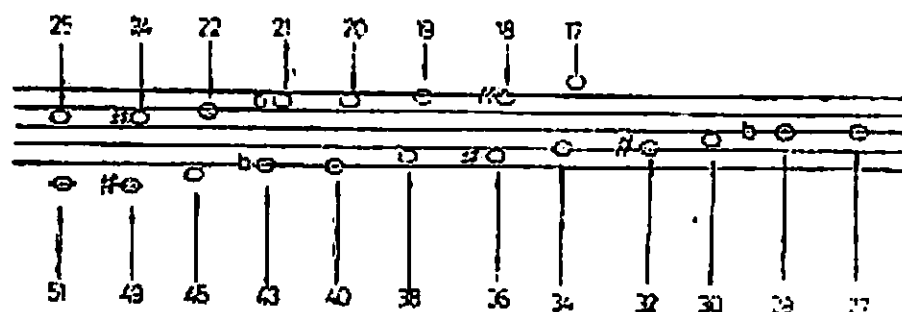


Fig. 9.2. Parametrul de frecvență al notelor.

Pentru a obține (genera) notele dorite, parametrul de frecvență F, va avea valorile din figura 9.2. Programul următor asociază tastelor numerice 1, 2, 3, 4 notele Do, Re, Mi, Fa.

```

5 D=50
7 F=51
10 AS=INKEY$
15 IF AS="" THEN 25
20 ON VAL (AS) GOSUB 30, 40, 50, 60
25 CALL (1, F, D)
27 GOTO 10
30 F=51
35 RETURN
40 F=45
45 RETURN
50 F=40
55 RETURN
60 F=38
70 RETURN

```

O notă va dura până la acționarea unei taste (1-4), corespunzătoare altei note.

Încercați să extindeți programul, încât să asociați și celelalte note unor taste.

Programul următor generează note folosind generatorul de numere aleatoare pentru frecvență și durată :

```

10 CALL (1, INT (RND (0)*20+10), INT (RND (0)*50+5))
20 GO TO 10

```

Generarea de note cu frecvențe din ce în ce mai mari se poate realiza cu programul :

```

10 FOR I=100 TO 10 STEP-1
20 CALL (1, I, 50)
30 NEXT I

```


Observație: Ștergerea programelor de mai sus, în vederea introducerii altora ce utilizează subrutina în limbaj de asamblare, nu se poate face cu SCR pentru că se va șterge și subrutina. Pentru a nu șterge programele linie cu linie, se va ieși de sub controlul interpretorului BASIC acționând RESET, intrând sub controlul Monitorului, iar apoi se va reintra în BASIC cu comanda GØ815. Astfel, subrutina în limbaj mașină va rămâne și se va șterge doar programul BASIC.

3. Se va utiliza o subrutină în limbaj mașină ce permite generarea de sunete de durată mai mare, folosindu-se pentru durată un contor pe doi octeți. Subrutina va primi tot doi parametri, primul reprezentând frecvența iar cel de-al doilea durată.

	ORG	635EH	
	DW	MDISP	; adresa memoriilor disponibile
	DB	1	; programului în limbaj BASIC
	DW	NOTA	; adresa subrutinei
	DB	255	
NOTA :	POP	H	; adresa parametrului 2
	CALL	CVINT	; (durata)
	MOV	B, A	; durata în reg. B
	POP	H	; adresa primului parametru
	MOV	B, A	; în B frecvența
IAR :	CALL	SØ	
	DCR	E	; decrementare contor
	JNZ	IAR	
	RET		
SØ :	LXI	H, 35FH	
S1 :	MOV	D, B	; în D frecvența
	MVI	A, 8	; 1 în bitul 3
	OUT	22H	
S2 :	DCX	H	; buclă în care se
	MOV	A, H	; generează 1 pe durata
	ANA	A	; dată de reg. D.
	RZ		
	DCR	D	
	JNZ	S2	
	MOV	D, B	
	MVI	A, 0	; 0 în bitul 3
	OUT	22H	
S3 :	DCX	H	; buclă în care
	MOV	A, H	; se generează 0
	ANA	A	; pe durata dată de
	RZ		
	DCR	D	; reg. D
	JNZ	S3	
	JMP	S1	

Subrutina CVINT este identică cu cea din exemplul precedent.

Programul următor va apela subrutina în limbaj de asamblare descrisă mai sus pentru interpretarea unei melodii. Frecvențele și duratele sînt date în instrucțiuni DATA, fiind citite cu instrucțiunea READ.

```
1Ø READ A, B
2Ø CALL (1, A, B)
```

```

30 GO TO 10
100 DATA 51, 3, 45, 1, 40, 3, 51, 1, 40, 2
110 DATA 51, 2, 40, 4, 45, 3, 40, 1, 38, 2, 40, 1
120 DATA 45, 1, 38, 8, 40, 3, 38, 1, 34, 3
130 DATA 40, 1, 34, 2, 40, 2, 34, 4, 38, 2
140 DATA 34, 1, 30, 2, 34, 1, 38, 30, 8, 34, 3
150 DATA 51, 1, 45, 1, 40, 1, 38, 1, 34, 1
160 DATA 30, 8, 30, 3, 45, 1, 40, 1, 38, 1
170 DATA 34, 1, 30, 1, 27, 8, 27, 3, 40, 1
180 DATA 38, 1, 34, 1, 30, 1, 27, 1, 25, 7
190 DATA 27, 1, 30, 2, 38, 2, 27, 2, 34, 2, 25, 6
200 END

```

Execuția programului se va termina prin apariția erorii 21, deoarece au fost epuizate toate constantele din instrucțiunile DATA. Modificați programul utilizând instrucțiunea RESTORE, relufund interpretarea melodiei.

9.4.13. Instrucțiuni de calcul cu matrice. Cu toate că instrucțiunile deja prezentate permit efectuarea de prelucrări asupra tablourilor, prin utilizarea variabilelor indexate (cu unul sau doi indici), limbajul BASIC conține un set de instrucțiuni care permit prelucrarea tablourilor (cu una sau două dimensiuni) fără referiri la fiecare element al tabloului. În acest sens sînt disponibile următoarele instrucțiuni:

MAT READ A, B, ... Citește datele din instrucțiuni DATA pentru
MAT READ A (3, 4), B (10) tablourile A, B dimensionate în prealabil sau în instrucțiune.

Observație: Datorită prefixului MAT, utilizat în instrucțiuni, tablourile vor fi numite matrice (indiferent dacă au una sau două dimensiuni).

MAT INPUT A, B, ...	Citește datele pentru matricile A, B de la consolă.
MAT INPUT A (2, 4), B (3, 3), ...	Datele introduse sînt memorate linie cu linie, în matrice.
MAT PRINT A, B, ...	Tipărește linie cu linie valorile curente ale matricilor A, B dimensionate în prealabil sau redimensionate în instrucțiune.
MAT PRINT A (4, 4), B (3, 2), ...	
MAT A=B	Matricea A primește dimensiunile lui B și apoi B este copiat în A.
MAT A=B+C	Adună sau scade matricile B și C. Matricile B și C trebuie să aibă aceleași dimensiuni. Matricea A are dimensiunile matricii rezultat.
MAT A=B-C	
MAT A=B*C	Înmulțește matricile B și C și depune rezultatul în matricea A, care va primi dimensiunile rezultatului. Dimensiunile matricilor B și C trebuie să fie compatibile (produs de matrice).
MAT A=(expresie)*B	Realizează produsul matricii B cu un scalar egal cu valoarea expresiei dintre paranteze. Rezultatul produsului este depus în matricea A, care este dimensionată corespunzător.
MAT A=INV(B)	Inversează matricea pătrată B.
MAT A=INV(B), D	Inversa este memorată în matricea A care va avea dimensiunea lui B. Valoarea determinantului va fi atribuită variabilei D.
MAT A=TRN(B)	Calculează transpusa matricii B și depune valoarea în matricea A, care va avea dimensiunile matricii rezultat. Tablourile A și B trebuie să fie distincte.

MAT A=ZER
MAT A=ZER (3, 4)
MAT A=ZER (7)
MAT A=CON
MAT A=CON (3, 10)
MAT A=CON (8)
MAT A=IDN
MAT A=IDN (1, 5)
MAT A=IDN (7)

Atribuire tuturor elementelor matricii A valoarea zero. Matricea A poate fi redimensionată conform specificațiilor din instrucțiune.

Atribuire tuturor elementelor matricii A valoarea 1. Matricea A poate fi redimensionată conform specificațiilor din instrucțiune.

Atribuire elementelor diagonalei principale valoarea 1, celorlalte elemente li se atribuie valoarea zero. Dacă A este matrice pătrată ea va deveni matrice unitate.

Observație :

Instrucțiunile matriciale pot fi împărțite în două categorii :

— instrucțiuni de intrare/ieșire

MAT READ
MAT INPUT
MAT PRINT

— instrucțiuni de atribuire — în care matricii din membrul sting i se atribuie rezultatul unei operații matriciale.

Matricile introduse prin instrucțiunile de intrare/ieșire sau matricea din membrul sting al unei instrucțiuni de atribuire, pot fi alocate în execuția instrucțiunii MAT respective, nefiind necesară declararea lor în instrucțiunea DIM. În cazul în care matricile au fost declarate în prealabil, într-o instrucțiune DIM, sau au fost introduse printr-o instrucțiune MAT anterioară, ele pot fi redimensionate cu respectarea condițiilor :

— numărul de dimensiuni ale tabloului să fie păstrat (un vector nu poate deveni matrice și nici invers),

— numărul de elemente al matricii redimensionate trebuie să nu depășească numărul de elemente al matricii inițiale.

Dimensiunile specificate în instrucțiunile matriciale pot fi constante, variabile sau expresii.

Exemplu :

MAT READ A (M+N, N+1), B

Citirea matricelor. Citirea matricelor se realizează cu instrucțiunile MAT READ și MAT INPUT.

```

20 MAT READ B (2, 3), A (3)
.
.
.
30 DATA 5, 11, -17, 1, 2, 3, 1E7, 0, 1

```

După execuția instrucțiunii 20 matricile A și B vor conține :

```

B (1, 1)=5 ; B (1, 2)=11 ; B (1, 3)=-17
B (2, 1)=1 ; B (2, 2)=2 ; B (2, 3)=3
A (1)=1E7 ; A (2)=0 ; A (3)=1.

```

Matricile A și B puteau fi declarate și în instrucțiunea DIM caz în care, în instrucțiunea 20, nu mai trebuiau specificate dimensiunile

```

10 DIM A (3), B (2, 3)
20 MAT READ B, A
.
.
.

```

Aceleași observații sînt valabile și pentru instrucțiunea MAT INPUT. În acest caz, datele sînt citite de la consolă (tastatură)

```
10 DIM A (10), B (3, 3)
20 MAT INPUT A (3), B (2, 3)
```

```
·
·
```

Matricile A, B vor fi alocate la execuția instrucțiunii DIM. Instrucțiunea MAT INPUT va redimensiona matricile A și B și apoi va citi de la consolă valorile elementelor (pe linii).

Se observă că redimensionarea satisface cele două cerințe :

- păstrarea tipului (vector sau matrice),
- noul număr de elemente este mai mic decît cel alocat inițial.

Observație : Instrucțiunea 10 DIM nu era absolut necesară, matricile puteau fi alocate în instrucțiunea MAT INPUT.

Tipărirea matricilor. Pentru tipărirea matricilor se utilizează instrucțiunea MAT PRINT. Matricile se tipăresc linie cu linie. În cadrul unei linii, spațierea între elemente se face conform separatorului utilizat în lista de matrici din MAT PRINT (separatori pot fi „,” sau „;”). După tipărirea unei linii a matricii se lasă o linie liberă, pentru ca liniile să apară mai clar. Dacă lista din MAT PRINT conține mai multe matrici, între liniile unei matrici și ale următoarei din listă, se lasă două linii libere.

```
10 DIM A (3, 2)
20 MAT READ A, B (3)
30 MAT PRINT A, B
40 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
50 END
```

Instrucțiunea MAT READ va citi tablourile A și B.

La execuția instrucțiunii 30 se vor tipări următoarele valori la consolă (display) :

```
1          2
2          4
3          6          ; Matricea A (3, 2)
4          8
5          ; Vectorul B (3) va fi tipărit ca
6          ; matrice coloană B (3, 1)
7
8
9
```

Elementele matricii A pot fi tipărite comasat, dacă se folosește separatorul „,”. În acest caz, instrucțiunea 30 va fi :

```
30 MAT PRINT A, B;
```

B va fi tipărit la fel ca mai sus.

În cazul cînd în MAT PRINT, se specifică dimensiuni pentru un tablou din listă, tabloul va fi mai întîi redimensionat și apoi tipărit cu noile dimensiuni. De exemplu, dacă instrucțiunea 30 de mai sus ar fi fost :

```
30 MAT PRINT A (2, 2), B (2)
```

atunci s-ar fi tipărit :

```

1           5           ; matricea A (2, 2)
3           2
7           ; vectorul B (2)
8

```

Observație : La redimensionarea unei matrice, trebuie avut în vedere faptul că elementele matricei sînt memorate liniar, formînd un vector din coloane puse cap la cap. Deci, elementele de pe aceeași poziție, în matrice nu vor fi mereu aceleași după redimensionare. De exemplu, matricea :

$$A(4, 3) = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{bmatrix}$$

redimensionată ca A(3, 3) va fi :

$$A(3, 3) = \begin{bmatrix} 1 & 10 & 8 \\ 4 & 2 & 11 \\ 7 & 5 & 3 \end{bmatrix}$$

Calculul inversei unei matrice. O matrice pătrată, nesingulară poate fi inversată. De remarcat că, o matrice poate fi inversată în ea însăși, adică, matricea din membrul stîng al instrucțiunii poate fi aceeași cu cea din membrul drept.

Exemplu : `10 MAT A=INV(A)` este corectă.

Dacă se dorește și calculul determinantului matricii A (sau numai determinantul) se va folosi instrucțiunea : `10 MAT A=INV(A), V1`.

La execuția instrucțiunii `10`, variabilei `V1` i se va atribui valoarea determinantului matricii A.

```

10 MAT READ A(2, 2)
20 MAT B=INV(A), D
30 MAT C=A*B
40 MAT PRINT A, B, C
50 PRINT D
60 DATA 1, 2, 3, 4
70 END

```

Execuția acestui program va determina tipărirea matricelor :

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad B = A^{-1} = \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$$

$$C = A*B = A*A^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ — matrice unitate}$$

Se va tipări și valoarea determinantului matricii A

$$D = \det(A) = (1*4) - (2*3) = -2$$

Observație : Pentru calculul inversei unei matrice, se scrie transpusa matricei, se înlocuiește fiecare element cu complementul sau algebric (minorul cu semnul corespunzător) împărțit la valoarea determinantului matricei. Inversa se poate calcula mai eficient prin alte metode.

Transpusa unei matrice. Transpusa unei matrice $A (M, N)$ va avea dimensiunile (N, M) . Un vector fiind tratat ca matrice coloană, transpusa sa va fi o matrice linie.

```
10 MAT INPUT V (10)
20 MAT A=TRN (V)
30 END
```

Instrucțiunea 10 va citi de la consolă vectorul V , de 10 elemente. Instrucțiunea 20 va alocă matricea $A (1, 10)$ și-i va atribui valorile elementelor vectorului V . Transpusa unei matrice linie este o matrice coloană.

```
10 DIM V (10)
20 MAT INPUT A (1, 10)
30 MAT V=TRN (A)
40 END
```

În acest program transpusa matricei linie A va fi memorată în vectorul V . **Observații :**

1. În instrucțiunile matriciale se poate lucra cu vectori atâta timp cât operațiile efectuate nu necesită transformarea vectorilor (tablouri cu o dimensiune) în matrice (tablouri cu două dimensiuni). Acest lucru (permițerea utilizării vectorilor în instrucțiunile matriciale) este util în descrierea majorității problemelor în care se prelucrează matricei (ex. : sisteme de ecuații liniare — termenii liberi și necunoscutele sînt notați ca vectori $B (N)$, $X (N)$ nu ca matrice $B (N, 1)$, $X (N, 1)$)

2. O matrice nu poate fi transpusă în ea însăși. Instrucțiunea :

```
10 MAT A=TRN (A) este incorectă.
```

Produsul a două matrice. Pentru a putea înmulți două matrice numărul de coloane al primei matrice trebuie să fie egal cu numărul de linii al celei de-a doua.

Se consideră instrucțiunea :

```
50 MAT A=B*C
```

Dacă B are dimensiunile (P, N) , iar $C (N, Q)$, matricea A va avea dimensiunile (P, Q) .

În cazul în care matricea A nu a fost în prealabil alocată, ea va fi alocată la execuția instrucțiunii 50, cu dimensiunile (P, Q) . Dacă matricea A a fost alocată ea va căpăta dimensiunile (P, Q) .

Observație : Matricea rezultat nu poate figura ca matrice factor. Deci instrucțiunile :

```
10 MAT B=B*C
sau
10 MAT C=B*C
```

sînt incorecte.

Adunarea și scăderea matricelor. Pentru a putea aduna sau scădea două matrice, ele trebuie să aibă aceleași dimensiuni. Aceleași dimensiuni vor fi atribuite și matricei rezultat.

Într-o instrucțiune nu se poate executa decât o singură operație. Instrucțiunea :

10 MAT A=B+C-D este greșită.

Matricea din membrul stâng al atribuirii, poate figura și în membrul drept.

510 MAT A=A+B este corectă

Înmulțirea unei matrice cu un scalar.

10 MAT A=(expresie)*B

Matricea A va primi dimensiunile lui B și va avea ca elemente, elementele matricei B, înmulțite cu valoarea expresiei. Expresia trebuie să fie inclusă între paranteze.

[Exemplu : 30 MAT A=(COS (X)+SIN (X))*B

Inițializarea unei matrice. Generarea unei matrice cu toate elementele zero se poate realiza cu instrucțiunea :

50 MAT A=ZER (3, 2)

Dacă matricea A a fost alocată înainte de execuția instrucțiunii 50 va primi dimensiunile (3, 2), dacă nu, va fi alocată cu aceleași dimensiuni.

În cazul în care se urmărește inițializarea matricii, fără alterarea dimensiunilor, nu se vor mai specifica dimensiunile, în instrucțiunea MAT

10 DIM A (3, 5)

:

:

:

60 MAT A=ZER

Pentru tablouri cu o singură dimensiune se poate utiliza :

10 MAT V=ZER (10)

Pentru a genera o matrice cu toate elementele 1, se pot folosi, de exemplu, instrucțiunile :

60 MAT B=CON (3, 3)

70

70 MAT B=CON

Instrucțiunea 70 va fi utilizată în cazul în care B este alocat în prealabil.

Pentru a genera o matrice unitate (cu unu pe diagonala principală și zero în rest) se va folosi funcția IDN. Exemplu :

10 MAT A=IDN (3, 3)

Dacă matricea nu este pătrată (sau nu se specifică dimensiuni egale în funcția IDN), prin diagonala principală se va înțelege, diagonala ce pleacă din

elementului cu poziția (1, 1) și merge diagonal pînă la epuizarea numărului de linii sau de coloane.

De exemplu :

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

va genera :

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

9.4.14. **Instrucțiuni de prelucrare grafică.** Soluțiile multor probleme constau din șiruri lungi de numere a căror interpretare este destul de dificilă. Reprezentarea sub formă grafică a acestor șiruri de valori numerice facilitează aprecierea cantitativă și calitativă a soluțiilor.

Problemele în care se utilizează facilitățile grafice impun reprezentarea grafică a unor valori numerice (tablouri, valori ale unor funcții etc.) sau realizarea unor desene, hărți etc.

În primul caz utilizatorul este interesat de forma graficului și de încadrarea lui pe ecranul dispozitivului de afișare. În cel de-al doilea caz, pentru utilizator va fi importantă specificarea explicită a scării de reprezentare, pentru a efectua, eventual, măsurători pe desene etc.

Una din soluțiile de realizare a graficelor folosind limbajul BASIC poate consta în scrierea unor subrutine în limbaj de asamblare, care utilizează un anumit display, și utilizarea lor, în BASIC, cu ajutorul instrucțiunii CALL. Această soluție este puțin flexibilă (nu este independentă de tipul perifericului grafic) și destul de greoaie pentru un începător.

De aceea au fost introduse instrucțiuni speciale pentru prelucrări grafice.

Instrucțiunea MOVE. Formatul instrucțiunii :

```
nr. linie MOVE X, Y
```

unde : X, Y pot fi constante, variabile sau expresii.

Instrucțiunea MOVE este folosită pentru a poziționa spotul în punctul de coordonate (X, Y). De menționat că instrucțiunea MOVE execută numai poziționarea în punctul de coordonate (X, Y) nu marchează punctul respectiv.

Instrucțiunea DRAW. Formatul instrucțiunii este :

```
nr. linie DRAW X, Y
```

unde : X, Y pot fi constante, variabile sau expresii.

Instrucțiunea DRAW este folosită pentru a trage o linie între punctul în care se află spotul la întîlnirea instrucțiunii și punctul de coordonate X, Y specificate în instrucțiune.

De exemplu dacă se dorește reprezentarea datelor conținute în vectorul $V(N)$ se poate folosi secvența

```

100 MOVE 1, V(1)
110 FOR J=1 TO N
120 DRAW J, V(J)
130 NEXT J

```

Instrucțiunea 100 execută o poziționare în punctul de coordonate (1, $V(1)$), apoi în ciclul FOR se unesc prin linii elementele vectorului V . Se observă că instrucțiunile MOVE și DRAW sînt suficiente pentru a face reprezentări grafice.

Instrucțiunile WINDOW și VIEWPORT. Coordonatele X, Y specificate în instrucțiunile MOVE și DRAW sînt exprimate în unitățile de măsură în care sînt exprimate mărimile de reprezentat (metri, kg, volți etc.).

Acest sistem de coordonate definește „spațiul utilizatorului” sau spațiul virtual. Spațiul virtual este limitat practic de precizia aritmeticii mașinii ($\pm 10_{18}$). Acest „spațiu utilizator” va fi reprezentat la o anumită scară pe suprafața display-ului.

După cum s-a arătat la începutul paragrafului, în problemele de reprezentare grafică a unor date, utilizatorul preferă să definească limitele spațiului în care sînt cuprinse valorile de prezentat și să accepte scara și originea implicit determinate, fără a mai specifica o scară de reprezentare și o origine.

Instrucțiunea prin care utilizatorul specifică limitele „spațiului utilizator” în care sînt cuprinse datele de reprezentat este :

nr. linie WINDOW A, B, C, D

unde : A, B, C, D sînt variabile, constante sau expresii care reprezintă limitele spațiului utilizator, în ordinea :

A = limita stîngă	C = limita inferioară
B = limita dreaptă	D = limita superioară

Cele patru limite definesc un spațiu dreptunghiular. Orice punct de coordonate (X, Y) pentru care :

$$A \leq X \leq B$$

$$C \leq Y \leq D$$

va fi reprezentat grafic. Punctele care cad în afara dreptunghiului nu vor avea imagine pe suprafața display-ului. Din linia trasă cu instrucțiunile :

```

10 MOVE X1, Y1
20 DRAW X2, Y2

```

va fi reprezentată numai porțiunea interioară dreptunghiului (ferestrei), definite mai sus, de limitele A, B, C, D.

(Observație : De remarcat că poate fi luat implicit spațiul utilizator maxim posibil (impus de aritmetică)

10 WINDOW -10 + 18, 10 + 18, -10 + 18, 10 + 18

În acest caz toate valorile numerice cu care se poate lucra într-un program sînt reprezentate grafic. Dezavantajul constă în aceea că scara implicată este extrem de mare, și un domeniu de valori destul de mare ($\approx 10^6$) se va reprezenta printr-un singur punct (pe un display cu latura de 1000 puncte).

Laturile ferestrei (dreptunghiului) declarate în instrucțiunea WINDOW trebuie să fie cît mai apropiate de domeniul de valori de reprezentat, pentru ca graficul obținut să fie cît mai fin.

Pînă acum s-a considerat că spațiul utilizator, definit prin instrucțiunea WINDOW, va fi reprezentat pe toată suprafața display-ului. În unele aplicații se va dori scrierea unor comentarii alături de grafic sau realizarea mai multor grafice pe aceeași suprafață.

Pentru a descrie porțiunea din suprafața display-ului pe care va fi realizat graficul (va fi proiectat spațiul utilizator) se folosește instrucțiunea :

m. Unte VIEWPORT A, B, C, D

unde : A, B, C, D sînt variabile, constante sau expresii și reprezintă limitele zonei din suprafața display-ului. În aceeași ordine ca pentru WINDOW : stîngă, dreaptă, inferioară, superioară.

Spre deosebire de WINDOW unitățile în care se exprimă limitele A, B, C, D sînt unități fizice.

Alegerea unității fizice în care să se exprime limitele A, B, C, D trebuie să satisfacă cerințele de independență față de tipul perifericului grafic, (unele display-uri au suprafața pătrată, altele dreptunghiulare etc.). Independența de tipul perifericului, asigură ca un program scris pentru un anumit display să se poată folosi și pentru alte tipuri de echipamente de afișare.

Unitatea în care se exprimă limitele A, B, C, D s-a ales ca fiind un procent din latura pătratului cel mai mare, care poate fi înscris în suprafața display-ului. Aceasta se va numi unitate grafică (UG). Originea suprafeței display-ului se consideră în colțul din stînga-jos. În aceste condiții instrucțiunea :

10 VIEWPORT 0, 100, 0, 100

9

va specifica dimensiunile celui mai mare pătrat înscris în suprafața display-ului (dacă suprafața display-ului este pătrată, va reprezenta întreaga suprafață).

Acastă instrucțiune este executată implicit de sistem.

De asemenea implicit este executată și instrucțiunea :

WINDOW 0, 100, 0, 100

Aceste două instrucțiuni realizează o corespondență unu la unu între unitățile utilizator (pe orizontală și pe verticală) și unitatea grafică (UG).

Cu aceste inițializări implicite un cerc va apărea nedistorsionat, deoarece, pe ambele axe de coordonate se folosesc aceleași unități de măsură, iar suprafața graficului este pătrată (aceleași scară pe ambele axe).

Programul următor va trasa un cerc pe orice display :

```

100 MOVE 85, 50
110 FOR J=0 TO 2*PI STEP PI/10
120 DRAW 35*COS (J)+50, 35*SIN (J)+50
130 NEXT J
140 END
    
```

Cercul va avea centrul în punctul de coordonate (50, 50), iar raza va fi 35.

Se observă că, în program, nu a fost necesară utilizarea instrucțiunilor WINDOW sau VIEWPORT. Nu este necesară cunoașterea particularităților display-ului. Este suficient să se știe că orice display are o suprafață de cel puțin 100/100 unități (UG). Evident, pentru reprezentări de date care nu variază între 0 și 100 se impune utilizarea instrucțiunii WINDOW (altfel se reprezintă doar porțiunea din grafic cuprinsă în ferestra 0, 100, 0, 100). Dacă se va dori ca reprezentarea graficului să se facă numai pe o porțiune din suprafața display-ului, va trebui utilizată instrucțiunea VIEWPORT. De pildă, pentru colțul din dreapta sus se va folosi instrucțiunea :

```

210 VIEWPORT 75, 100, 70, 100
    
```

Instrucțiunile prezentate mai sus : MOVE, DRAW, WINDOW, VIEWPORT formează setul minim de instrucțiuni grafice, cu ajutorul cărora se poate rezolva orice problemă de reprezentare grafică. Sînt prevăzute, de asemenea, instrucțiuni care înlesnesc utilizarea subrutinelor pentru realizarea graficelor, manipularea figurilor (rotații, translații) și instrucțiuni care realizează independența programelor față de tipul perifericului grafic.

Instrucțiunile RMOVE și RDRAW. Formatul instrucțiunilor :

nr. linie RMOVE X, Y₂

nr. linie RDRAW X, Y

unde : X, Y pot fi constante, variabile sau expresii.

Instrucțiunile RMOVE și RDRAW se deosebesc de MOVE și DRAW prin faptul că coordonatele X, Y nu sînt raportate la originea sistemului, ci la poziția spotului (punctului grafic), la momentul execuției instrucțiunii. Aceste instrucțiuni sînt utile în scrierea subrutinelor care generează diverse figuri. Figurile respective vor putea fi desenate în orice loc, pe suprafața display-ului, prin utilizarea instrucțiunii MOVE, înainte de apelul rutinei care generează figura. Astfel, se realizează translația figurilor.

Instrucțiunea SCALE. În unele aplicații, realizarea de hărți, diagrame, desene etc., este utilă indicarea explicită a scării de reprezentare dorite.

Formatul instrucțiunii folosite în acest caz este :

nr. linie SCALE S1, S2

unde : S1, S2 sînt constante, variabile sau expresii care reprezintă factorii de scară pe orizontală și pe verticală. Factorul de scară indică numărul unităților utilizator reprezentate pe o unitate grafică

$$S_i = \frac{\text{unități utilizator}}{\text{unități grafice (UG)}}$$

Observație : În cazul în care se dorește efectuarea de măsurători pe grafic, utilizatorul va trebui să știe câți milimetri (de exemplu) reprezintă o unitate grafică (UG) pentru perifericul respectiv (se reamintește că 1UG reprezintă a suta parte din latura pătratului maxim inscriptibil în suprafața display-ului).

Originea sistemului de coordonate se consideră în punctul în care se află spotul (punctul grafic) la execuția instrucțiunii SCALE. Pentru aceasta se va folosi, eventual, instrucțiunea MOVE înainte de instrucțiunea SCALE. Punctul specificat ca origine va fi reprezentat pe grafic în colțul din stînga jos.

Se poate remarca faptul că instrucțiunea WINDOW este echivalentă cu SCALE, avînd în plus specificarea originii. Coordonatele originii vor lua locul parametrilor A, C din instrucțiunea WINDOW, iar factorii de scară vor determina limitele B și D din WINDOW.

Instrucțiunea ROTATE. Formatul instrucțiunii ROTATE este următorul :

nr. linie ROTATE U

unde : U este o constantă, variabilă sau expresie.

Instrucțiunea ROTATE are efect numai asupra instrucțiunilor RMOVE și RDRAW, realizînd rotația cu unghiul U a vectorilor generați cu RDRAW sau a poziționării realizate cu RMOVE.

Dacă instrucțiunea ROTATE este utilizată înainte de apelul unei rutine care generează o figură cu ajutorul instrucțiunilor RMOVE și RDRAW, figura generată va apărea rotită cu unghiul specificat (în radiani). Rotația figurii respective va fi realizată față de punctul în care s-a început generarea figurii, punctul unde se află spotul la începutul secvenței de program (rutinei) ce desenează figura. Instrucțiunile MOVE și DRAW nu sînt afectate de ROTATE, deoarece coordonatele absolute, specificate în aceste instrucțiuni trebuie să rămîină nealterate, pentru a se putea executa poziționări în punctele dorite înainte de generarea unor figuri cu RMOVE și RDRAW. Astfel, se permite utilizarea simultană și independentă a translației și rotației figurilor.

Exemplu : Următorul program va trasa un pătrat cu latura de 10 unități (Se consideră inițializările implicite deci nu se vor folosi WINDOW și VIEWPORT)

```

90 MOVE 0, 0
100 RDRAW 10, 0
110 RDRAW 0, 10
120 RDRAW -10, 0
130 RDRAW 0, -10
140 END

```

Pătratul va fi trasat începînd din origine. Instrucțiunea 90 poziționează spotul în originea spațiului utilizator care pentru inițializările implicite, coincide cu originea suprafeței grafice (colțul din stînga jos).

Pătratul poate fi micșorat, mărit sau transformat în dreptunghi cu ajutorul instrucțiunii SCALE S1, S2. Dacă S1=S2≠1 pătratul va fi micșorat sau mărit, dar laturile vor rămîine egale între ele. Pentru S1=S2=1 pătratul va fi

trasat neschimbat față de cel din exemplul de mai sus. În cazul în care $S1 \neq S2$ laturile trasate pe display nu vor mai fi egale, iar pătratul va fi reprezentat ca dreptunghi.

Următorul program va desena pătratul de două ori mai mare decât cel trasat de programul precedent.

```

10 MOVE 0, 0
20 SCALE 1/2, 1/2
30 GOSUB 100
40 STOP
100 RDRAW 10, 0
110 RDRAW 0, 10
120 RDRAW -10, 0
130 RDRAW 0, -10
140 RETURN
150 END

```

Se observă că secvența de program care desena pătratul a fost scrisă ca subrutină. Dacă se dorește ca pătratul să fie desenat în alt loc pe suprafața display-ului se va schimba instrucțiunea 10. De exemplu, punând :

```
10 MOVE 50, 50
```

pătratul va fi trasat (în sens trigonometric) începând din punctul de coordonate (50, 50).

Observație : Latura pătratului va avea tot 10 unități utilizator (s-a folosit aceeași secvență de instrucțiuni pentru generarea pătratului), însă datorită indicării scării de 1/2, dimensiunea laturii pătratului desenat va fi 20 unități grafice (UG), deci dublă față de latura pătratului trasat de programul precedent.

Pătratul generat de subrutina care începe cu instrucțiunea 100 poate fi trasat începând din punctul de coordonate (50, 50) și rotit cu unghiul $\pi/4$ cu ajutorul programului :

```

10 MOVE 50, 50
20 ROTATE PI/4
30 GOSUB 100
40 STOP

```

În cazul în care se dorește ca pătratul să fie micșorat și rotit, în sens invers trigonometric, se va putea folosi programul :

```

10 MOVE 20, 30
20 SCALE 2, 2
30 ROTATE -PI/3
40 GOSUB 100
50 STOP

```

Pătratul va fi trasat începând din punctul de coordonate (20, 30), micșorat la jumătate (latura de 5UG) și rotit în sens orar cu unghiul $\pi/3$.

La instrucțiunea 100 începe aceeași subrutină de generare a pătratului ca și în programele precedente.

Instrucțiunea INIT. Formatul instrucțiunii este următorul :

```
nr. linie INIT [P]
```

Execuția instrucțiunii INIT constă în ștergerea ecranului display-ului și efectuarea inițializărilor grafice implicite, anume :

```

WINDOW 0, 100, 0, 100
și
VIEWPORT 0, 100, 0, 100

```

Deci instrucțiunile grafice SCALE, WINDOW, VIEWPORT și ROTATE executate înainte de INIT își pierd valabilitatea.

Dacă parametrul P este prezent, atunci display-ul trece în mod pagină, altfel rămâne în mod defilare.

Instrucțiunea GSINPUT. Formatul instrucțiunii GSINPUT este următorul:

nr. linie GSINPUT V1, V2

unde : V1, V2 sînt variabile.

Instrucțiunea GSINPUT este utilizată pentru a realiza independența totală față de tipul perifericului grafic. Execuția instrucțiunii constă în atribuirea variabilelor V1, V2 a dimensiunilor exprimate în unități grafice (UG) ale suprafeței display-ului. De exemplu, un display poate avea suprafața dreptunghiulară cu latura orizontală mai mare sau cu cea verticală mai mare (fig. 9.3 a, b).

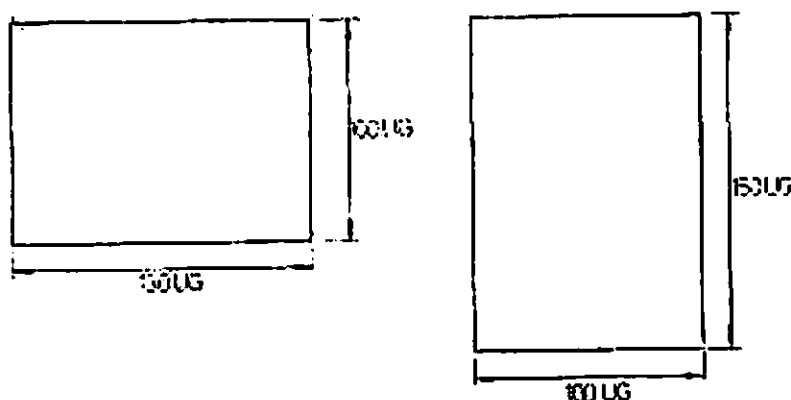


Fig. 9.3. Stabilirea unităților grafice pe suprafața ecranului de afișare : a) V1=130 UG ; V2=100 UG ; b) V1=100 UG ; V2=150UG.

Execuțind instrucțiunea GSINPUT V1, V2 pentru display-ul din figura 9.3. a, variabilele V1, V2 vor primi valorile V1=130, V2=100, iar pentru display-ul din figura 9.3 b, V1=100, V2=150.

Utilizînd variabilele V1, V2 în instrucțiunea VIEWPORT se vor obține programe care utilizează întreaga suprafață a display-ului, indiferent de forma ei :

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

Instrucțiunile PLOT și UNPLOT. Instrucțiunile PLOT și UNPLOT împreună cu caracterele speciale (semigrafice), utilizabile în instrucțiunea PRINT, oferă facilități semigrafice limbajului BASIC. Formatul instrucțiunilor este:

```
nr. linie PLOT X, Y
și
nr. linie UNPLOT X, Y
```

unde : X, Y pot fi constante, variabile sau expresii.

Execuția instrucțiunilor constă în a aprinde sau stinge pătrățelul de coordonate X, Y de pe suprafața display-ului. Suprafața display-ului este formată din 64 × 64 pătrățele, un pătrățel conținând 4 × 4 = 16 puncte grafice. Originea suprafeței, (pătrățelul de coordonate 0, 0) se consideră în colțul din stânga, jos al suprafeței display-ului. Prin urmare, valorile expresiilor X și Y vor fi cuprinse între 0 și 63. Următorul program, trasează două linii, una orizontală, cealaltă verticală, care se intersectează în mijlocul ecranului

```
10 INIT
20 FOR I=0 TO 63
30 PLOT 22, I
40 PLOT I, 32
50 NEXT I
60 END
```

9.5. Mesajele de eroare ale interpretorului BASIC

În timpul introducerii programului, instrucțiunile (liniile ce încep cu un număr) nu sînt analizate sintactic, ci sînt depuse în memorie, în ordinea numerelor de linie. O linie care nu începe cu un număr va fi interpretată de sistem drept o comandă. Dacă ea nu reprezintă o comandă validă (de exemplu este o instrucțiune fără număr de linie) sistemul va tipări mesajul: WHAT?, indicînd faptul că ultima linie introdusă este incorectă. Linia va trebui să fie reintrodusă corect.

Erorile depistate în timpul execuției programului sînt tipărite la consolă sub forma:

ERROR XX IN LINE YYYY

unde:

- XX reprezintă codul erorii
- YYYY reprezintă numărul liniei în care a fost depistată eroarea

Semnificațiile codurilor de eroare (XX) posibile, sînt date în continuare:

Codul erorii	Semnificația
1	— programul nu se termină cu instrucțiunile END sau STOP
2	— tip de instrucțiune nerecunoscut (cuvînt cheie incorect)
3	— există instrucțiuni gresite după instrucțiunea END
4	— numărul liniei destinație este incorect (în instrucțiunile IF, GOTO, GOSUB, ON)
5	— numărul liniei destinație este inexistent (în instrucțiunile IF, GOTO, GOSUB, ON)
6	— caracter ilegal
7	— instrucțiune neterminată
8	— expresie incorectă
9	— eroare în conversia în virgulă mobilă
10	— utilizare incorectă a funcțiilor GET sau PUT
11	— tentativă de redimensionare (prin DIM) a unui tablou
12	— tablou utilizat înainte de a fi definit (stocat)
13	— argumentul funcțiilor SIN, COS sau TAN prea mare (>10°)
14	— relație incorectă în instrucțiunea IF
15	— depășirea stivelor în evaluator — expresie prea complicată

10	— eroare în ridicarea la putere (0 la puterea 0 sau număr negativ la putere reală)
17	— instrucțiune FOR utilizată fără instrucțiunea NEXT corespunzătoare (bucă FOR neînchisă)
18	— instrucțiune NEXT utilizată fără instrucțiunea FOR corespunzătoare
19	— depășire activă FOR (peste 3 cicluri cuprinse unul într-altul)
20	— indice mai mare de 254 sau egal cu 0
21	— se dorește citirea mai multor constante decât au fost definite în instrucțiunile DATA
22	— depășirea valorilor declarate ale indicilor
23	— radical din număr negativ
24	— logaritm din număr negativ
25	— dimensiuni necompatibile pentru produs de matrice
26	— matricea din membrul sting al egalității nu poate apărea în membrul drept (pentru transpusă sau produs de matrice)
27	— matrice singulară — nu poate fi inversată
28	— redimensionarea incorectă — noile dimensiuni depășesc pe cele maxime
29	— prin redimensionare nu poate fi schimbat numărul de dimensiuni al unui tablou
30	— matricea nu este pătrată — nu poate fi inversată
31	— matricile nu au același dimensiuni (pentru sumă sau diferență)
32	— subrutina în limbaj de asamblare cu numărul dorit nu a fost găsită în tabela de subrutine
33	— parametri incorecți în instrucțiune de prelucrare grafică (VIEWPORT, WINDOW sau SCALE)

Fie în faza de introducere a programului, fie în timpul execuției sale poate apărea la consolă mesajul MEMORY FULL, indicând faptul că nu mai există memorie disponibilă, de alocat pentru variabile sau de stocat noi linii de program. În acest moment se poate introduce o singură cifră (de la consolă) care va indica acțiunea care va avea loc în continuare.

Număr introdus

Semnificație

0	RUN	— execută programul din memorie
1	SAVE	— copiază programul din memorie pe casetă
2	LIST	— listează programul din memorie
3	SCR	— șterge programul din memorie

Microcalculatorul aMIC în matematicile elementare și statistică

Dar, mai întâi, o vedere de ansamblu a programării „personale” pe aMIC.

Utilizarea echipamentelor de calcul într-o serie de activități economico-sociale a fost mult timp frînată de regimul de lucru specific puterii de calcul centralizate: prelucrarea lucrărilor pe loturi. Timpul de răspuns, adică durata între momentul predării spre execuție a unei lucrări și momentul primirii rezultatului prelucrării, poate fi de ordinul orelor sau chiar zilelor, funcție de o serie de factori specifici exploatării pe loturi la centrele de calcul. Valoarea mare a timpului de răspuns implică două inconveniente majore:

a) apariția de discontinuități mari în activitatea de elaborare, testare și depanare a programelor, având ca rezultat o eficiență scăzută a activității de programare.

b) imposibilitatea utilizării puterii de calcul în activități care solicită o colaborare permanentă om-calculator pe durata execuției programelor.

Regimul de lucru cu acces direct al utilizatorilor la echipamentul de calcul, fie că este vorba de acces local sau de la distanță, rezolvă în cea mai mare parte inconvenientele de mai sus. Însă această rezolvare nu este în beneficiul unui număr mare de utilizatori. Principalul obstacol în răspândirea acestui mod de lucru îl constituie costul ridicat al unui sistem de calcul interactiv și numărul limitat de terminale ce pot fi gestionate simultan în cadrul unui asemenea sistem.

Apariția microcalculatoarelor, considerate ca fiind sisteme total interactive, permite diseminarea puterii de calcul pentru orice tip de aplicații, prin avantajele specifice pe care le posedă: preț de cost scăzut, programare în limbaje conversaționale de nivel înalt, posibilitatea conectării prin linii de teletransmisie la sisteme de calcul de capacitate medie-mare, interfață simplă cu operatorul uman etc.

Microcalculatorul personal aMIC pune la dispoziția utilizatorilor săi o serie de facilități destinate implementării cel mai simple a puterii de calcul într-un domeniu vast de aplicații.

Interfața cu operatorul este concepută pe două niveluri: nivelul monitor și nivelul BASIC. Ambele au făcut obiectul unei tratări detaliate în capitolele precedente. Relieșăm doar faptul că această ierarhizare, neobișnuită la microcalculatoarele personale, permite dezvoltarea de aplicații în două direcții: cea a domeniilor specializate, necesitând elaborarea de programe în limbaj de asamblare, și cea a domeniilor de larg interes, scrierea aplicațiilor făcându-se în limbajul BASIC-aMIC, al căror obiect îl constituie prezentul capitol.

Limbajul BASIC implementat pe microcalculatorul aMIC este simplu și ușor de învățat, fiind accesibil unei categorii largi de utilizatori, chiar nespecialiști în informatică. Crearea, modificarea și testarea programelor se desfășoară rapid, datorită modului de lucru conversațional al interpretorului BASIC; numărul de erori posibile semnalate la execuția programelor este relativ redus, depanarea programelor neridicând astfel probleme deosebite. Corectarea liniilor de program eronate se poate executa imediat, programul fiind disponibil pentru o nouă rulare, ciclul modificare-testare fiind foarte scurt. Acest mod de lucru, gen „încearcă să vezi ce se întâmplă” nu este cel mai indicat, dar prezintă nete avantaje pentru nespecialiști, eliminând o bună parte din bariera psihologică ridicată de utilizarea unui instrument nou de lucru în cadrul activității clasice a acestora.

Domeniile de utilizare inclund cu preponderență calcule matematice complexe, prelucrării de tip gestiune economică cu un volum mic-mediu de date, aplicații tehnico-științifice, învățământ, aplicații grafice etc. Toate exemplele date în continuare au mai mult un caracter demonstrativ, didactic, încercându-se acoperirea unui număr cât mai mare de domenii diferite.

Pe măsură ce programarea în BASIC este însușită, utilizatorul își poate defini propriile sale aplicații plecând de la exemplele demonstrative; partea cea mai dificilă o constituie abordarea algoritmică a aplicației ce se dorește a fi transpusă pe aMIC. Scopul acestui capitol este și acela de a demonstra că un număr surprinzător de activități pot fi abordate prin prisma transpunerii lor pe microcalculator, și că aceasta este practică la îndemâna tuturor.

10.1. Rezolvarea ecuației de gradul II :

$AX^2+BX+C=0$, unde A, B, C sînt numere reale. RE (X), IM (X) sînt partea reală și partea imaginară ale rădăcinilor complexe conjugate.

Programul se bazează pe organigrama dată în figura 10.1

```

10 PRINT"AX+BX+C=0"
15 PRINT"INTRODUCETI COEFICIENTII:"
25 INPUT A,B,C
30 IF A<>0 THEN 45
35 PRINT"ECUAȚIA NU ESTE DE GR.2"
40 GO TO 95
45 B=B-A*A/C
50 IF B=0 THEN 80
55 PRINT"ECUAȚIA ARE RĂDĂCINI COMPLEXE"
60 PRINT"PARTEA REALĂ",B/A/2
70 PRINT"PARTEA IMAGINARĂ",-SQRT(-B)/A/2
72 GO TO 95
80 PRINT"RĂDĂCINI REALE"
85 Z=SQRT(B)
86 U=2*A
90 PRINT"(-B+Z)/U,(-B-Z)/U"
95 END

```

```

REZOLVAREA ECUAȚIEI
AX+BX+C=0
CU A,B,C NUMERE REALE
INTRODUCETI COEFICIENTII:
3          9          45.6
ECUAȚIA ARE RĂDĂCINI COMPLEXE
PARTEA REALA .666667
PARTEA IMAGINARA -2.8413

```

```

REZOLVAREA ECUAȚIEI
AX+BX+C=0
CU A,B,C NUMERE REALE
INTRODUCETI COEFICIENTII:
0          23          1.8
ECUAȚIA NU ESTE DE GR.2

```

```

REZOLVAREA ECUAȚIEI
AX+BX+C=0
CU A,B,C NUMERE REALE
INTRODUCETI COEFICIENTII:
12          13.12          -45
RĂDĂCINI REALE
1.4651          -2.5584

```

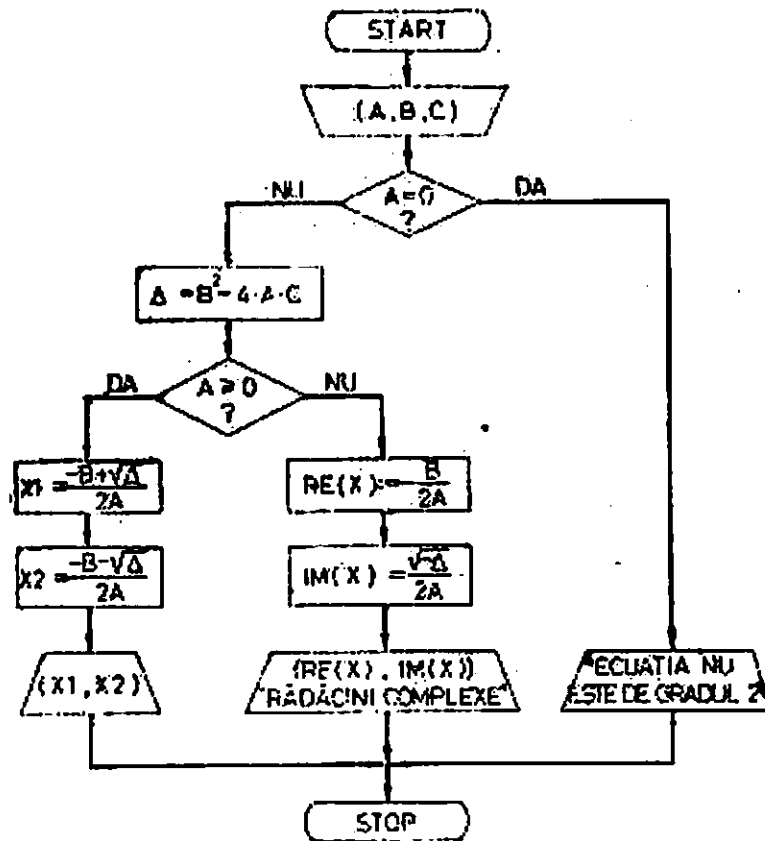


Fig. 10.1. Organigrama rezolvării ecuației de gradul II.

10.2. Rezolvarea inecuației

$$AX^2 + BX + C < 0.$$

Programul se bazează pe organigrama din figura 10.2.

```

0  PRINT "SA SE REZOLVE"
10 PRINT "AX²+BX+C < 0"
15 PRINT "INTRODUCETI COEFICIENTII:"
20 INPUT A,D,C
25 IF A=0 THEN 35
30 X=-B-D/2A
35 IF D<0 THEN 70
40 PRINT "EQUAȚIA NU ARE"
45 PRINT "SOLUȚIE ÎN R"
50 GO TO 170
55 PRINT "EQUAȚIA NU ESTE"
60 PRINT "DE GRADUL 2"
65 GO TO 170
70 V=SQRT(D)
75 U=-B+V
80 T=(-B-V)/U
85 U=(-B-V)/U
90 IF T < U THEN 110
95 U=U
100 W=T
105 T=U
110 GOSUB 140
115 GO TO 170
140 PRINT "X1 = " T ", X2 = " U
150 GO TO 170
160 PRINT "T = " T ", U = " U
165 RETURN
170 END
  
```

```

10 SE REZOLVE
AX²+BX+C < 0
INTRODUCETI COEFICIENTII:
2.2 4.1 70.9
EQUAȚIA ARE
SOLUȚIE ÎN R
  
```

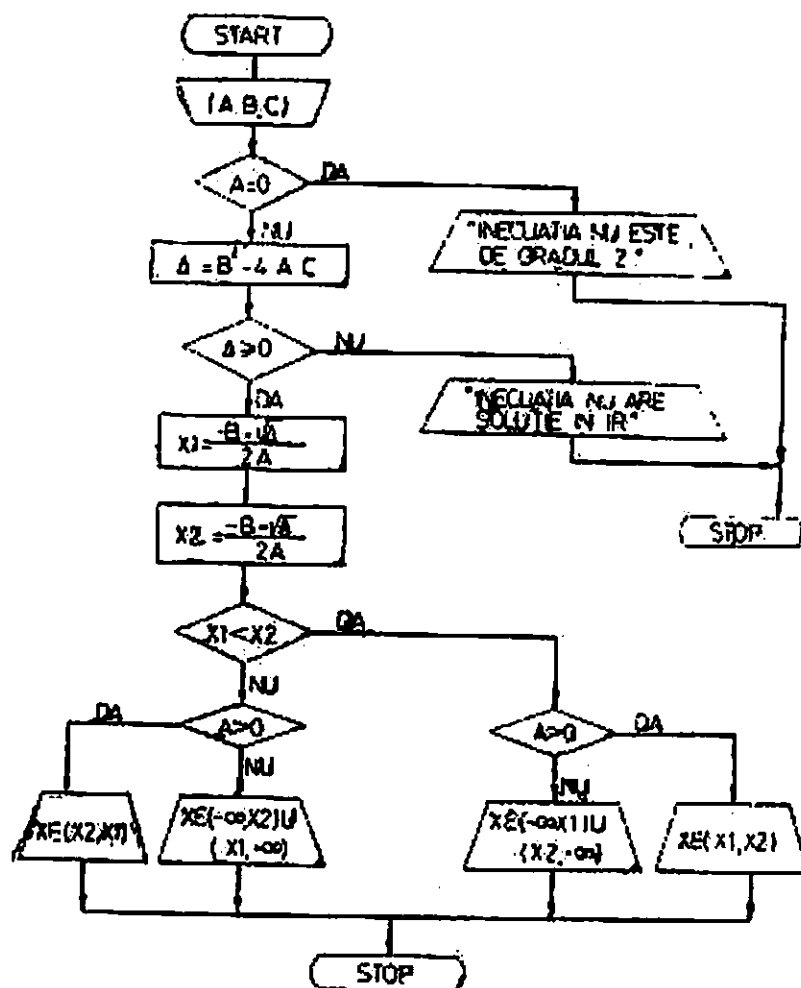


Fig. 10.2. Organigrama rezolvării ecuației $AX^2 + BX + C < 0$.

10.3. Rezolvarea unui sistem (Cramer) de 5 ecuații cu 5 necunoscute

Programul rezolvă un sistem de forma :

$$C \cdot X = D$$

unde :

- C este matrice de 5×5 elemente
- X este vector coloană de 5 elemente
- D este vector coloană de 5 elemente.

Elementele matricilor C și D nu sînt solicitate prin dialog, ele sînt pre-fixate în program la liniile 160, 170 și 180. Dacă se doresc alte valori, se vor modifica aceste linii.

Rezultatul se afișează sub forma unui vector cu 5 elemente.

```

10 PRINT " REZOLVAREA UNUI SISTEM DE ECUAȚII "
20 DIM C(5,5),D(5),E(5,5),F(5),G(5),X(5)
30 MAT READ C,D
40 PRINT " MATRICA COEFICIENTILOR "
50 MAT PRINT C
60 PRINT " TERMENII LIBERI "
70 MAT PRINT D
80 MAT E=INV(C)
90 MAT X=E*D
100 PRINT " VECTORUL SOLUȚIE "
110 MAT PRINT X
120 MAT P=C*X
130 MAT G=D-P
140 PRINT " VECTORUL ERORII "
150 MAT PRINT G
160 DATA 11,3,0,1,7,0,4,2,0,1,3,2,7,1,0
170 DATA 4,0,4,10,1,2,3,1,3,13
180 DATA 51,13,19,20,92
190 STOP
200 END

```

10.4. Afișarea unui șir finit de numere prime

Programul generează și afișează câte un număr prim, pornind de la mulțimea numerelor prime determinate anterior. Mulțimea de pornire este 2,3.

Programul se bazează pe următoarele ipoteze :

- 1) orice număr prim este un număr impar
- 2) având mulțimea primelor n numere prime,

$\{P_1, P_2, \dots, P_n\}$ numărul K , $K > p_n$, va fi număr prim dacă nu este divizibil cu numerele prime $P_i \leq K$.

Intrucât prin BASIC se poate rezerva un masiv cu dimensiunea maximă de 254, se pot reține maximum 254 numere prime. Din acest motiv afișarea se oprește la lipărirea numărului prim $K=P_{255}$, adică la $1613^2=2.601.769$.

```

10 PRINT " LISTARE NUMERE PRIME "
20 DIM P(254)
30 P(1)=2
40 P(2)=3
50 PRINT P(1)
60 PRINT P(2)
70 K=5
80 I=3
90 J=2
100 IF P(J)*2 > K THEN 180
110 IF INT(K/P(J))<K/P(J) THEN 160
120 J=J+1
130 IF J<=254 THEN 100
140 STOP
150 REM " K NU ESTE PRIM "
160 K=K+2
170 GOTO 90
180 REM " K ESTE PRIM "
190 IF I>=254 THEN 210
200 P(I)=K
210 PRINT K
220 K=K+1
230 I=I+1
240 GOTO 90
250 END

```

10.5. Verificarea dacă un număr dat este sau nu prim

Programul solicită prin dialog introducerea numărului de verificat. În baza ipotezelor de la programul anterior se verifică dacă numărul este prim sau nu și se afișează rezultatul (fig. 10.3).

Numărul se compară cu mulțimea numerelor prime $P_i < \text{INT}(\sqrt{X}) + 1$. Din același motiv ca la programul anterior (masiv cu mărime maximă 254) numărul dat la verificare trebuie să fie:

$$X < 2.601.769$$

```

10 PRINT " VERIFICAREA DACA UN NUMAR E PRIM "
20 PRINT " DAFTI NUMARUL "
30 INPUT X
40 IF X<0 THEN 60
50 IF INT(X)-X=0 THEN 80
60 PRINT " NUMAR NEGATIV SAU REAL "
70 GOTO 30
80 DIM P(254)
90 P(1)=2
100 P(2)=3
110 N=INT(SQR(X)+1)
120 K=5
130 I=3
140 REM " COMPARARE CU NUMERLE PRIME "
150 J=2
160 IF P(J)+2>K THEN 230
170 IF INT(K/P(J))=K/P(J) THEN 190
180 J=J+1
190 K=K+2
200 IF K<X THEN 150
210 PRINT X: " NU E PRIM "
220 STOP
230 P(I)=K
240 IF X=K THEN 280
250 X=X+2
260 I=I+1
270 GOTO 150
280 PRINT X: " ESTE PRIM "
290 STOP
300 END

```

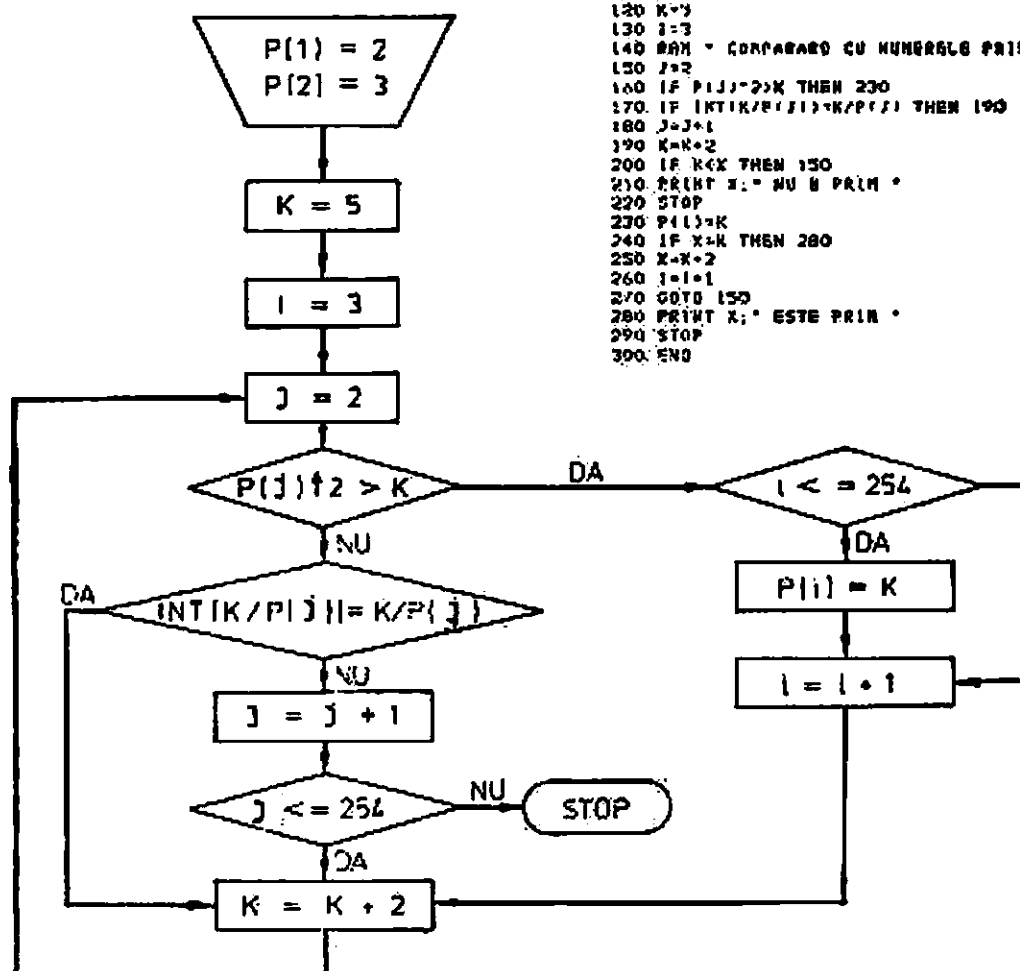


Fig. 10.3. Organigrama verificării dacă un număr este sau nu prim.

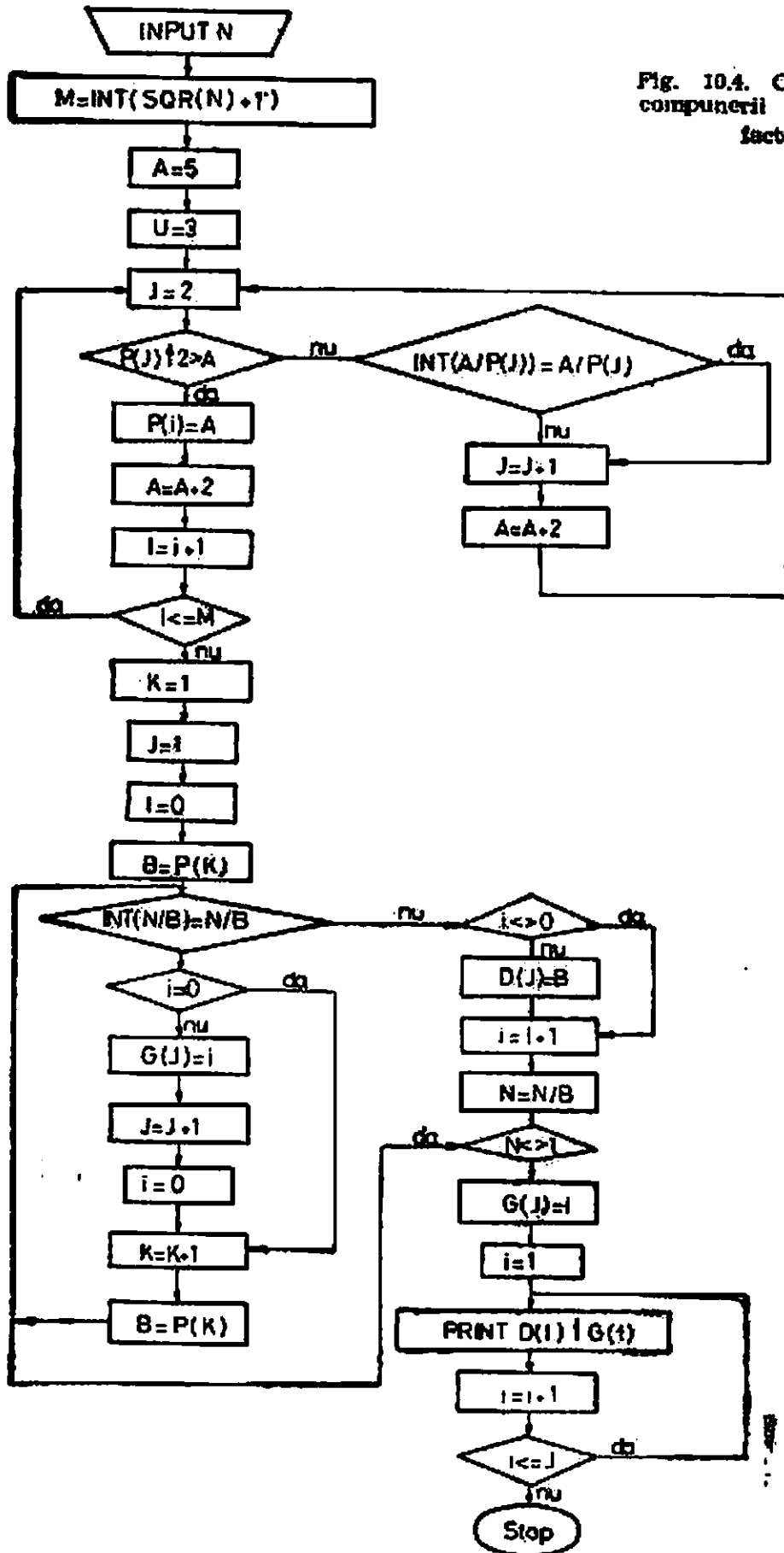


Fig. 10.4. Organigrama descompunerii unui număr în factori primi.

10.6. Descompunerea unui număr în factori primi

Programul solicită prin dialog numărul de descompus în factori primi. Se aplică ipotezele de la 10.4 și 10.5 pentru a obține șirul de numere prime-piață la \sqrt{N} , N fiind numărul dat. Se contorizează apoi de câte ori se cuprinde fiecare număr prim în numărul dat.

Considerând descompunerea de forma :

$$N = q_1^{k_1} \cdot q_2^{k_2} \cdot \dots \cdot q_m^{k_m}$$

se afișează :

$q_1 \uparrow k_1$

$q_2 \uparrow k_2$

.

.

.

$q_m \uparrow k_m$

```

5  PRINT " DESCOMPUNEREA IN FACTORI PRIMI "
10  DIM P(254),B(20),G(20)
15  PRINT " DATI NUMARUL "
20  INPUT N
25  M=INT(SQR(N))+1
30  P(1)=2
35  P(2)=3
37  U=3
40  A=5
45  FOR I=U TO M
50  J=2
55  IF P(J)**2>A THEN 90
60  IF INT(A/P(J))=A/P(J) THEN 90
65  J=J+1
70  IF I<254 THEN 55
75  STOP
80  A=A+2
85  GOTO 50
90  IF I>254 THEN 100
95  P(I)=A
100 A=A+2
105 NEXT I
110 REM " DESCOMPUNEREA "
115 K=1
120 J=1
125 I=0
130 B=P(K)
135 C=N/B
140 IF (INT(C)-C)=0 THEN 160
145 IF I=0 THEN 165
150 G(I)=1
155 J=J+1
160 I=0
165 K=K+1
170 B=P(K)
175 GOTO 135
180 IF I<>0 THEN 190
185 D(I)=0
190 I=I+1
195 N=C
200 IF C<>1 THEN 135
205 G(I)=I
210 REM " TIPARIREA "
215 FOR I=1 TO J
220 PRINT G(I);"";G(I)
225 PRINT
230 NEXT I
235 PRINT " CONTINUATI (DA=1) "
240 INPUT D
245 IF D=1 THEN 235
250 STOP
255 PRINT " DATI NUMARUL "
260 INPUT U
265 IF U<N THEN 115
270 M=U
275 U=N
280 N=(INT(SQR(N))+1)
285 REM " A= ULTIM NUMAR VERIFICAT "
290 GOTO 45
300 END

```


10.7. Determinarea celui mai mare divizor comun

Programul solicită prin dialog cele două numere întregi și afișează cel mai mare divizor comun al lor.

```

5 PRINT " CEL MAI MARE DIVIZOR COMUN "
10 PRINT " DATI CELE DOUA NUMERE "
15 INPUT A,B
20 IF A=B THEN 50
25 IF A>B THEN 40
30 B=B-A
35 GOTO 20
40 A=A-B
45 GOTO 20
50 PRINT A
85 GOTO 10
90 END

```

10.8. Simplificarea unei fracții

Programul solicită prin dialog numărătorul și numitorul fracției. Se afișează un vector cu toate valorile cu care se poate simplifica fracția, inclusiv fracția simplificată.

```

5 PRINT " SIMPLIFICAREA FRACȚIEI "
10 PRINT " DATI NUMERATORUL "
15 INPUT A
20 PRINT " DATI NUMITORUL "
25 INPUT B
30 DIM S(100)
35 I=0
40 D=2
45 IF D<=B THEN 95
50 IF I=0 THEN 80
55 PRINT " SE POATE SIMPLIFICA CU: "
60 MAT PRINT S(I)
65 PRINT " FRACȚIA SIMPLIFICATĂ: ";
70 PRINT A;" / ";B
75 STOP
80 PRINT " NU SE POATE SIMPLIFICA "
85 PRINT " FUNCȚIA DATA: ";
90 GOTO 70
95 C=B/D
100 IF INT(C)=C THEN 115
105 D=D+1
110 GOTO 45
115 E=A/D
120 IF INT(E)=E THEN 130
125 GOTO 105
130 A=A-E
135 B=B-C
140 I=I+1
145 S(I)=D
150 GOTO 40
155 END

```

10.9. Calculul aproximativ al factorialului unui număr

Programul solicită prin dialog numărul n și determină factorialul său pe baza formulei:

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

Formula este indicată pentru valori mari ale lui n , deoarece nu există înmulțiri succesive, mari consumatoare de timp.

```

5 PRINT " FACTORIAL "
10 PRINT " DATI NUMARUL "
15 INPUT N
20 IF N<0 THEN 35
25 PRINT " NUMAR NEGATIV SAU ZERO "
30 STOP
35 F=1:FOR I=1 TO N:GOTO 40
40 PRINT N*I:GOTO 45
45 STOP
50 END

```

10.10. Permutări, aranjamente, combinări

Programul solicită mai întâi tipul operației, și corespunzător acesteia, parametrii necesari. Calculele se execută după formulele:

a) permutări de n elemente:

$$P_n = n!$$

b) combinări de n elemente luate câte r :

$$C_n^r = \frac{n!}{(n-r)!r!}$$

c) aranjamente de n elemente luate câte r :

$$A_n^r = \frac{n!}{(n-r)!}$$

Se afișează apoi rezultatul operației.

```

10 PRINT " P. PERMUTARE A N ELEMENTE
20 PRINT " A. ARANJAMENTE A R ELEMENTE DIN N
30 PRINT " C. COMBINARE A N ELEMENTE LUATE CITE R
40 AS=INKEY$
50 IF AS="P" THEN 40
60 IF AS="A" THEN 100
70 IF AS="C" THEN 150
80 IF AS="Q" THEN 180
90 GOTO 10
100 PRINT " INTRODUCETI P > N
110 INPUT P
115 IF AS="P" THEN 170
120 PRINT " INTRODUCETI R > N
130 INPUT R
140 IF R<=N THEN 170
150 PRINT " R > N
160 GOTO 130
170 Z = P
180 GOSUB 400
190 IF AS="P" THEN 340
200 B = A
210 Z = N-R
220 GOSUB 400
230 C = B
240 IF AS="A" THEN 280
250 Z = R
260 GOSUB 400
270 C = C*A
280 PRINT "N",A,"R",":",B/C
290 PRINT " CONTINUATI IO/N "
300 AS=INKEY$
310 IF AS=" " THEN 300
320 IF AS="D" THEN 10
330 STOP
340 PRINT "MP",":",A
350 GOTO 290
400 A = 1
410 FOR X=1 TO Z
420 A = A*X
430 NEXT X
440 RETURN
450 END

```

Subprogramul de la linia 400 calculează exact valoarea factorului $Z!$. Se apelează cu valorile Z egale cu n , $n-r$, r de care avem nevoie în formulele anterioare.

10.11. Ordonarea unui șir de numere

Programul solicită mai întâi câte numere se ordonează, apoi șirul de numere efective. Se execută ordonarea în ordine crescătoare sau descrescătoare, afișând șirul ordonat.

```

5 PRINT " ORDONAREA UNUI ȘIR DE NUMERE
10 PRINT " ÎN ORDINE CRESCĂTOARE/DESCRESCĂTOARE.
15 PRINT " CITE NUMERE INTRODUCETI ? "
20 INPUT N
25 DIM A(N)
30 PRINT " DATI ȘIRUL DE NUMERE "
35 MAT INPUT A
40 PRINT " ÎN ORDINE CRESCĂTOARE (DA=1)
45 INPUT B
55 FOR I=1 TO N-1
60 FOR J=I+1 TO N
65 IF B=1 THEN 80
70 IF A(I)>A(J) THEN 110
75 GOTO 85
80 IF A(I)<A(J) THEN 110
85 B=A(I)
90 FOR K=J TO I+1 STEP -1
95 A(K)=A(K-1)
100 NEXT K
105 A(I)=B
110 NEXT J
115 NEXT I
120 MAT PRINT A
125 STOP
130 END

```

10.12. Calculul sumei celor mai mari numere dintr-un șir de numere date

```

10 PRINT " ADUNAREA A N NUMERE DIN B
20 PRINT " AVIND VALOAREA CEA MAI MARE
30 PRINT " CITE NUMERE INTRODUCETI ? "
40 INPUT M
50 DIM A(M)
60 PRINT " DATE NUMERELI
70 MAT INPUT B
80 PRINT " CITE NUMERE SE ADUAA
90 INPUT N
100 IF N<=M THEN 130
110 PRINT " N > M SE ADUAA FOARTE NUMERELI
120 B=M
130 DIM B(N)
140 FOR I=1 TO M
150 B(I)=A(I)
160 FOR J=I+1 TO N
170 IF B(I)>=B(J) THEN 210
180 C=B(I)
190 B(I)=B(J)
200 B(J)=C
210 NEXT J
220 NEXT I
230 B=0
240 FOR I=1 TO N
250 B=B+B(I)
260 NEXT I
270 PRINT " SUMA CELOR "B;" NUMERE "
275 PRINT " CU VALOAREA CEA MAI MARE "
280 PRINT B
290 STOP
300 END

```

Programul solicită prin dialog numărul total de elemente și valorile lor, apoi solicită valoarea m .

Se afișează suma celor m numere.

10.13. Calculul valorii medii ponderate a unei variabile aleatoare

Se efectuează calculul valorii medii ponderate după formula :

$$V = \frac{\sum_{i=1}^n a_i \cdot x_i}{\sum_{i=1}^n a_i}$$

Se introduce prin dialog numărul total de măsurători, valorile fiecărei măsurători și ponderile cu care se efectuează calculul.

Se afișează valoarea V.

```

5  PRINT " VALOAREA MEDIEI MATEMATICE CU PONDERE "
10 PRINT " DATI NUMARUL DE MASURATORI "
15 INPUT N
20 DIM N(N),X(N)
25 PRINT " DATI VALORILE MASURATORILOR "
30 NAT INPUT N
35 PRINT " DATI PONDERILE "
40 NAT INPUT X
45 A=0
50 B=0
55 REM " CALCULUL "
60 FOR I=1 TO N
65 A=A+N(I)*X(I)
70 B=B+N(I)
75 NEXT I
80 PRINT " V = " :A/B
85 STOP
90 END

```

10.14. Calculul valorii medii și abaterii standard a unei variabile aleatoare

[Se determină] valoarea medie și abaterea standard după formulele :

$$V = \frac{1}{n} \sum_{i=1}^n X_i$$

$$S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - V)^2$$

Se introduce prin dialog numărul total de măsurători și valorile lor. Se afișează valoarea medie și abaterea standard determinată.

```

0 PRINT "CALCULUL VALORII MEDII SI A ADAPTERII"
10 PRINT "STANDARD"
15 PRINT "CITE MASURATORI ATI EFECTUAT"
20 INPUT N
25 PRINT "DAITI VALORILE MASURATORILOR"
30 MAT INPUT N:IN
35 S=0
45 FOR X=1 TO N
50 S=S+IN(X)
55 NEXT X
60 V=S/N
65 S=0
75 FOR X=1 TO N
80 S=S+(IN(X)-V)^2
85 NEXT X
90 Z=SQR(S/(N-1))
95 PRINT "VALOAREA MEDIE: ",V
100 PRINT "ADAPTEREA STANDARD: ",Z
105 STOP
110 END

```

10.15. Tabela valorilor unei funcții definite pe intervale

Programul solicită prin dialog trei intervale ce vor alcătui domeniul de definiție, de forma :

[A, B], (B, C), [C, D]

Funcția este definită astfel :

- pe intervalul [A, B] în linia 35
- pe intervalul (B, C) în linia 45
- pe intervalul [C, D] în linia 55

```

5 PRINT "FUNCTIA VARIABILA IN INTERVALE"
10 PRINT "DAITI INTERVALELE A,B,C,D"
15 INPUT A,B,C,D
20 FOR X=A TO D STEP 0.25
25 K=4*SIN(X-A)+SIN(X-B)+SIN(X-C)+SIN(X-D)
30 ON K GOTO 33,35,45,47,49,53,55
35 F=0
40 GOTO 60
45 F=SIN(X-B)
50 GOTO 60
55 F=SIN(X-B)+C*X
60 PRINT "X";X;"P(X)";F
65 NEXT X
70 STOP
75 END

```

Definițiile pot fi modificate în cadrul programului. Linia 20 definește pasul cu care se efectuează tabelarea valorilor funcției. Ca rezultat, se afișează valorile funcției și ale argumentului pe intervalele date.

10.16. Calculul volumului butoiului

Se determină volumul butoiului pe baza formulei aproximative :

$$V \approx \pi \left(\frac{2D+d}{3} \right)^2 \cdot L.$$

unde :

D este diametrul secțiunii prin vrană

d este diametrul fundului butoiului

L este înălțimea butoiului.

Se solicită prin dialog circumferința la vrană și la fund și înălțimea, afișând ca rezultat volumul butoiului.

```

10 PRINT " VOLUMUL BUTOIULUI "
20 PRINT " DATI CIRCUMFERINȚA LA VRANĂ "
30 INPUT C1
40 PRINT " DATI CIRCUMFERINȚA LA FUND "
50 INPUT C2
60 PRINT " DATI ÎNĂLȚIMEA "
70 INPUT L
80 D1=C1/PI
90 D2=C2/PI
100 V=PI*L*((D1+D2)/2)**2
110 PRINT " VOLUMUL BUTOIULUI : ";V
120 STOP
130 END

```

10.17. Calculul volumului și suprafeței torului

Se determină volumul și suprafața torului pe baza formulelor :

$$V = 2\pi^2 Rr^2$$

$$A = 4\pi^2 Rr$$

unde :

R este raza torului

r este raza secțiunii torului

Se solicită prin dialog valorile R și r , și se afișează suprafața și volumul determinate.

```

10 PRINT " VOLUMUL ȘI SUPRAFAȚA TORULUI "
20 PRINT " DATI RAZA TORULUI "
30 INPUT R1
40 PRINT " DATI RAZA SECȚIUNII "
50 INPUT R2
60 D1=2*R1
70 D2=2*R2
80 F=PI**2*D1*C2
90 U=F*D2/4
100 PRINT " SUPRAFAȚA : ";F
110 PRINT " VOLUMUL : ";U
120 STOP
130 END

```

10.18. Calculul perimetrului și suprafeței unui triunghi

Programul permite calcularea perimetrului și ariei unui triunghi în două cazuri : fie prin introducerea lungimilor celor trei laturi, fie prin introducerea lungimii unei laturi și a măsurii celor două unghiuri adiacente. Se afișează perimetrul și suprafața determinată.

```

10 PRINT "CALCULUL PERIMETRULUI SI "
15 PRINT "SUPRAFETEI TRIUNGHULUI "
20 PRINT "SE CUNOSC LUNGIMILE LATURILOR "
30 PRINT "DATI LUNGIMILE LATURELOR "
40 INPUT X,Y,Z
50 P=X+Y+Z
60 Q=P/2
70 S=SQR(Q*(Q-X)*(Q-Y)*(Q-Z))
80 PRINT "PERIMETRUL: ";P
90 PRINT "SUPRAFATA: ";S
100 STOP
110 END

```

```

10 PRINT "INTR-UN TRIUNGH: SE CUNOSC: "
20 PRINT "LUNGIMEA UNEI LATURI SI "
30 PRINT "UNGHIELE PE ACEASTA LATURA "
40 PRINT "DATI LUNGIMEA LATURII "
50 INPUT Y
60 PRINT "DATI UNGHIELE "
70 INPUT U
80 INPUT V
90 W=360-(U+V)
100 X=YSIN(WPI/180)/SIN(UPI/180)
110 Z=YSIN(WPI/180)/SIN(VPI/180)
120 P=X+Y+Z
130 Q=P/2
140 S=SQR(Q*(Q-X)*(Q-Y)*(Q-Z))
150 PRINT "LATURILE: ";X,Y,Z
160 PRINT "UNGHIELE: ";U,V,W
170 PRINT "PERIMETRUL: ";P
180 PRINT "SUPRAFATA: ";S
190 STOP
200 END

```

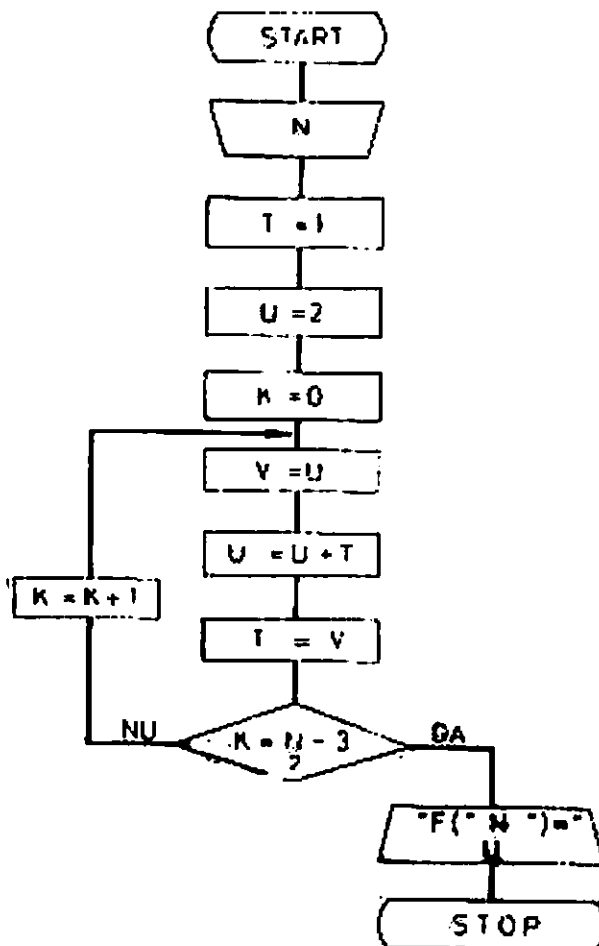
10.19. Calculul celui de-al N-lea număr din șirul lui Fibonacci.

S-a plecat de la relația de recurență :

$$F(N) = F(N-1) + F(N-2), \quad (\forall) N \geq 3$$

$$F(1) = 1; F(2) = 2.$$

Programul se bazează pe organigrama dată în figura 10.5.



```

10 PRINT "PROGRAMUL TIPARESTE"
20 PRINT "AL N-LEA NUMAR DIN"
30 PRINT "SIRUL LUI FIBONACCI"
40 PRINT "N="
50 INPUT N
60 T=1
70 U=2
80 FOR K=0 TO N-3
90 V=U
100 U=U+T
110 T=V
120 NEXT K
130 PRINT "F( "N" ) = "U
140 END

```

Fig. 10.5. Organigrama calculului celui de-al N-lea număr din șirul lui Fibonacci.

10.20. Calculul aproximativ al rădăcinii

$$\sqrt[n]{X} = Z, n \geq 2, X > 0$$

S-a folosit relația de recurență :

$$Z_{k+1} = \frac{1}{n} \left[(n-1)Z_k + \frac{X}{Z_k^{n-1}} \right] \quad (V) \quad K \geq 0$$

$$Z_0 = \begin{cases} x, & \text{dacă } x > 1, \\ 1, & \text{dacă } x \leq 1. \end{cases}$$

Programul se bazează pe organigrama dată în figura 10.6.

Procesul de calcul se oprește când este satisfăcută relația :

$$|Z_{k+1} - Z_k| < \epsilon$$

```

7  PRINT "CALCULUL RADICALULUI"
10 PRINT "DE GRADUL N=2 DINTR-UN"
15 PRINT "NUMAR POZITIV X"
20 PRINT "N="
25 INPUT N
30 IF N=2 THEN 45
35 PRINT "EROR"
40 GOTO 20
45 PRINT "X="
50 INPUT X
55 IF X=0 THEN 65
60 IF X=1 THEN 75
65 PRINT "EROR"
70 GOTO 45
75 PRINT "REZULTATUL ESTE 0."
80 GOTO 130
85 IF X=1 THEN 100
90 Y=1
95 GOTO 105
100 Y=X
105 Z=(N-1)*Y+X/(Y*(N-1))/N
110 IF ABS(Z-Y)<=10*(1-5) THEN 125
115 Y=Z
120 GOTO 105
125 PRINT "RADICALUL DEH ".C." ESTE:".Z
130 END
    
```

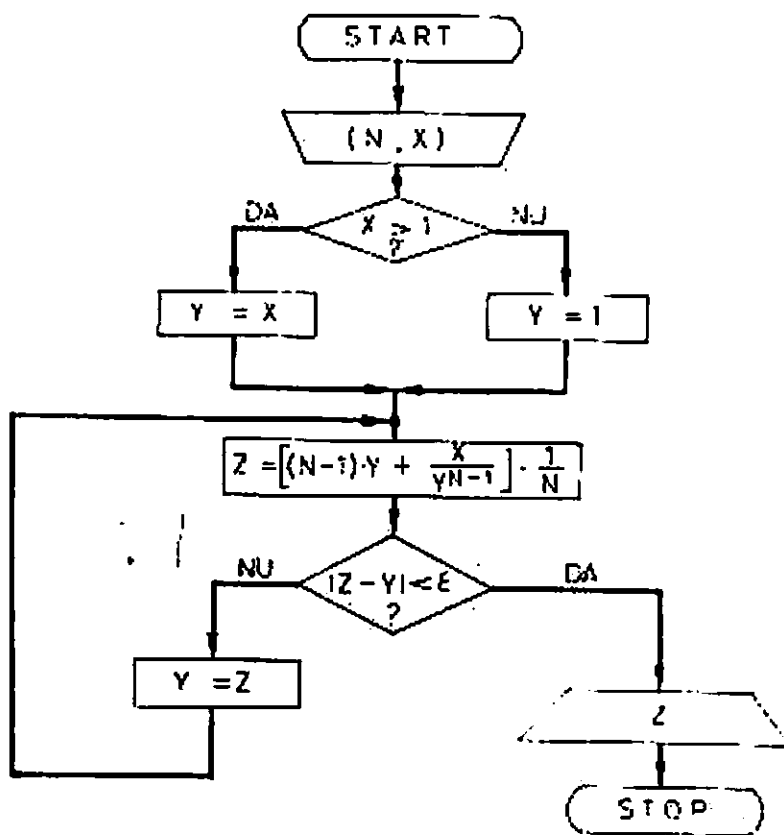


Fig. 10.6. Organigrama calculului aproximativ al rădăcinii $\sqrt[n]{X} = Z, n \geq 2, X > 0$.

11.1. Antecalculația de preț pentru un produs

Programul efectuează calculul prețului de producție și livrare a unui produs, în baza formulei:

$$\text{Preț producție și livrare} = (\text{cost complet prod.}) + (\text{beneficiu})$$

$$\text{Cost complet prod.} = (\text{cost uzină}) + (\text{cheltuieli desfășurare})$$

$$\text{Beneficiu} = 15\% (\text{cost complet producție}).$$

```

5 PRINT " ANTECALCULAȚIA DE PREȚ "
10 PRINT " MATERII PRIME ȘI MATERIALE "
15 PRINT " DIRECTE: "
20 INPUT A
25 PRINT " DISTRIBUȚIE DIRECTE "
30 INPUT B
35 REM " CALCUL IMPOZIT "
40 I = (B*14)/100
45 PRINT " IMPOZIT 14%: ";I;" LEI "
50 REM " CALCUL C.A.S. "
55 C = (I+B)*15/100
60 PRINT " C.A.S.: ";C;" LEI "
65 REM " CHELT. FUNC. UTILAJE "
70 K = B*I+C
75 U = (K*18)/100
80 PRINT " CHELT. FUNC. UT.: ";U;" LEI "
85 REM " CHELT. GEN. SECȚIE "
90 D = (K*80)/100
95 PRINT " CHELT. GEN. SECȚIE: ";D;" LEI "
100 REM " COST SECȚIE "
105 S = A+K+D+U
110 PRINT " COST SECȚIE: ";S;" LEI "
115 REM " CHELT. GEN. INTREPR. "
120 E = (S*12)/100
125 PRINT " CHELT. GEN. INTREPR.: ";E;" LEI "
130 REM " COST UZINA "
135 F = S+U
140 PRINT " COST UZINA: ";F;" LEI "
145 REM " CHELT. DESFĂȘURARE "
150 G = (F*3)/100
155 PRINT " CHELT. DESFĂȘURARE: ";G;" LEI "
160 REM " COST COMPLET PROD. "
165 P = F+G
170 PRINT " COST COMPLET PROD.: ";P;" LEI "
175 REM " BENEFICIU 15% "
180 N = (P*15)/100
185 PRINT " BENEFICIU 15%: ";N;" LEI "
190 REM " CALCUL PREȚ PROD. ȘI LIVRARE "
195 Q = P+N
200 PRINT " PREȚ PROD. ȘI LIVRARE: "
205 PRINT " = ";Q;" LEI "
210 PRINT " >>> ALT PRODUS? (DA=1) <<< "
215 INPUT B
220 IF B = 1 THEN 10
225 STOP
230 END

```

Cost uzină = (cost secție) + (cheltuieli generale ale întreprinderii).

Cheltuieli desfășurare = 3% (cost uzină).

Cheltuieli generale ale întreprinderii = 12% (cost secție)

Cost secție = (cost materii prime) + (retribuții directe) + (impozit 16%) + (C.A.S.) + (chelt. funcționare utilaje) + (cheltuieli generale secție).

Se solicită prin dialog variabilele: cost materii prime și valoarea retribuțiilor directe. Se afișează toți parametri intermediari enumerați mai sus, iar la sfârșit, prețul final de producție și livrare. Procentajele din cadrul calculului unor parametri constituie un caz particular, fiecare utilizator putând modifica programul conform cerințelor proprii.

Programul este util în cazul în care se dorește aflarea rapidă a variației prețului de producție în funcție de variația costului materiilor prime ce intră în componența produsului și a valorii retribuțiilor necesare la executarea sa.

11.2. Calculul primei acordate după grupa de vechime

Se consideră următoarele grupe de vechime :

- | | |
|----------------|---------------|
| I. < 5 ani | IV. 15—20 ani |
| II. 5—10 ani | V. > 20 ani |
| III. 10—15 ani | |

Programul solicită prin dialog totalul sumei ce se va acorda drept prime pentru fiecare grupă de vechime. Se determină suma ce revine fiecărui angajat conform cu grupa din care face parte, și afișează lista tuturor angajaților, luată de retribuiția tarifară și prima acordată.

```

10 PRINT "VALOAREA PRIMEI IN FUNCTIE
15 PRINT " DE GRUPE DE VECHIME "
20 PRINT " GRUPE VECHIME " : G 3-10
25 PRINT " 10-15 " 15-20 " > 20 ANI "
30 DIM P(2),N(10) : S
40 PRINT " INTRODUC SUMAZERURA CE SA "
45 PRINT " ABEREA DREPT PRIMA
50 NAT INPUT "
60 PRINT " INTROD NR TOTAL ANGAJATI "
70 INPUT M
80 NAT I = 0
90 FOR I = 1 TO M
100 READ N(I)
110 READ U
120 K = INT(U/5)+1
130 T(K) = C(K)+1
140 READ S
150 NEXT I
160 FOR I = 1 TO S
170 NEXT I
180 NEXT I
190 RESTORE
200 FOR I = 1 TO M
210 READ N(I)
220 READ U
230 K = INT(U/5)+1
240 READ S
250 PRINT N(I);S;P(K)
260 NEXT I
270 STOP
280 REM "NUME , VECHIME , RETRIBUIE "
290 DATA "IONESCU",0,1000
300 DATA "DOBESCU",0,2000
310 DATA "..."
320 END

```

Datele referitoare la angajați se introduc prin DATA conform cu liniile de la sfârșitul programului. Fiecare linie va conține numele angajatului, vechimea sa în muncă și retribuiția tarifară.

11.3. Determinarea beneficiului pentru o structură de fabricație pe produse dată

Programul poate fi utilizat ca mijloc de determinare a celei mai avantajoase structuri de fabricație pe grupe de produse. Se solicită prin dialog următoarele date :

- numărul și numele produselor de fabricație ;
- numărul și numele materialelor (materiilor prime) necesare fabricării produselor ;
- pentru fiecare produs, cantitatea de materii prime ce intră în fabricația sa ;
- pentru fiecare materie primă, cantitatea existentă în stoc ;
- pentru fiecare produs, beneficiul estimat ;
- structura de fabricație (câte bucăți se vor fabrica din fiecare produs).

Se verifică mai întâi dacă în baza datelor introduse, se poate realiza structura de fabricație cerută (nedepășirea stocului de materii prime), iar dacă rezultatul verificării este pozitiv, se afișează beneficiul total pentru această structură. Se poate repeta întregul proces, modificând doar structura de fabricație, până la determinarea unei structuri optimizate (fig. 11.1).

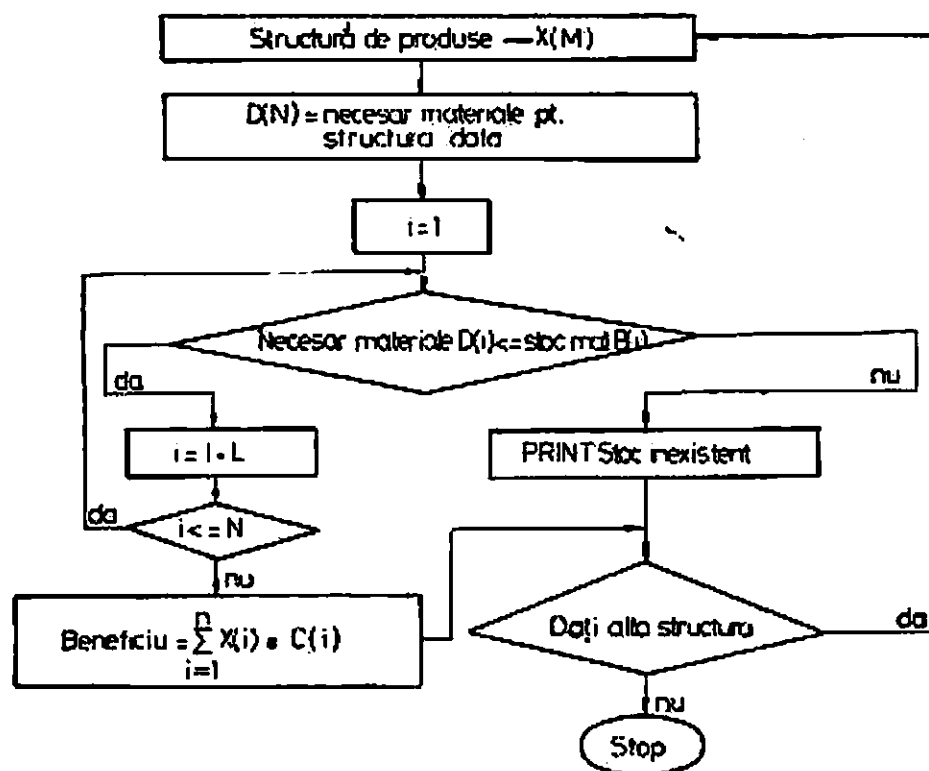


Fig. 11.1. Organigrama programului pentru determinarea beneficiului pentru o structură de fabricație pe produse date.

Datele despre produse și materiale rămân nemodificate pe parcursul iterațiilor, introducerea lor fiind necesară numai în faza inițială, de creare a tabelor de date din program. Odată salvat pe casetă cu aceste date introduse, programul poate fi reluat oricând direct de la dialogul de stabilire a structurii de fabricație.

Variabilele M , N conțin numărul de produse respectiv de materiale. Matricile utilizate conțin următoarele date:

$D \{ M \}$ — denumirile produselor

$P(N)$ — denumirile materialelor

$A(M, N)$ — cite unități din materialul N necesită produsul M

$B(N)$ — stocul de materiale existente

$C(M)$ — beneficiul estimat pe produse

Ciclul de verificare începe de la linia 300:

```

10 PRINT " DETERMINAREA BENEFICIULUI "
12 PRINT " IN FUNCTIE DE O STRUCTURA "
13 PRINT " DE FABRICATIE ALBATA "
20 PRINT " (INTROD.NR.TOTAL PRODUSE "
25 PRINT " SI NR.TOTAL MATERIALE "
30 INPUT M,N
40 DIM A(M,N),X(M),B(N),C(N),D(M)
50 DIM D$(M,10)
60 DIM P$(N,10)
70 PRINT " INTROD.DENUMIRI PRODUSE "
80 FOR I = 1 TO M
90 INPUT D$(I)
100 NEXT I
110 PRINT " INTROD.DENUMIRI MATERIALE "
120 FOR I = 1 TO N
130 INPUT P$(I)
140 NEXT I
145 PRINT " INTROD.CONSUM MAT/PRD. "
150 FOR I = 1 TO M
160 FOR J = 1 TO N
170 PRINT D$(I); " NR.DUC.DIN: ";P$(J)
180 INPUT A(I,J)
190 NEXT J
200 NEXT I
205 PRINT " INTROD. STOC MAT "
210 FOR J = 1 TO N
220 PRINT " DIN ";P$(J); " CITS U/S AVETI "
230 INPUT B(J)
240 NEXT J
245 PRINT " INTROD.BENEF.ESTIMAT "
250 FOR I = 1 TO M
260 PRINT D$(I); " CE BENEF.REALIZATA "
270 INPUT C(I)
280 NEXT I
290 PRINT " INTROD.STRUCTURA DE FABR. "
295 PRINT " NR. PRODUSE PE TIPURI "
300 FOR I = 1 TO M
310 PRINT D$(I); " PRODUS "
320 INPUT X(I)
330 NEXT I
340 MAT B = IER
350 FOR I = 1 TO M
360 FOR J = 1 TO N
370 D(I) = D(I) + X(I) * A(I,J)
380 NEXT J
390 NEXT I
400 J = 0
410 FOR I = 1 TO M
420 IF J <> 0 THEN 460
430 IF D(I) <= B(I) THEN 460
440 J = 1
450 PRINT " STRUCTURA DATA NU SE POATE "
460 PRINT " REALIZA "
470 NEXT I
475 IF J = 0 THEN 520
480 PRINT " ALTA STRUCTURA? (DA=1) "
490 INPUT E
500 IF E = 1 THEN 290
510 STOP
520 E = 0
530 FOR I = 1 TO M
540 S = S + X(I) * C(I)
550 NEXT I
560 PRINT " BENEFICIUUL DATE: ";S
570 GOTO 480
580 END

```

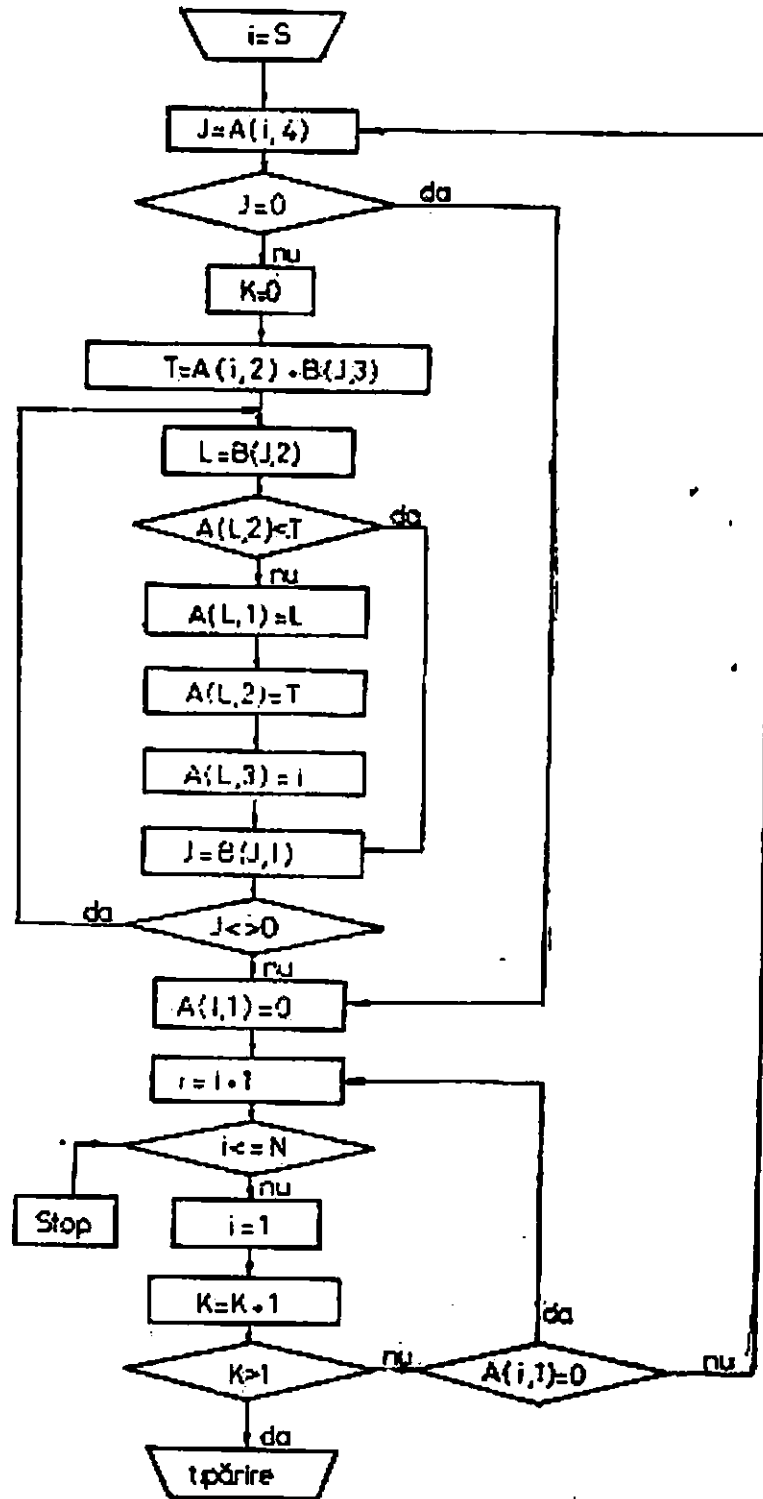


Fig. 11.2. Organigrama programului pentru determinarea drumului minim între două noduri ale unui graf dat.

11.4. Determinarea drumului minim între două noduri ale unui graf dat

Programul consideră un graf ca fiind definit prin noduri și arce, fiecare nod fiind etichetat cu un număr întreg pozitiv. Se solicită prin dialog:

- numărul total de noduri = N
- numărul total de arce = E
- eticheta (numărul) nodului de pornire (primul nod din drumul de minimizat) = S
- eticheta (numărul) nodului de sosire (ultimul nod din drum) = M
- pentru fiecare arc:
 - nodul inițial
 - nodul final
 - lungimea sa.

Din datele introduse se completează masivul $A(N, 4)$ în care $A(L, 2) = 1000000$ considerat un timp foarte mare respectiv $A(S, 2) = 0$ pentru nodul de pornire. Masivul $B(E, 3)$ se completează cu numărul nodurilor pe arce și cu timpii necesari parcurgerii arcelor. Conectorul $A(L, 4)$ va conține legătura către liniile din masivul B .

În $A(L, 2)$ se totalizează timpii necesari parcurgerii grafului, conform schemei din figura 11.2.

Se afișează sub forma unui șir de etichete (numere) drumul minim între cele două noduri specificate. Afișarea se face în ordinea inversă parcurgerii sale (de la nodul de destinație la nodul de pornire).

Programul poate fi utilizat într-o serie de domenii unde este necesară optimizarea distanțelor parcurse, planificarea activităților de producție etc. Se va considera lungimea unui arc drept măsură pentru distanțe de parcurs, timpii necesari în efectuarea unor operații tehnologice etc.

```

10 PRINT "DRUMUL MINIM"
20 PRINT "NUMAR DE NODURI"
30 INPUT N
40 PRINT "NUMAR DE ARCE"
50 INPUT E
60 DIM A(N,4),B(E,3)
70 PRINT "NR.-UL NODULUI DE PORNIRE"
80 INPUT S
90 PRINT "NR.-UL NODULUI DE SOSIRE"
100 INPUT M
110 FOR I = 1 TO N
120 A(I,1) = 1
130 A(I,2) = 1000000
140 A(I,3) = 0
150 A(I,4) = 0
160 NEXT I
165 REM "TIMPUL DE PORNIRE" = 0
170 A(S,2) = 0
180 PRINT "NOD 1, NOD 2, LUNGIME"
190 FOR P = 1 TO E
200 INPUT A,B,L
210 B(P,1) = A
220 B(P,2) = B
225 REM "INLANTIREA TABLOU A SI B"
230 B(P,3) = A+B*L
240 A(A,4) = P
250 NEXT P
260 I = S
270 J = A(I,4)
280 IF J = 0 THEN 390
290 K = 0
300 I = A(I,2)+B(K,1)
310 L = B(I,2)
320 IF A(I,2) < L THEN 240
330 A(I,1) = 1
340 A(I,2) = L
350 A(I,3) = 1
360 J = B(I,1)
370 IF J <> 0 THEN 300
380 A(I,1) = 0
390 I = I+1
400 IF I <= N THEN 420
420 I = 1
430 K = K+1
440 IF K > 3 THEN 170
450 IF A(I,1) = 0 THEN 390
460 GOTO 270
470 PRINT "SIN "L:" IN "J:" DRUMUL"
480 PRINT "CEL MAI SCURT ESTE:"
490 PRINT "PRIN NODURILE: "J:
500 X = A(I,3)
510 IF X = 0 THEN 390
520 PRINT " "X:
530 X = A(X,3)
540 GOTO 310
550 STOP
560 END

```

11.5. Gestiunea unui stoc de magazie de tehnică dentară

Se consideră că există o magazie care primește materiale specifice dentare, și le distribuie la mesele de lucru.

Fiecare masă de lucru elaborează o anumită lucrare, pentru care se consumă o anumită cantitate de materiale.

Fiecărei lucrări îi este asociat un barem de consumuri utilizat pentru verificarea încadrării în normele stabilite.

Programul oferă următoarele opțiuni de lucru :

- punerea la zi a magaziei centrale (primire/eliberare materiale) ;
- vizualizarea stocului existent în magazia centrală ;
- punerea la zi a stocului de materiale la mesele de lucru, cu verificarea consumului de materiale ;
- vizualizarea stocului de materiale existent la mesele de lucru.

```

10 PRINT "MAGAZIA DE TEHNICA DENTARA "
20 PRINT " 1- SE TINE LA ZI MAGAZIA CENTRALA "
30 PRINT " 2- VIZUALIZARE STOC MAGAZIE CENTRALA "
40 PRINT " 3- VIZUALIZARE STOC LA MESA "
50 PRINT " 4- SE TINE LA ZI STOCUL LA MESA "
55 PRINT " 5- STOP "
60 PRINT " DATI NUMARUL PROLUCRARI "
70 INPUT X
80 ON X GOTO 90,400,780,880,1340
90 PRINT " SE TINE LA ZI MAGAZIA CENTRALA "
100 PRINT " PRIMITE SAU ELIBERATI MAT.(P/E) "
110 INPUT A$
120 IF A$ = "E" THEN 450
130 IF A$ <> "Y" THEN 90
140 PRINT " DATI MATERIALUL SI CANTITATEA "
150 INPUT B$(1),X
160 Y = 0
170 FOR I = 1 TO X
180 IF B$(I) <> "0" THEN 220
190 G(I) = G(I)+X
200 Y = Y + 1
210 I = I + 1
220 NEXT I
230 IF Y <> 0 THEN 340
240 PRINT " MATERIAL NOU7 (D/N) "
250 INPUT A$
260 IF A$ = "D" THEN 280
270 GOTO 340
275 REM " SE ADUGA UN MATERIAL NOU "
280 N = N+1
290 D(N) = X
300 H$(N) = B$
310 FOR I = 1 TO X
320 PRINT " CONSUM SPECIFIC PRODUS "(P$1);
325 PRINT " DEN MATERIAL "(B$
330 INPUT P(I),N
335 NEXT I
340 PRINT " MAT PRIMITE MATERIAL (D/N) "
350 INPUT A$
360 IF A$ = "D" THEN 140
370 PRINT " DORITI VIZUALIZARE STOC (D/N) "
380 INPUT A$
390 IF A$ <> "B" THEN 10
400 PRINT " VIZUALIZARE STOC MAGAZIE CENTRALA
410 FOR I = 1 TO N
420 PRINT H$(I), " " G(I)
430 NEXT I
440 GOTO 10
450 PRINT " SE ELIBERAZA MATERIAL LA MESA "
460 PRINT " MATI NUMARUL MUSHI "
470 INPUT S
480 IF S <= 0 THEN 400
490 IF S > 0 THEN 400
500 PRINT " DATI MATERIALUL SI CANTITATEA "
510 INPUT B$(1),X
520 Y = 0
530 FOR I = 1 TO X
540 IF B$(I) <> "0" THEN 570

```

```

530 Y = 1
540 I = N
570 NEXT I
580 IF Y <> 0 THEN 610
590 PRINT " NU EXISTA MATERIALUL IN MAGAZIE "
600 GOTO 680
610 Q(Y) = Q(Y)-X
620 IF Q(Y) >= 0 THEN 640
630 PRINT " NU EXISTA CANTITATEA IN MAGAZIE "
640 Q(Y) = Q(Y)+X
650 GOTO 680
660 RM = SE ABUNA LA STOCUL MESEI "
670 A(S,Y) = A(S,Y)+X
680 PRINT " MAT ELIBERATI LA MASA "S;" (D/N) "
690 INPUT AS
700 IF AS = "0" THEN 500
710 PRINT " ELIBERATI LA ALTA MASA? (D/N) "
720 INPUT AS
730 IF AS = "0" THEN 640
740 PRINT " VRETI VIZUALIZAREA STOC MASA (D/N) "
750 INPUT AS
760 IF AS = "0" THEN 780
770 GOTO 10
780 PRINT " VIZUALIZAREA STOC MASA "
790 PRINT " DAT: NUMARUL MESEI "
800 INPUT S
810 IF S <= 0 THEN 790
820 IF S > 0 THEN 790
830 FOR I = 1 TO N
840 PRINT " M<I>," "A(S,I) "
850 NEXT I
860 PRINT " MAI VIZUALIZATI STOC LA MASA (D/N) "
870 GOTO 750
880 PRINT " SE DAU REALIZARILE LA MESE "
890 PRINT " DAT: NUMARUL MESEI "
900 INPUT S
910 IF S <= 0 THEN 890
920 IF S > 0 THEN 890
930 MAT D(N) = ZER
940 PRINT " DAT: PRODUSUL SI NR. PROD. REALIZATE "
950 INPUT B(10),X
960 Y = 0
970 FOR I = 1 TO N
980 IF B(I) <> 0 THEN 1010
990 Y = 1
1000 I = N
1010 NEXT I
1020 IF Y <> 0 THEN 1130
1030 PRINT " PRODUS NOUT M/N: "
1040 INPUT AS
1050 IF AS <> "0" THEN 1130
1055 RM = SE ABANDA UN PRODUS NOU "
1060 N = N+1
1070 P(N) = B
1080 PRINT " DAT: CONSUMURILE SPECIFICE PROD.NOU "
1090 FOR I = 1 TO N
1100 PRINT B(I);

```



```

1110 INPUT PIR, I
1120 NEXT I
1130 PRINT " MAI DATI REALIZARE? (D/N) "
1140 INPUT A$
1150 IF A$ = "D" THEN 930
1155 MAT C(1) = ZER
1160 PRINT " DATI CONSUM MATERIALE LA NARA " :
1170 INPUT B(1) : X
1180 Y = 0
1190 FOR I = 1 TO N
1210 IF B(I) <> N(1) THEN 1270
1220 Y = 1
1230 IF A(I, 1) < X THEN 1260
1240 Y = 2
1250 C(I) = X
1260 I = N
1270 NEXT I
1280 IF Y = 2 THEN 1340
1290 IF Y = 1 THEN 1390
1300 PRINT " MATERIAL INEXISTENT "
1310 GOTO 1340
1320 PRINT " STOC INSUFICIENT "
1330 PRINT " MAI DATI CONSUM? (D/N) "
1340 INPUT A$
1350 IF A$ = "D" THEN 1170
1360 FOR I = 1 TO N
1380 A(I, 1) = A(I, 1) - C(I)
1390 NEXT I
1400 MAT D = ZER
1410 FOR J = 1 TO M
1420 FOR I = 1 TO N
1430 D(I, J) = D(I, J) + B(I) * P(I, J)
1440 NEXT J
1450 NEXT I
1460 Y = 0
1470 FOR I = 1 TO N
1480 IF D(I, 1) <= C(I) THEN 1500
1490 PRINT " S-A DEPASIT CONSUMUL LA " : N(1) :
1495 PRINT " MATERIAL CU " : C(I) - D(I)
1500 Y = 1
1510 NEXT I
1520 IF Y = 1 THEN 10
1530 PRINT " VAZI INCAURAT IN CONSUM SPECIFIC "
1540 GOTO 10
1550 STOP
1600 REM " CREATI FIȘIER STOC "
1610 REM " SE CREEAZĂ LA MAXIM 100 MATERIALE "
1615 REM " 20 PRODUSE ȘI 5 MĂRE "
1620 DIM O(100), P(20), C(100), D(100)
1630 DIM A(5, 100), P(20, 100)
1640 DIM P(120, 30), N(100, 10)
1650 REM " SE INTRODUC 5 MATERIALE 2 PRODUSE "
1655 REM " BESTIUL ȘI INTRODUC PRIN PROGRAM "
1660 N = 5
1670 M = 2
1680 O = 5
1690 MAT O = ZER
1700 P(1) = "PLAN INCL."

```

```

1710 P(2) = "MUNOBLUC"
1720 N(1) = "CEARA"
1730 N(2) = "MASA ANBL."
1740 N(3) = "ALIAJ IROX"
1750 N(4) = "OIPS"
1760 MAT A = IER
1770 MAT P = IER
1780 REM " CONSUMURILE SPECIFICE "
1790 FOR I = 1 TO N
1800 FOR J = 1 TO M
1810 READ P(I, J)
1820 NEXT J
1830 NEXT I
1835 REM " CONSUM SPECIFIC DUPA BARRA "
1840 DATA 5, 10, 14, 600, 60
1850 DATA 15, 0, 0, 200, 60
1860 GOTO 10
1900 END

```

Programul poate fi utilizat ca model de referință pentru gestiunea stocurilor la magazine mici-medii, spațiul de memorie ocupat de datele referitoare la magazine depinzând în mare măsură de modul în care sînt codificate materialele din stoc. Asocierea unui număr mare de date unui material dat va reduce numărul total de materiale ce pot fi cuprinse în tabelele de descriere a magaziei. Implicit, numărul maxim de materiale diferite nu poate fi mai mare de 254, deoarece interpretorul BASIC nu admite tablouri a căror dimensiune să fie mai mare de 254.

În prima lansare se utilizează comanda RUN 1600 care dimensionează matricile utilizate. Prin inițializare s-au introdus 5 materiale, 2 produse și consumurile specifice aferente. Prin program se mai introduce restul de materiale și produse. După inițializare programul se lansează cu comanda GOTO 10 pentru a nu inițializa masivul completat cu stocul magaziei centrale, respectiv cu stocul meselor de lucru.

11.6. Balanța de verificare debit-credit

Programul constituie un exemplu simplu de realizare a unei balanțe de verificare. La prima rulare, programul solicită introducerea numărului total de conturi și a numerelor de cont pentru care se vor face operațiile de debit/credit. Se introduce apoi, pentru fiecare număr de cont, debitele și creditele pe luna în curs. Se afișează apoi balanța de verificare pe subconturi; la umplerea ecranului, afișajul se oprește, putînd fi reluat prin apăsarea unei taste oarecare. După afișarea balanței pe subconturi, se afișează balanța de verificare pe conturi, împreună cu totalurile debitoare, creditoare și pe conturi.

Rulările următoare nu mai solicită introducerea numerelor de cont, ci doar debitul și creditul pe luna curentă. Verificările se fac începînd cu soldul lunii precedente. Pentru a nu distruge masivul cu soldul lunii precedente, lansarea ulterioară a programului se realizează cu comanda GOTO 100.

```

10 PRINT "BALANȚA DE VERIFICARE "
20 PRINT "DAȚI NR. TOTAL DE CONTURI "
30 INPUT N
40 DIM X(N),C(N),D(N)
50 DIM R(N),R(N),S(N)
60 MAT C = ZER
70 MAT D = ZER
80 PRINT "INTRODUCETI CONTURILE "
90 MAT INPUT K
100 PRINT "INTRODUCETI CREDIT/DEBIT "
105 PRINT "LUNA CURENTA "
110 FOR I = 1 TO N
120 PRINT X(I);";"
130 PRINT "DB;";
140 INPUT B(I)
150 PRINT "CR;";
160 INPUT R(I)
170 NEXT I

```

```

180 FOR I = 1 TO N
190 C(I) = C(I)+R(I)
200 D(I) = B(I)+C(I)
210 S(I) = D(I)-C(I)
220 NEXT I
230 INIT P
240 PRINT AT(1,1); " BALANTA DE VERIFICARE
245 B = INKEY$
250 U = 3
260 FOR I = 1 TO N
270 PRINT AT(U,1);K(I),S(I)
280 U = U+1
290 IF U < 31 THEN 350
300 A$ = INKEY$
310 IF A$ = B$ THEN 300
320 U = 3
330 INIT P
340 PRINT AT(1,1); " BALANTA DE VERIFICARE
350 NEXT I
360 A$ = INKEY$
370 IF A$ = B$ THEN 380
380 T = S(I)
390 N = INT(K(I)/1000)
400 INIT P
410 PRINT AT(5,1); " TOTAL CONT
420 U = 3
430 FOR I = 2 TO N
440 IF N <> INT(K(I)/1000) THEN 470
450 T = T+S(I)
460 GOTO 520
470 PRINT AT(U,1);N,T
480 N = INT(K(I)/1000)
490 T = S(I)
500 U = U+1
510 IF U < 31 THEN 570
520 A$ = INKEY$
530 IF A$ = B$ THEN 520
540 U = 3
550 INIT P
560 PRINT AT(1,1); " TOTAL CONT "
570 NEXT I
580 PRINT AT(U,1);N,T
590 A$ = INKEY$
600 IF A$ = B$ THEN 610
610 INIT P
620 PRINT AT(1,1); " TOTAL BALANTA "
630 K = 0
640 U = 0
650 Q = 0
660 FOR I = 1 TO N
670 H = R+C(I)
700 U = U+D(I)
710 Q = Q+S(I)
720 NEXT I
730 PRINT AT(3,1); "TOTAL DB: ";U
740 PRINT AT(3,1); "TOTAL CR: ";H
750 PRINT AT(7,1); "BALANTA : ";Q
760 STOP

```

11.7. Transformarea stea-triunghi și reciproc

Programul determină rezistențele echivalente celor două transformări după formulele :

$$a) \begin{cases} R_{12} = R_1 + R_2 + \frac{R_1 \cdot R_2}{R_3} \\ R_{23} = R_2 + R_3 + \frac{R_2 \cdot R_3}{R_1} \\ R_{31} = R_3 + R_1 + \frac{R_3 \cdot R_1}{R_2} \end{cases} \quad (\text{transformarea stea-triunghi})$$

$$b) \quad r_1 = \frac{r_1 k - r_{12}}{r_{12} + r_{13} + r_{23}} \quad (\text{transformarea triunghi-stea})$$

```

10 PRINT "CALCUL REZISTENTE ECHIVALENTA"
20 PRINT "1= SCHEMA ELECTRICA STEA IN TRIUNGH"
25 PRINT "2= SCHEMA ELECTRICA TRIUNGH IN STEA"
30 INPUT X
40 IF X = 1 THEN 210
50 PRINT "SCHEMA ELECTRICA TRIUNGH IN STEA"
60 PRINT "DAZI REZISTENTELE IN TRIUNGH"
70 PRINT "R12 = "
90 INPUT A
90 PRINT "R23 = "
100 INPUT B
110 PRINT "R31 = "
120 INPUT C
130 E = A+B+C
140 D = (A*B)/E
150 E = (B*A)/E
160 F = (C*D)/E
170 PRINT "R1 = "D
180 PRINT "R2 = "E
190 PRINT "R3 = "F
200 STOP
210 PRINT "SCHEMA ELECTRICA STEA IN TRIUNGH"
220 PRINT "DAZI REZISTENTELE IN STEA"
230 PRINT "R1 = "
230 INPUT J
240 PRINT "R2 = "
250 INPUT K
260 PRINT "R3 = "
270 INPUT L
280 A = E+L-(D+E)/F
290 B = E+L-(E+F)/D
300 C = F+D-(L+D)/J
310 PRINT "R12 = "A
320 PRINT "R23 = "B
330 PRINT "R31 = "C
340 STOP
350 END

```

Se solicită prin dialog tipul transformării și valorile rezistențelor corespunzătoare. Se afișează valorile rezistențelor echivalente.

11.8. Dimensionarea liniilor de alimentare in curent continuu

Se consideră un conductor de cupru enplat la o sursă de curent continuu. De la acest conductor sînt alimentați mai mulți consumatori, fiecare avînd un consum P_i și o distanță L_i de la sursa de alimentare. Secțiunea conductorului se deduce pe baza formulei :

$$S = \frac{2 \cdot 100}{\gamma \Delta U \% U^2} \sum P L_i$$

unde :

$\gamma = \frac{1}{\rho}$ este conductivitatea conductorului

$\Delta U \%$ este căderea de tensiune maxim admisă

U este tensiunea nominală

P este puterea absorbită de fiecare consumator

L_i este distanța consumatorului față de sursa de alimentare.

11.10. Dimensionarea grinzilor de beton armat

Programul determină înălțimea utilă și aria armăturii unei grinzi de beton armat, pe baza formulelor :

$$h_0 = \frac{\sqrt{\frac{m}{1 \cdot r_b}}}{\sqrt{\xi \left(1 - \frac{\xi}{2}\right)}}$$

$$a = \frac{m}{r_a \cdot h_0 \cdot \left(1 - \frac{\xi}{2}\right)}$$

unde :

- h_0 este înălțimea utilă ;
- a este aria armăturii ;
- m este momentul de încovoiere
- l este lățimea grinzii
- r_a este rezistența armăturii
- r_b este rezistența betonului
- ξ este factor limită = 0,6

Se introduce prin dialog valorile pentru r_a , r_b , l și m . Se afișează valorile înălțimii utile și ale ariei armăturii.

```

10 PRINT "DIMENSIONARE GRINZI DE BETON ARMAT"
20 REM "SE CONSIDERA C S I LIMITA = 0.6"
30 P = 0.4
40 PRINT "REZISTENTA ARMATURII IN KG/CM^2"
50 INPUT A
60 PRINT "REZISTENTA BETONULUI IN KG/CM^2"
70 INPUT B
80 PRINT "LATIMEA GRINZII IN CM"
90 INPUT D
100 PRINT "MOMENTUL INCOVOTIETOR IN KG.CM"
110 INPUT M
120 T = L*P/2
130 S = P+1
140 R = 3/520153
150 H = R*SQRT(M/(B*D))
160 Z = H*(A+B+T)
170 REM "INALTIMEA UTILA"
180 PRINT "H0 = " ; H ; " CM"
190 PRINT "ARIA ARMATURII = " ; Z ; " CM^2"
200 STOP
210 END

```

11.11. Calculul secțiunii elementelor de construcție

Programul determină caracteristicile principale ale diafragmelor utilizate în construcții, pornind de la descompunerea secțiunii diafragmei în dreptunghiuri elementare. Se solicită ca date inițiale numărul de dreptunghiuri elementare ce compun secțiunea, precum și dimensiunile unui asemenea dreptunghi. Se afișează drept rezultat :

- aria secțiunii diafragmei ;
- momentul de inerție ;
- coordonatele centrului de greutate.

```

10 PRINT "CALCULUL SECȚIUNII ELEMENTELOR "
15 PRINT "DE CONSTRUCȚII "
20 PRINT "DATE NR.DREPTUNGHIDRI ELEMENTARE "
30 INPUT N
40 DIM B(N),X(N)
50 PRINT "DATE LATIMEA,INALTIMEA IN CM "
60 FOR J = 1 TO N
70 INPUT B(J)
80 INPUT H(J)
90 NEXT J
100 B = 0
110 H = 0
120 A = 0
130 I = 0
140 FOR J = 1 TO N
150 X = B(J)*H(J)
160 L = X+H(J)/2
170 D = (A+D+K+L)/(A+K)
180 A = A+K
190 I = I+(H(J)^3*B(J))/12
200 H = H(J)+H
210 NEXT J
220 H = 0
230 FOR J = 1 TO N
240 L = H+H(J)/2
250 I = I+B(J)*H(J)*(L-D)^2
260 H = H+H(J)
270 NEXT J
280 PRINT "ARIA = "A;" CM^2 "
290 PRINT "MOMENT,INERTIA = "I;" CM^4 "
300 PRINT "IS = "D;" CM "
310 PRINT "HD = "H-D;" CM "
320 STOP
330 END

```

11.12. Determinarea momentelor de incastrare perfectă ale unei grinzi de beton armat

Se consideră o grindă de beton supusă atât unei forțe uniform distribuite, cât și unui număr oarecare de forțe concentrate. Se determină valoarea momentului de incastrare stînga și dreapta. Programul solicită ca date inițiale : lungimea grinzii, valoarea forței uniform distribuite, numărul total de forțe concentrate iar pentru fiecare forță concentrată, valoarea ei și distanța față de limita stîngă a grinzii. Se determină valorile celor 2 momente de incastrare și se afișează.

```

10 PRINT "MOMENT DE INCASTRARE PERFECTA "
20 PRINT "LUNGIMEA GRINZII IN M "
30 INPUT L
40 PRINT "FORȚA UNIFORM DISTRIBUITA IN KGF/M "
50 INPUT P
60 T = P*L^2/12
70 R = T
80 PRINT "NR.-UL FORTELOR CONCENTRATE "
90 INPUT N
100 IF N = 0 THEN 200
110 DIM F(N),X(N)
120 PRINT "MARIMELE SI DISTANTELE FORTELOR "
125 PRINT "CONCENTRATE PATA DE STINGA GRINZII "
130 FOR J = 1 TO N
140 INPUT F(J)
150 INPUT X(J)
160 IF X(J) > L THEN 290
170 NEXT J
180 FOR J = 1 TO N
190 A1 = X(J)/L
200 A2 = L-A1
210 U = A1^4-2*A1^3+L
220 T = T+V
230 V = A1^2*A2^2+F(J)*L
240 R = R+V
250 NEXT J
260 PRINT "MSL = "T;" KGF.M "
270 PRINT "MSR = "R;" KGF.M "
280 STOP
290 PRINT "ABSCISA GRESITA "
300 STOP
310 END

```

11.13. Optimizarea consumului de îngrășăminte chimice în agricultură

Se pornește de la funcțiile de producție pentru terenuri irigate și neirigate :

$$Y_{\text{IRIGAT}} = K_0 a_0 + K_1 a_1 dN - K_2 a_2 dN^2$$

$$Y_{\text{NEIRIGAT}} = a_0 + a_1 dN - a_2 dN^2$$

unde :

dN este doza de îngrășămintă la hectar

a_i sînt parametrii funcției de producție

K_i sînt coeficienți de multiplicare

```

10 PRINT " CONSUMUL DE ÎNGRĂȘĂMINTE
20 PRINT " CHIMICE ÎN AGRICULTURĂ "
30 PRINT " DATE PARAMETRII A0,A1,A2 "
35 PRINT " A FUNCȚIILOR DE PRODUCȚIE "
40 INPUT A0,A1,A2
50 PRINT " DATE COEFICIENȚII K0,K1,K2 "
55 PRINT " DE MULTIPLICARE "
60 INPUT K0,K1,K2
70 N1 = A1*(2*A0/A2)
80 N2 = K1*(A1/12+K2*A0/A2)
90 N = N2-N1
100 INPUT
110 PRINT AT(1,1); " FUNCȚIA DE PRODUCȚIE
120 PRINT AT(2,1); " NEIRIGAT " :
130 PRINT AT(3,1); " IRIGAT " :
140 PRINT AT(4,1); " ÎNREGISTRAT " :
150 PRINT AT(5,1); " DOZA MAXIMĂ DE NEIRIGAT " :
160 PRINT AT(6,1); " DOZA MAXIMĂ DE IRIGAT " :
170 PRINT AT(7,1); " DIFERENȚA " :
180 Y1 = A0+A1*N1-A2*(N1^2)
190 Y2 = K0+A1*N2-K2*(N2^2)
200 Y = Y2-Y1
210 PRINT AT(8,1); " MAXIMUL TEHNIC NEIRIGAT " :
220 PRINT Y1
230 PRINT AT(9,1); " MAXIMUL TEHNIC IRIGAT " :
240 PRINT Y2
250 PRINT AT(10,1); " DIFERENȚA " :
260 X1 = Y1-A0
270 X2 = Y2-K0
280 X = X2-X1
290 PRINT AT(11,1); " SPORUL MAXIM NEIRIGAT " :
300 PRINT X1
310 PRINT AT(12,1); " SPORUL MAXIM IRIGAT " :
320 PRINT X2
330 A0 = INKEY $
340 IF A0 = " " THEN 360
350 INPUT
360 WINDOW -10,0,30,10
370 MOVE -10,0
380 DRAW N2+30,0
390 MOVE 0,10
400 DRAW 0,Y2+10
410 FOR I=25 TO 25+INT((N2-30)/25) STEP 25
420 MOVE I,-50
430 DRAW I,90
440 NEXT I
450 D = INT((Y2+10)/1000)
460 FOR I=1000 TO 1000+INT((Y2-0)/1000) STEP 1000
470 MOVE -30,I
480 DRAW 50,I
490 NEXT I
500 N = A0
510 D = A1
520 C = A2
530 FOR K = 1 TO 2

```



```

490 MOVE 0,0
500 FOR I=10 TO 100 IN 10 (N2=301/10) STEP 10
510 Y = A+E*(C-I*2
520 DRAW I,Y
530 NEXT I
540 A = X0*AO
550 B = X1*A1
560 C = K2*A2
570 NEXT K
580 MOVE N1,0
590 DRAW N1,Y1
600 MOVE N2,0
610 DRAW N2,Y2
620 AS > INKEYS
630 IF AS = " " THEN 630
640 REM " CITIRE CONSTANTE "
650 READ N,D,C,B,E,F
670 INITP
680 DIM N(10,2)
690 MAT M = ZBK
700 Z = 10
800 X = 0
810 WINDOW -10,300,-30,10000
820 MOVE -10,0
830 DRAW 300,0
840 MOVE 0,-10
850 DRAW 0,10000
860 FOR I = 1 TO 180 STEP 25
870 MOVE I,-50
880 DRAW I,50
890 NEXT I
900 FOR I=1000 TO 8000 STEP 1000
910 MOVE -50,I
920 DRAW 50,I
930 NEXT I
940 FOR Z = 10 TO 100 STEP 10
950 X = 0
960 GOSUB 2000
970 U = Z/10
980 MOVE X,Y
990 N(U,1) = X
1000 N(U,2) = Y
1010 FOR X = 10 TO 180 STEP 10
1020 GOSUB 2000
1030 DRAW X,Y
1040 IF Y < N(U,2) THEN 1070
1050 N(U,1) = X
1060 N(U,2) = Y
1070 NEXT X
1080 N(12,10) = N
1090 NEXT Z
1100 MOVE N(1,1),N(1,2)
1110 FOR J = 2 TO 10
1120 DRAW N(1,1),N(1,2)
1130 NEXT J
1140 STOP
2000 B = A*(Z*D)
2010 V = B*(Z^E)*X
2020 U = C*(Z^F)*(X^2)
2030 Y = 0+V-U
2040 RETURN
2050 DATA 36,1,17,11,0,796
2060 DATA 1,1087,0,11858,-0,60889
2070 END

```

Se determină doza maximă de îngrășămint pentru teren neirigat și teren irigat. Se determină apoi maximul tehnic de producție și sporul maxim în cele două condiții de lucru.

Se vizualizează graficul funcțiilor de producție în condițiile de irigare și neirigare (coord. x=doze de îngrășămint în kg/ha; coord. y=producția medie în kg/ha). Particularizînd pentru funcțiile de producție caracteristice culturilor de grâu, avem (vezi liniile 2050 și 2060 din program):

$$Y = 36,1 NB^{1,1087} + 17,11 NB^{0,11858} dN - 0,796 NB^{-0,60889} dN^2$$

În baza acestei formule se trasează graficul producțiilor medii de grâu, în funcție de notele de bonitare ale terenului agricol.

Modificarea liniilor 2050, 2060 permite calculul funcției de producție pentru alte tipuri de culturi.

11.14. Calculul volumului rezervorului de compensație pentru rețeaua de apă potabilă

Variația consumului de apă pe parcursul unei zile se determină în baza debitului înregistrat la fiecare oră. Considerăm că alimentarea de la rețeaua de apă se face cu un debit mediu constant; în acest caz, rezervorul de compensație va trebui să acumuleze apa necesară consumului în orele de vîrf.

Programul solicită mai întîi introducerea valorilor celor 24 de debite orare. Se determină apoi debitul mediu, pe baza formulei:

$$D_m = \frac{1}{24} \sum_{i=1}^{24} D_i$$

Rezultă volumul rezervorului de compensație:

$$V_r = \sum_{i=1}^{24} (D_i - D_m) * ((SGN(D_i - D_m) + 1) / 2)$$

Se afișează consumul mediu și volumul rezervorului de compensație.

```

10 PRINT "REZERVORUL DE COMPENSAȚIE "
20 DIM Z(24)
30 PRINT "DAȚI CONSUMURILE PE ORE "
40 MAT INPUT Z
50 X = 0
60 FOR I = 1 TO 24
70 X = X + Z(I)
80 NEXT I
90 X = X / 24
100 U = 0
110 FOR I = 1 TO 24
120 IF Z(I) - X <= 0 THEN U = U + (Z(I) - X)
130 U = U - (Z(I) - X)
140 NEXT I
150 PRINT "CONSUM MEDIU :"; X
160 PRINT "VOLUMUL REZERVORULUI :"; U
170 STOP
180 END

```

11.15. Studiul unui filtru „trece-jos“

Fiind dat filtrul RC „trece-jos“ din figura 11.3, ecuația va fi

$$U_1(t) = U_2(t) + RC \frac{dU_2}{dt}$$

Ecuația poate fi aproximată cu:

$$U_1(n) = U_2(n) + \frac{RC}{\Delta t} [U_2(n) - U_2(n-1)]$$

unde $RC/\Delta t = T$ este constanta de timp normată a filtrului. Rezultă relația de recurență:

$$U_2(n) = \frac{U_1(n)}{1+T} + \frac{T}{1+T} U_2(n-1)$$

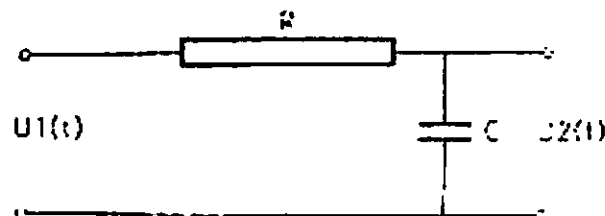


Fig. 11.3. Filtru RC „trece-jos”.

Forma semnalului de intrare este definită în cadrul programului în liniile 150—180, obținându-se la ieșire un semnal ca în figura 11.4. Programul solicită introducerea constantei de timp normale și a valorii inițiale a semnalului de ieșire U_2 . Se trasează graficul suprapus a celor două semnale, obținându-se diferite forme ale semnalului U_2 în funcție de constanta de timp T ; regimul tranzitoriu poate fi observat cu ușurință pe grafic.

```

10 PRINT "STUDIUL FILTRULUI : "
20 PRINT "DATE CARACTERISTICE : "
30 INPUT T
40 PRINT "DATE VALORILE INITIALE U2 = "
50 INPUT U2
60 INITP
70 WINDOW -10,20,-1,2
80 MOVE -10,0
90 DRAW 50,0
100 MOVE 0,-1
110 DRAW 0,2
120 U1 = 1
130 MOVE 0,U1
140 FOR X = 1 TO 50
150 IF SIGN(-INITP/T)*10<10-3 *- C THEN P=0
160 U1 = 0
170 GOTO 190
180 U1 = 1
190 DRAW X,U1
200 MOVE X,-1,U2
210 U2 = U1/(1+T*(57/(1+T)))*U2
220 DRAW X,U2
230 MOVE X,U1
240 NEXT X
250 STOP
260 END

```

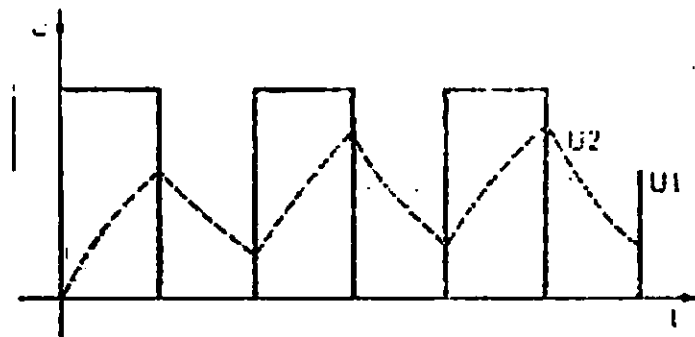


Fig. 11.4. Semnalul la ieșirea filtrului „trece-jos”.

11.16. Calculul salinității unui canal de ecluză

Programele rezolvă determinarea salinității apei într-un canal de ecluză ce leagă două ape cu salinități diferite. Se introduc datele inițiale și în diferite ipoteze de lucru se obține numărul de ecluzări necesare.

```

5 PRINT " P M Q R X 1 "
10 REM "CALCULUL SALINITĂȚII CANALULUI"
15 REM "IPOTEZA 1 = DIFERENȚA V > 0"
20 REM "CITIRE DATE INITIALE"
30 READ C1,C2,C3,D,M1,M2,A,Y,E
35 REM "CALCUL COEFICIENȚI"
40 D1=D/Y
50 C1=D/(A*M1)
60 D3=D/(A*(M1+M2))
65 REM "CONSTANȚE"
70 X1=D3
80 Y1=D2
90 Z=1/(X+Y)
95 REM "SALINITATE ADMISA S A S 0 0"
100 Q=C1*(1-Z)+C2/Z+C3*(1-D3)*Y
110 B=(C3*C2-C1*(C2+C3)+Y*C1)*Z
120 X=A/Z
130 Q=X*A-B
220 Z=X-SDR(D)
225 REM "SALINITATE INITIALA S A S = M"
230 Y=D*(C2+D3)+Y*D3+Y*C3
235 REM "ADINCIMEN AEMENENT"
240 X=X*(C3-C1)/(C2-C1)
245 REM "VARIATIA SALINITATII"
250 N=0
260 I=C1
270 M=C3
280 H=X+1
290 V=H*(1-M1)/X+M
300 I=D*(C2-D3)+Y*(V+M)+V*Y
310 M=D1+V*(1-D1)*M
315 REM "VERIFICARE LIMITE"
320 IF I>=M+E THEN 360
330 IF Y>=D+E THEN 360
340 IF H>=C3+E THEN 360
350 GOTO 280
360 PRINT " NUMARUL DE FCLIZARI = "N
370 STOP
375 REM "DATE INITIALE"
380 DATA 0.15,17,0.5
390 DATA 21000,7,0
400 DATA 0000,30,000
410 DATA 0.01
3660 END

5 PRINT " P A C R W 2 "
10 REM "CALCULUL SALINITATII LAMINARULUI"
15 REM "IPOTEZA 2 = DIFERENȚA V > 0"
20 REM "CITIRE DATE INITIALE"
30 READ C1,C2,C3,D,M1,M2,A,Y1,Y2,E
35 REM "CALCUL COEFICIENȚI"
40 U1=D/Y1
50 D2=D/(A*M1)
60 D3=D/(A*(M1+M2))
65 D4=V2/Y2
70 D5=(1-D2)*(1-U1)
75 S=(1-D5)*(C3-D4+C1)/(D1*(1-D4))
80 S1=D2*(C2+D3*(1-D1))+D3*(1-D2)*C3
90 R=(1-D5)*(1-D3)*(C3-C1)/(D1*(1-D4)+(C2-C1))
95 REM "VARIATIA SALINITATII"
100 N=0
110 I=C2
120 M=C3
130 W=C1
140 V=R*(1-W2)/(1-D1)*M2
150 I=D2*(C1-D5)*(1-D1)+Y*(V+M1)+V*(1-D2)
160 W2=(1-D4)*(D1*(1-D5)+M1)+D4*(1-D1)
170 M1=D1*(1-D1)*M2
175 REM "VERIFICARE LIMITE"
180 IF W<100 THEN 210
190 IF I>=M+E THEN 220
200 IF V>=D+E THEN 220
210 IF W2>=C3-E THEN 220
220 GOTO 130
230 REM "AJUȚ LA LIMITE"
240 PRINT N,I,V,W
250 STOP
290 REM "DATE INITIALE"
300 DATA 0.15,17,0.5
310 DATA 21000,7,0
320 DATA 0000,30,000,10,000
330 DATA 0.01
340 END

```

```

5 PRINT " P P G R M 3 "
10 REM "DISPERIȘUTIA GALINITATII IN CANAL"
15 REM "INTECIIA I = DIFERENȚIA V = 0"
20 REM "CITISE DATE INITIALE"
30 READ C1,C2,C3,U,M1,M2,A,V,L,T,V1,E,M1,M2
35 REM "CALCUL COEFICIENTI"
40 D1=C/V
45 D2=D/(A+M1)
50 D3=D/(L*(M1+M2))
55 D4=L/V3
60 D5=D4/V1
65 DIM K(250),L(250)
70 Y=1-D2
80 X=1-D3
90 Z=1/(K*Y)
95 REM "SALINITATEA ADMISA IN S A S = 0"
100 A=C1*(1+Z)+C2*X+C3*(1-D1*Y)*Z
110 B=M1*C2-C1*(D2+C2+D3*Y+C3)*Z
120 U=1/2
130 H=U*U-D
140 S=U-SQR(H)
150 REM "SALINITATEA INITIALA S A S = M"
160 M=D2+C2+D3*Y+C3
170 REM "ADINCINEA NEMANENTA = M"
180 N=2*(S-C1)/(C2-C1)
190 REM "VARIETIA GALINITATII"
200 J=1
210 H=0
220 HAT K=Z*Z
230 HAT L=Z*Z
240 J=C2
250 H=C1
260 N=H+1
270 V=H*(1-N)/X+K
280 I=D2+C2+D3*Y+(V-N)*Y*Y
290 H=D1*Y*(1-D1)*M
300 IF H=1 THEN 310
310 K(J)=V
320 L(J)=H
330 J=J+1
340 IF J<=D3 THEN 310
350 H=L(J)
360 H=L(J)
370 H=L(J)
380 H=L(J)
390 IF J<=D3 THEN 310
400 H=L(J)
410 H=0
420 FOR J=2 TO D5
430 H=L(J)-J*10/(V*(D3-J+1))*K(J)-M
440 IF H=0 THEN 450
450 H=0
460 H=L(J)
470 H=L(J)
480 H=L(J)
490 H=L(J)
500 GOTO 250
510 STOP
520 REM "DATE INITIALE"
530 DATA 0.15,17,0.5
540 DATA 21000,T,0
550 DATA 0000,30,0E0
560 DATA 5000,3000,1
570 DATA 0.01,000,100
580 END

```

11.17. Calculul hidraulic al ecluzelor

Programele determină caracteristicile hidraulice principale ale unei ecluze, funcție de condițiile de evacuare a apei.

```

5 PRINT " P R O G R A M S "
10 REM "CALCULUL HIDRAULIC AL ECLUZELOR"
15 REM "CALCULARE IN SIEF. VAL"
20 REM "DATE INITIALE"
30 READ A3,H,A1,C1,M1,T1,P,C2,R2,T2,V,E
35 DIM
40 I=0
45 I=0
50 I=0
55 I=0
60 I=0
65 L=0
70 N=0
75 U2=C2
80 N=(A1/A3)+0.01
85 U=(A1/A3)+0.01
90 U=0
95 U=0
100 I=0
110 I=0
120 I=0
130 IF T>N1 THEN 210
140 M=I*(C1/M1)
150 U1=(L+M)/2
160 L=M
170 C2=C2
200 GOTO 170
210 IF I<=0 THEN 250
220 U1=C1
230 GOTO 040
240 GOTO 130
250 L=C2
260 T=(T2-T1)+C2/T1
270 M=(T2-T1)+C2/T1
280 U1=L
290 T=T1
300 IF T>N1 THEN 320
310 L=0
320 GOTO 040
330 T=0
340 REM "SUBPROGRAM CALCUL ITERATIV BANCINA"
350 M=U1*(C1-M)/2-U1*(C1-M)/2
360 IF M<0 THEN 400
370 M=0
380 GOTO 400
390 M=0
400 REM "DIFFERENTA DE BANCINA"
410 DIM
420 REM "OCII AFLUENT"
430 U2=U2+0.001*(U1-U2)
440 REM "OCII EFLUENT"
450 U3=U3+0.001*(U1-U3)
460 REM "VOLUMUL EVACUAT"
470 V=V+0.001*(U1-U2)/2
480 I=I+1
490 I=0
500 REM "BANCINA ECLUZE"
510 I=I+1
520 PRINT I,U2,U3,C2,D
530 U1=U2
540 IF I<=0 THEN 550
550 I=0
560 RETURN
570 REM "DATE INITIALE"
580 DATA 2000,1.75
590 DATA 0.41,0.5,100,120
600 DATA 0.75,0.65,360,120
610 DATA 11000,0.01
620 END

```

```

5 PRINT " P R O G R A M "
10 REM "CALCUL HIDRAULIC AL ECULUIZELOR"
15 REM "EVACUARE ÎN BIEF AVAL"
20 REM "CONDITIE STATIONARE S A S"
25 REM "CITIRE DATE INITIALE"
30 READ A3,M,A1,C1,M1,T1,A2,C2,U,E
40 DIM
45 I=0
50 I=0
55 T=0
60 T=0
65 J=0
70 L=0
75 B=RR(2*Q,01)
80 U2=C2
85 N=(A1/A3)*B=J
90 D=(A2/A3)*D=J
95 D1=0
100 P=0
105 Z=0
110 I=I+1
115 T=T+J
120 I=T+J
125 IF I>M1 THEN 210
130 M=(0.5*C1/R)
135 U1=(L*V)/2
140 L=M
145 GOSUB 000
200 GOTO 135
210 U1=C1
215 U=0
220 C=U3-U2
230 T1=I
240 R=1+1.433R1*(P/2)-D+C2*30K*(M-(U*P/2))
250 Q=Q-H
260 Q3=C1+0.14H*55K(C)
270 M1=U3-C1/(1.4244*30K*(M-D))
280 C2=X
290 P=H
300 IF 0.09(C-P)>E THEN 240
310 GOSUB 000
320 I=I+1
330 T=I+J
340 IF C2>E THEN 240
350 U=0
355 R=209*(U1-C1+0.14H*(1+J-1.5))/(2443)
360 Q3=C1+0.14H*55K(R)
365 Q=Q-P
370 PRINT I,J,J,Q
375 I=I+1
380 T=I+J
385 IF D<E THEN 355
390 STOP
395 REM "SUBPROGRAM CALCUL ITERATIV SARCINA"
400 R=H*U1+54810*(P/2)-D+U2*30K*(M-(U*P/2))
410 IF ABS(R-P)<E THEN 450
420 P=R
430 GOTO 000
440 P=H
445 REM "DIFERENTA DE SARCINA"
450 Q=Q-P
455 REM "DEBIT AFLUENT"
460 Q2=U2+0.2444*30K*(M-D)
465 REM "DEBIT EFLENT"
470 Q3=U3+0.14H*55K(D)
475 REM "VOLUMUL EVACUAT"
480 V=(Q1+Q2)/2
490 P=1+J
495 REM "SARCINA ECULUIZELOR"
500 Z=Z+X
510 PRINT I,Z,U2,U3,C2,D
520 Q1=Q
530 RETURN
540 REM "DATE INITIALE"
545 DATA 2300,1.75
550 DATA 0.41,0.5,100,120
555 DATA 0.75,0.85
560 DATA 1100,0.01
565 END

```

```

5 PRINT " P R O G R A M "
10 REM "CALCUL HIDRAULIC AL RECUZEE"
15 REM "EVACUARE IN BAZIN SUBTERAN"
20 REM "CITIRE DATE INITIALE"
30 READ A3,A1,H2,Z,M,S,C,A1,C1,R1,T1,W2,C2,R2,T2,V,C
35 REM "INITIALIZARI"
40 D1=M1
41 I=0
45 T=0
50 T3=0
55 T5=0
60 J=20
65 L=0
75 B=0.001(2+V,811)
75 X=0
80 N=(A1/A3)*B+J
85 O=(A2/A3)*B+J
90 Q1=0
95 P=0
100 H1=M1+M2+Z
105 F=0
110 D2=H2
115 Z=0
120 I=I+1
125 T=T+J
130 IF T>R1 THEN 200
135 M=I+J+C1/R1
140 V1=(L+P)/2
145 L=0
150 IF T>R2 THEN 190
155 Y=I+J+C2/R2
160 U2=(X+Y)/2
165 X=V
170 GOTO 80
175 GOTO 130
180 IF T3<=B THEN 230
185 U1=C1
190 U2=C2
195 GOTO 80
200 IF T3<=B THEN 230
205 U1=C1
210 U2=C2
215 GOTO 80
220 GOTO 130
230 X=C2
235 I=I3+T2
240 Y=(I4-I1)*J+C2/T2
245 U2=(X+Y)/2
250 I=I+1
255 T=T+J
260 IF T>T4 THEN 280
265 X=V
270 GOTO 80
275 GOTO 240
280 U=0
285 P=SUR(U)-C1+A1*B*(I+J+T)/(2*A3)
290 C1=C1+1+0.0001(S)
300 D1=D1+0.1
310 PRINT I,Q1,C1
315 I=I+1
320 IF D1>E THEN 295
325 GOTO 280
330 REM "SUBPROGRAM CALCUL ITERATIV BAZINA"
340 READ A1,A2,A3,H1,H2,H3,D1,D2,D3,P,F,Z
350 H3=(H1+C1(2A01+P))/(S+C*(2+D2+R1))
360 G2=0.42+0.0001(H1-(D1-H/2)-(U2-P/2))*J/M
370 IF ABS(F-H1)<=E THEN 460
375 M=H1
380 F=0
385 D=0.450
390 P=0
400 F=0
405 REM "DIFERENTE DE BAZINA"
410 L1=H1-R
415 C2=D2+0
420 REM "DEBIT EFLUENT"
425 U1=0.1+0.0001(D1)
430 Q1="DEBIT EFLUENT"
435 G2=0.42+0.0001(H1-D1-D2)
440 REM "VOLUM EVACUAT"
445 F=(C1+U2)/2
450 K=M+J
455 REM "BENZINA RECUZEE"
458 Z=Z+K
460 PRINT I,Z,Q1,C2,D1
465 Q1=Z
470 IF Z<=0.0+V THEN 580
475 T=T+J
480 RETURN
485 DATA 2000,1,75
490 DATA 7,1,5
495 DATA 7,0,10
500 DATA 0,41,0,5,188,120
505 DATA 0,75,0,65,388,120
510 DATA 11588,0,0,1
515 END
518

```


Încercările mai vechi de utilizare a calculatoarelor în procesul de învățămînt s-au bazat pe folosirea terminalelor de tip display, cuplate la un calculator central. Deși utile, sistemele nu s-au extins în învățămîntul liceal, chiar în țările avansate industrial, datorită costurilor lor ridicate.

Apariția calculatoarelor personale permite abordarea acestei probleme de pe noi poziții, deoarece aceste calculatoare pot fi folosite de către elevi, atât în școală, cît și la domiciliu. Facilitățile hardware și software de care dispun calculatoarele personale permit cuplarea lor la un calculator central, dotat cu o bază de date (cunoștințe). În acest context asistarea procesului de învățămînt de către calculator capătă noi dimensiuni.

12.1. Modalități de integrare a calculatorului în procesul de predare-învățare, locul și rolul acestuia în asistarea procesului de învățămînt

Asistarea procesului de învățămînt cu calculatorul. Această problemă este de dată recentă, dar de mare perspectivă și cu consecințe pozitive pentru optimizarea predării-învățării, pentru integrarea învățămîntului cu cercetarea, cu producția, cu viața.

Este cunoscut faptul că în ultimele două decenii, colaborarea dintre informaticieni, constructori de calculatoare și specialiști din domenii ale instrucției și educației a permis inițierea unor programe concrete privind folosirea calculatoarelor în procesul de învățămînt. Ca urmare, au fost concepute, realizate și perfecționate variate limbaje de programare, cu ajutorul cărora să se poată edifica autoinstruirea automată, să fie prezentate la clasă lecții, ori secvențe ale acestora, în sprijinul optimizării procesului de predare-învățare. În consecință, psihopedagogicii i-a revenit sarcina edificării unei metodologii care să asigure eficiența asistării procesului de învățămînt cu calculatorul.

Conceptul de asistare cu calculatorul a procesului de învățămînt include atât predarea cu calculatorul a unei lecții de comunicare a noilor cunoștințe, de aplicare, de consolidare ori de sistematizare a acestora, cît și verificarea

* Cu orientare spre învățămîntul mediu.

automată a unei lecții sau a unui grup de lecții, a unei anumite discipline școlare la sfârșitul unui trimestru sau an școlar, a unei anumite programe școlare în cadrul examenelor de absolvire sau de admitere. Tot în cadrul „asistării” este inclusă și prezentarea sau verificarea cu calculatorul a unor secvențe ale lecțiilor desfășurate după metodologia clasică.

Calculatorul devine un auxiliar al procesului de învățământ.

Tehnica instruirii cu calculatorul. Instruirea cu ajutorul calculatorului se realizează fie în cadrul așa-ziselor „clase automatizate”, în care lecția se derulează secvență cu secvență pe ecranul de la pupitrul fiecărui elev, fie în cadrul laboratoarelor de matematică — mai ales, pentru prezentarea de la catedră, prin intermediul ecranului, a unor secvențe ale lecției clasice. O clasă automatizată dispune de un calculator și mai multe terminale (în jur de 30) care recepționează mesajele pe ecranul individual.

Predarea lecției prin intermediul ecranului se realizează printr-o succesiune de imagini, numite „imagini-ecran”, care afișează lecția, derularea lor fiind dirijată de către profesor — în concordanță cu normele psihopedagogice ale procesului de instruire la vârsta și în cadrul obiectului predat.

Instruirea cu ajutorul calculatorului se poate însă realiza și în afara schemei clasice a procesului de învățământ. Elevul, studentul, sau orice altă persoană interesată poate beneficia, într-un anumit cadru, de posibilitatea autoinstruirii automatizate. Aceasta se realizează fără profesor, activitatea sa fiind însă materializată în lecțiile afișate pe ecran. Dialogul între mașină și cel care învață singur se realizează cu ajutorul unei console legate de calculator, aceasta fiind construită dintr-o claviatură alfanumerică și din ecranul semnalat mai sus.

Lecția, sau grupul de lecții, sînt stocate în memoria calculatorului și cu ajutorul software-ului sînt afișate treptat, imagine cu imagine, pe ecranul consolei, furnizînd astfel materialul de învățat.

În cadrul acestei instruirii elevul își poate regla singur ritmul de afișare a imaginilor pe ecran; el își poate întrerupe pregătirea în orice moment al lecției, reluînd-o de la secvența la care s-a oprit, pe baza „recunoașterii” sale de către calculator. Această posibilitate de „recunoaștere” permite folosirea în timp a aceluiași lecții de către mai mulți elevi.

În lecțiile predate cu ajutorul calculatorului sînt prevăzute și exerciții de control, răspunsurile putînd fi adeseori selectate dintr-o listă de variante afișată pe terminal. Metoda folosită în acest caz este, oarecum, similară învățămîntului programat. În cazul că lecția urmărește testarea cunoștințelor, atunci fiecărei imagini-ecran i se asociază una sau mai multe întrebări. Răspunsurile la întrebări sînt cotate cu un anumit punctaj, acesta depinzînd de calitatea răspunsului și de timpul în care a fost dat.

În dialogul dintre mașină și cel care învață se folosește și funcția de „help” a calculatorului. Prin aceasta sînt puse la dispoziția utilizatorului, sub formă de imagini-ecran, comentariile și explicațiile necesare integrării și însușirii corecte a unor noțiuni — fără de care lecția nu poate continua. Evidențiem, cu acest prilej, „posibilitatea” mașinii de a detecta imediat eroarea în însușirea unei secvențe a lecției și de a facilita remedierea acesteia prin comenta-

riul făcut și indicarea informațiilor suplimentare necesare continuării instruirii. Cel care învață poate trece la secvența următoare a lecției numai dacă a răspuns corect la întrebările sau exercițiile propuse, interacțiunea dintre utilizator și sistem realizându-se permanent pe bază de dialog.

Calculatorul didactic poate fi înzestrat și cu „posibilitatea” de a încorpora performanțele celui ce învață: timpul necesar pentru răspuns, nota obținută, întrebările la care frecvența răspunsurilor slabe este mare etc. În acest mod sînt furnizate fie elevului, fie profesorului, o serie de date pedagogice individuale, sau caracteristice unei anumite populații școlare, care sînt necesare optimizării acestor programe și lecțiilor recapitulative sau de control curent ce se vor organiza.

Consecințe pedagogice ale asistării lecțiilor cu calculatorul. Folosirea calculatorului în școală oferă învățării lecției noi posibilități de dezvoltare și de evaluare. Așa, spre exemplu, prin intermediul consolei pot fi simulate pe ecran procese și fenomene în evoluția lor, unele experiențe greu accesibile laboratoarelor școlare fie datorită costului lor prea ridicat, fie pericolelor existente, protecției muncii și a mediului înconjurător, fie depărtării acestora în timp sau în spațiu etc.

Ca o nouă componentă a tehnologiei învățămîntului, folosirea calculatoarelor în munca instructiv-educativă va lărgi aria de posibilități și de funcționalitate a laboratoarelor școlare, în beneficiul tuturor disciplinelor de învățămînt care folosesc tehnică de calcul sau de reprezentare.

Cu ajutorul calculatorului poate fi optimizat randamentul predării prin prezentarea cu ajutorul ecranului a unei largi varietăți de exemple sau de modele asociate unor secvențe ale lecției. Toate acestea concurează la adîncirea sau lărgirea orizontului noțiunilor predate, adeseori extrapolîndu-le dincolo de obiectul predat: în tehnică, în economie, în știință, în practică etc. În acest mod s-ar putea vorbi despre o „dilatare” a sferei aplicative a noțiunilor predate, precum și de o „comprimare” a timpului necesar însușirii și aplicării creatoare a noțiunilor științei. Acest aspect poate conduce la stimularea inventivității și aplicativității, a spiritului participativ și anticipativ al celui ce învață.

Folosirea calculatoarelor ca mijloace de învățămînt va avea, desigur, consecințe importante asupra formării intelectuale a tinerelor generații în spiritul autoeducației — și, prin aceasta, al educației permanente. În această activitate formativă vor fi stimulate toate componentele gândirii logice în sens larg: gândirea logico-deductivă, inductivă, analogică, ... cu accent pe gândirea euristică. Va fi astfel stimulată mai bine și problema orientării profesionale a tineretului, capacitatea acestuia de reciclare rapidă pentru sectoare de muncă „înrudite” profesional. Fără îndoială, pentru atingerea acestui țel, considerăm că nici disciplinele tehnologice și nici atelierul școlar, prin și în care elevii trebuie să învețe să producă la nivelul tehnicii contemporane, nu trebuie să rămînă în afara folosirii calculatorului și a tehnicilor de calcul ale producției moderne. Prin aceasta s-ar putea realiza mai bine și componenta tehnică a educației multilaterale a personalității umane.

Desigur, progresele realizate astăzi pe linia dezvoltării și a ieftinirii circuitelor integrate, între care microprocesoarele ocupă un loc important, vor per-

mite construirea pe scară tot mai largă a unor echipamente așa-zise de "uz personal", numite și calculatoare personale (individuale). Acestea pot fi folosite și în procesul de învățământ, intrând — pentru început — în zestrea laboratoarelor de matematică (de ex.), în calitate de auxiliar al predării, al verificării cunoștințelor, ..., al învățării.

Dar pentru integrarea calculatorului în familia mijloacelor de învățământ, profesorului i se cere să stăpânească, în afara specialității sale, un volum apreciabil de cunoștințe din domeniul informaticii, un limbaj de programare a lecțiilor sale, cum și o mare varietate de tehnici pentru realizarea desenelor și a exercițiilor din lecțiile sale curente, de sinteză sau de control etc. Se înțelege de aci necesitatea acestor cunoștințe, cum și a unor deprinderi tehnice, în pregătirea generală a întregului personal didactic din învățământul secundar. Acestea se pot obține pe bază unor programe speciale, atât în pregătirea universitară a cadrelor cât și în perfecționarea postuniversitară. Mai mult, pentru folosirea calculatorului va fi necesară o pregătire generală a întregului tineret — în cadrul noilor programe ale învățământului liceal. Fără îndoială, este necesar ca — în perspectivă — să dispunem de limbaje de programare foarte simplificate, ameliorând astfel eforturile utilizatorilor pentru însușirea lor.

Trebue reamintit că aici pentru elev nu este chiar atât de simplă și de ușoară pregătirea individuală cu ajutorul calculatorului. Și acestuia îi este mai ușor dacă poate primi cunoștințele în limba sa proprie, dacă poate "conversa" cu mașina într-un limbaj natural — nu artificial. În plus, mașina care servește autoeducației automatizate nu poate dispune de prea multe mijloace pedagogice, în afara celor ce decurg din evidențierea reușitei la învățare, sau a seninătății însoțitoare și trimiterea la întrebări și materiale ajutătoare.

Este de apreciat faptul că, însușindu-și cunoștințele la fața consolei, elevul învață singur și sigur, adeseori în ritm optim propriu, fără emoții și fără perturbări ale comportamentului de către diverșii factori legați de mediul său înconjurător. El este scutit de emoțiile care apar atunci când se vede "urmărit" de către profesor, sau de către colegii săi de clasă (deși, aceste emoții pot avea frecvent și un rol pozitiv asupra personalității sale).

De asemenea, în fața consolei elevul primește obiectiv și cu optimism "nota" acordată răspunsurilor și drept pregătirii sale; învață de aici să-și aprecieze calitatea și durata pregătirii, performanțele atinse și, mai ales, învață să-și mobilizeze resursele de energie și de voință pentru o nouă calitate a omniei sale.

După cum se vede, științei învățării i se circumscrie un nou domeniu în care cercetarea pedagogică interdisciplinară va fi solicitată foarte mult. Absența acesteia poate frâna accesul calculatoarelor în domeniul învățământului.

Încă de pe acum au apărut o serie de întrebări în legătură cu așa-zisa "înstrăinare" a individului ca urmare a unui învățământ automatizat.

Nei considerăm că un învățământ complet automatizat nu poate exista, problema "înstrăinării" pierzându-și astfel suportul pe baza căruia a fost lansată. Practica va fi, desigur, în favoarea folosirii tehnologiilor celor mai avansate și în domeniul învățământului, atât dezvoltarea tehnicii cât și a științei educației fiind strâns legate de dezvoltarea întregii societăți. Or, societatea contribuie

nu numai la umanizarea tot mai înaltă a personalității umane, ci și a tehnicii, a matematicii chiar, a mijloacelor de învățămînt și educație. Mai mult, există și o întoarcere "reflexivă": tehnica, matematica, ..., au la rîndul lor un rol tot mai mare în umanizarea omului.

12.2. Modele de lecții sau secvențe ale acestora pentru instruirea asistată de calculator (IAC) a matematicii în liceu

Tema. Rezolvarea sistemelor de două ecuații de gradul I cu două necunoscute (lecție de recapitulare ținută în laboratorul de matematică la începutul clasei a IX-a)

În această lecție se urmărește ca elevii să alcătuiască corect programul de rezolvare a unui sistem de două ecuații de gradul I cu două necunoscute și schema logică a rezolvării sistemului; de asemenea, ei trebuie să se convingă de necesitatea folosirii calculatorului la rezolvarea unor sisteme care necesită un volum mare de calcul.

1° Programul de rezolvare a sistemului:

Pe baza recapitulării cu clasa, a principalelor noțiuni și metode privind rezolvarea sistemelor de două ecuații de gradul I cu două necunoscute, se stabilește programul de rezolvare a sistemului

$$\begin{cases} A \cdot X + B \cdot Y = C \\ D \cdot X + E \cdot Y = F \end{cases}$$

Astfel se notează: $M = A \cdot E - B \cdot D$, $M_1 = C \cdot E - F \cdot B$, $M_2 = A \cdot F - C \cdot D$ și se face discuția de mai jos:

dacă $M = 0$ și $M_1 \neq 0$, atunci sistemul este incompatibil;

dacă $M = 0$ și $M_1 = 0$, atunci sistemul este compatibil nedeterminat și are o infinitate de soluții;

dacă $M \neq 0$, atunci sistemul este compatibil determinat și are soluțiile

$$X = \frac{M_1}{M}, \quad Y = \frac{M_2}{M}.$$

2°. Schema logică

În perspectivă, noile programe de matematică vor beneficia — probabil de noțiunile privind schema logică, încă din clasele VII—VIII. În acest caz, fiind scema de program, se va alcătui cu clasa schema logică a rezolvării sistemului.

Observație. În cazul că programa nu prevede noțiunea de schemă logică, se va trece la rezolvarea unor sisteme de ecuații, verificându-se soluțiile cu calculatorul. (fig. 12.1)

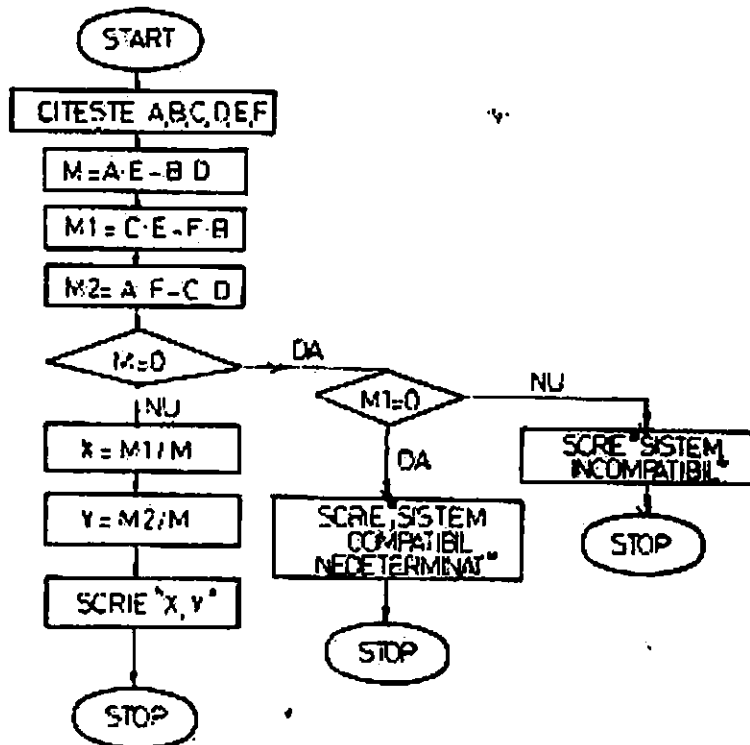


Fig. 12.1. Organigrama rezolvării unui sistem de două ecuații cu două necunoscute.

3°. Programul în BASIC

```

1 INPUT
5 PRINT "REZOLVAREA UNUI SISTEM DE
  DOUĂ ECUAȚII CU DOUĂ
  NECUNOSCUTE"
10 PRINT "DOUA ECUAȚII CU DOUĂ
  NECUNOSCUTE"
15 PRINT "NECUNOSCUTE"
20 PRINT "A * X + B * Y = C : D * X + E * Y = F"
30 INPUT A,B,C,D,E,F
40 M = A * E - B * D
50 M1 = C * E - F * B
60 M2 = A * F - C * D
70 IF M = 0 THEN 130
80 X = M1 / M
90 Y = M2 / M
100 PRINT "SOLUȚIILE SISTEMULUI"
110 PRINT "X=",X;"Y=",Y
120 GO TO 180
130 IF M1 < > 0 THEN 100
140 PRINT "SYSTEM COMPAT. NEDETERMINAT"
150 GO TO 180
160 PRINT "SYSTEM INCOMPATIBIL"
180 END.
  
```

4°. Aplicație pe calculator

Se cere rezolvarea sistemului :

$$\begin{cases} 5,025x + 375y = 721,851 \\ 105x + 3,25y = 163,9348 \end{cases}$$

Practic, o parte din elevi vor da soluții greșite în rezolvarea acestui sistem. Soluția este : $x=1,5024$; $y=1,9024$

Se trece apoi la rezolvarea cu calculatorul a sistemului.

Calculatorul „AMIC” oferă soluția: $x=1,50230$; $y=1,90239$.

În final, profesorul se va opri asupra unuia din obiectivele acestei lecții de recapitulare și anume acela al necesității alcătuirii schemei logice — ca dovadă a unor priceperi și deprinderi trainice în rezolvarea sistemelor de două ecuații cu două necunoscute. De asemenea, pe baza rezolvării ultimului sistem, elevii se conving că este necesară folosirea calculatorului pentru rezolvarea unora din sistemele cu coeficienți mai „complicați”. Ei vor fi solicitați să ofere calculatorului, spre rezolvare, o serie de sisteme compatibile sau incompatibile, să compare timpul lor de rezolvare cu timpul necesar calculatorului etc. Pe această bază apare și dorința însușirii unui limbaj de programare în vederea accesului lor la calculatorul școlii.

Tema. Funcția (secvența dintr-o lecție de sinteză de la sfîrșitul clasei a IX-a — ca toate serile ca o lucrare de control al cunoștințelor recapitulate). Prin intermediul consolei se proiectează pe ecran funcția

$$f: (-5, 0] \rightarrow (-4, 4]$$

și ecranul grafic este dat în figura 12.2 și se cere să se scrie:

- 1°. mulțimea valorilor funcției;
- 2°. lungimea fiecărui segment al figurii (în ordinea de pe figură);
- 3°. punctul fiecăreia din dreptele ce conțin segmentele figurii (în ordine);
- 4°. ecuațiile dreptelor ce conțin segmentele figurii (în ordine);
- 5°. dacă funcția este injectivă sau surjectivă;
- 6°. mulțimea valorilor argumentelor pentru care se anulează funcția;
- 7°. mulțimea valorilor argumentelor pentru care $f(x) > 0$ (respectiv $f(x) < 0$);
- 8°. mulțimea pentru care funcția este strict crescătoare (respectiv strict descrescătoare);
- 9°. reprezentarea în același sistem a funcțiilor date de $y = f \circ x_a$ și $y = f \circ x_b$, unde f este funcția dată inițial.

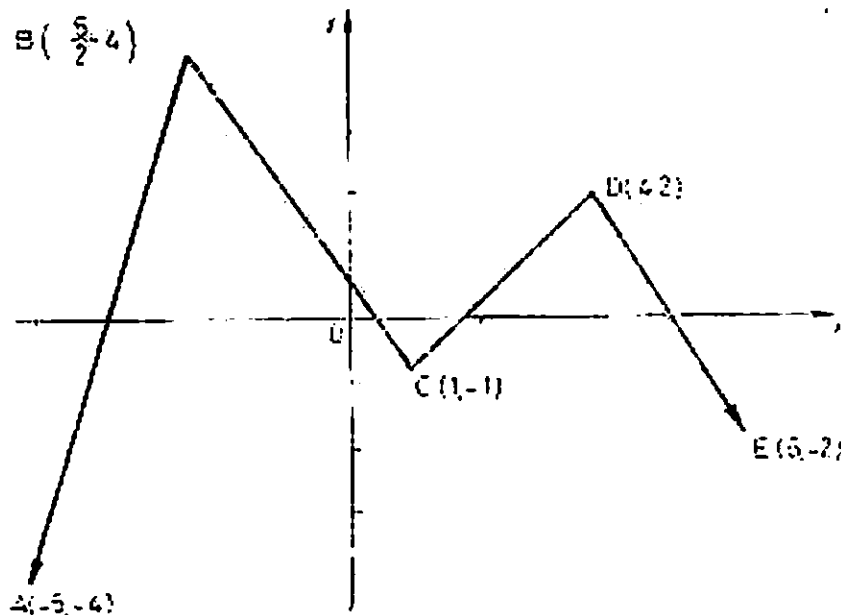


Fig. 12.2. Funcție dată sub formă grafică.

Observație. Pentru fiecare din întrebările puse (excepând 9^o) sînt atașate cîte două răspunsuri din care unul este corect și nu prea greu de sesizat. Pentru (9^o), explicațiile suplimentare conțin trei sisteme de axe de coordonate: primul reprezentînd $y=f(x)$ și $y=|f(x)|$, al doilea pentru $y=f(x)$ și $y=-|f(x)|$, iar al treilea conține toate cele trei grafice în ordine (linie plină, punctat și liniște). Elevul îl subliniază cu o linie pe cel corect și predă lucrarea profesorului. În tema de acasă, elevul trebuie să răspundă la întrebările de mai sus, motivînd care sînt elementele de care n-a ținut seama în cazul răspunsurilor incorecte. Acest mod de lucru îl va stimula să înțeleagă funcția de „help” a calculatorului, construind el însuși „comentariul și trimiterea la material auxiliar”

Tema. Funcții putere de forma

$$f: \mathbb{R} \rightarrow \mathbb{R}, f(x) = x^n, n \in \mathbb{N}$$

și funcții putere de forma:

$$f: \mathbb{R} \setminus \{0\} \rightarrow \mathbb{R}, f(x) = x^n, n \in \mathbb{Z}.$$

Reținem o secvență din prima parte a unei lecții de sinteză de la sfîrșitul clasei a IX-a, sau de la începutul clasei a X-a. Calculatorul este folosit atît de cîtrea profesor pentru antrenarea gîndirii „productivă” a întregii clase.

1^o. Prin intermediul consolei vor fi proiectate pe ecran cîteva grafice corespunzătoare lui $n \in \{0, 1, 2, 3, 4, 5\}$, în două etape:

1^o. Cazurile particulare (pentru $n \neq 0$ și pentru $n = 1$):

($n=0 \Rightarrow f(x)=1$) — dreapta de ecuație $y=1$, paralelă cu (Ox) , figura 12.3;

($n=1 \Rightarrow f(x)=x$) — dreapta de ecuație $y=x$, bisectoarea I, figura 12.4;

Se reține de aici ideea că graficele trec prin punctul $P(1, 1)$.

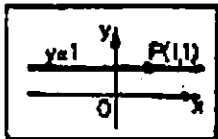


Fig. 12.3

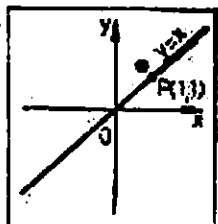


Fig. 12.4

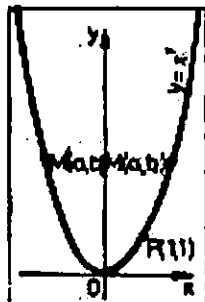


Fig. 12.5

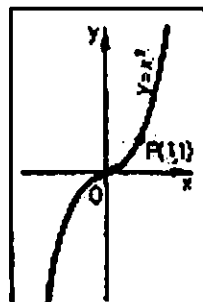


Fig. 12.6

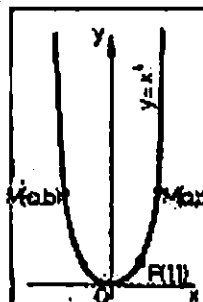


Fig. 12.7

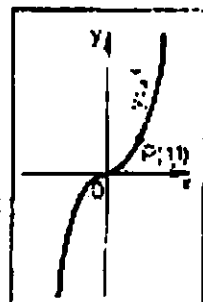


Fig. 12.8

Fig. 12.3. Funcția $y=1$.

Fig. 12.4. Funcția $y=x$.

Fig. 12.5. Funcția $y=x^2$.

Fig. 12.6. Funcția $y=x^3$.

Fig. 12.7. Funcția $y=x^4$.

Fig. 12.8. Funcția $y=x^5$.

2^o. În cea de-a doua etapă se proiectează graficele următoarelor funcții:

($n=2 \Rightarrow f(x)=x^2$) — parabolă de ecuație $y=x^2$, figura 12.5

($n=3 \Rightarrow f(x)=x^3$) — parabolă cubică de ecuație $y=x^3$, figura 12.6

($n=4 \Rightarrow f(x)=x^4$) — grafic de tip parabolă, avînd ecuația $y=x^4$, figura 12.7

($n=5 \Rightarrow f(x)=x^5$) — grafic de tip parabolă cubică, avînd ecuația $y=x^5$, figura 12.8

($n=2p, p \in \mathbb{N}, p \geq 1$) \Rightarrow ($f(x)=x^{2p}$) — grafic de tip parabolă, de ecuație $y=x^{2p}$;

($n=2p+1, p \in \mathbb{N}, p \geq 1$) \Rightarrow ($f(x)=x^{2p+1}$) — grafic de tip parabolă cubică, de ecuație $y=x^{2p+1}$.

2. Comentarii ce se pot desprinde din analiza graficelor de funcții $f(x) = x^n$, $x \in \mathbb{R}$, $n \in \mathbb{N}$.

1°. Pentru orice n , n -natural, graficul trece prin punctul $P(1, 1)$, și pentru orice n , $n \in \mathbb{N}^*$, graficul trece prin originea axelor, deoarece propozițiile „ $0 = 0^n$ ” și „ $1 = 1^n$ ” sînt adevărate.

2°. Dacă n este număr natural par ($n = 2p$, $p > 0$), graficul este simetric în raport cu axa (Oy) , deoarece: $f(-x) = (-x)^{2p} = x^{2p} = f(x)$, funcția f fiind pară. Dacă $M(a, b)$ aparține graficului funcției $f(x) = x^{2p}$, $x \in \mathbb{R}$; atunci este adevărată egalitatea $b = a^{2p}$. De asemenea, este adevărată și egalitatea $b = (-a)^{2p}$; adică, $M'(-a, b)$ aparține graficului funcției date. Fiecărui punct de pe grafic $M(a, b)$ îi corespunde pe același grafic simetricul $M'(-a, b)$ în raport cu axa (Oy) .

3°. Dacă n este impar, $n = 2p + 1$, atunci graficul este simetric în raport cu originea axelor de coordonate $O(0, 0)$. Avem:

$$f(-x) = (-x)^{2p+1} = -x^{2p+1} = -f(x), \text{ funcția fiind impară.}$$

Dacă $M(a, b)$ aparține graficului, atunci este adevărată egalitatea $b = a^{2p+1}$ și, de asemenea, egalitatea $-b = (-a)^{2p+1}$; adică, $M'(-a, -b)$ aparține graficului. Fiecărui punct de pe grafic $M(a, b)$ îi corespunde pe grafic simetricul $M'(-a, -b)$ în raport cu $O(0, 0)$.

4°. Graficul oricărei funcții $f: \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = x^n$, n par, este simetric în raport cu axa (Oy) , iar pentru $n > 0$ are forma similară parabolei funcției $f(x) = x^2$, $x \in \mathbb{R}$ (spre exemplu).

Graficul oricărei funcții $f: \mathbb{R} \rightarrow \mathbb{R}$, $f(x) = x^n$, n impar, este simetric în raport cu $O(0, 0)$ și are forma similară graficului funcției $f(x) = x^3$.

Observație. Pentru a verifica dacă noțiunile recapitulate, ale acestei secvențe, au fost însușite corect, secvența imediat următoare (fie că aceasta este derulată cu ajutorul calculatorului, fie prin metodele clasice obișnuite) ar consta din următorii pași:

i) Desenați, schematic, graficele funcțiilor date prin formulele $y = x^n$ și $y = x^7$, $x \in \mathbb{R}$ și scrieți mulțimile de valori ale argumentului pentru care: a) funcțiile sînt nule; b) funcțiile au valori pozitive; c) funcțiile au valori negative.

ii) Pentru funcția dată prin formula $f(x) = x^{26}$, $x \in \mathbb{R}$, comparați $f(2)$ și $f(-2)$; găsiți $f(-a)$ știind că $f(a) = 32$.

iii) Pentru funcția dată prin formula $f(x) = x^{10}$, $x \in \mathbb{R}$, comparați $f(3)$ și $f(-3)$ și găsiți $f(-a)$ știind că $f(a) = 52$.

IV) Fie punctul arbitrar $M(a, -b)$ aparținînd graficului unei funcții f . Dacă funcția este dată printr-una din formulele $y = x^k$ sau $y = x^h$ să se precizeze dacă punctele $N(-a, b)$ și $P(-a, -b)$ aparțin graficului lui f . Evident, pentru acest ultim punct este necesar un mic comentariu.

Temă. Logaritmi (definiție și proprietăți bazate pe definiție) — lecție de comunicare, rezultatul pentru autoinstruirea automatizată.

1. Noțiunea de logaritm. Să considerăm ecuația exponențială

$$a^x = N, \quad N > 0, \quad a > 0, \quad a \neq 1. \quad (1)$$

Se observă că ecuația (1) nu are soluție pentru $N < 0$, deoarece

$$f: \mathbb{R} \rightarrow (0, \infty), f(x) = a^x$$

are valori pozitive.

Pentru $N > 0$, ecuația (1) are soluție unic determinată, aceasta decurgând din proprietatea de bijectivitate a funcției exponențiale. În figura 12.9 respectiv 12.10 este prezentată rezolvarea grafică a ecuației $a^x = N$, $a > 1$ (respectiv $0 < a < 1$).

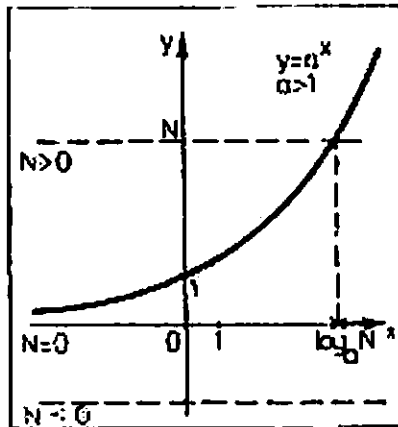


Fig. 12.9. Graficul funcției exponențiale $a^x = N$, $a > 1$.

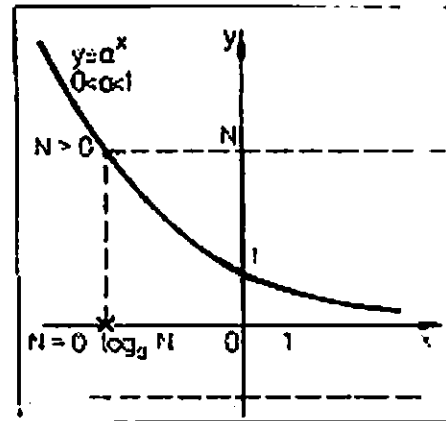


Fig. 12.10. Graficul funcției exponențiale $a^x = N$, $0 < a < 1$.

Soluția ecuației $a^x = N$, unde $a > 0$, $a \neq 1$ și $N > 0$, se numește logaritmul numărului real pozitiv N în baza a .

Prin definiție, logaritmul numărului real pozitiv N în baza a , unde $a > 0$ și $a \neq 1$, este exponentul la care trebuie ridicată baza a pentru a obține N .

Logaritmul numărului N în baza a se notează „ $\log_a N$ ”, se citește „logaritmul în baza a al lui N ” sau „logaritmul lui N în baza a ” și se scrie „ $x = \log_a N$ ”.

Conform definiției avem:

$$\log_2 16 = \log_2 2^4 = 4; \quad \log_3 \frac{1}{243} = \log_3 \left(\frac{1}{3}\right)^5 = \log_3 3^{-5} = -5.$$

Avem deci relația: $\log_a a^n = n$.

Din (1) și (2) rezultă egalitatea: $a^{\log_a N} = N$, $N > 0$.

Dacă în (1) punem $x = 1$, atunci $a^1 = a$ și conform cu (2) avem relația: $\log_a a = 1$.

Dacă baza a este 10 , atunci în loc de „ $\log_{10} N$ ” se scrie „ $\lg N$ ” și se citește „logaritmul lui N în baza 10 ” sau „logaritmul zecimal al lui N ”; iar dacă baza a este numărul irațional $e \approx 2,71828 \dots$, atunci în loc de $\log_a N$ se scrie „ $\ln N$ ” și se citește „logaritmul natural al numărului N ”.

2. Aplicații privind folosirea definiției și a proprietăților bazate pe ea. (Pentru a nu mai prezenta răspunsurile și comentariile corespunzătoare lor, fiecare grupă de exerciții începe cu un exercițiu model, rezolvarea celorlalte fiind analogă modelului).

1°. Dovediți că $\log_5 \frac{1}{125} = -3$.

R: Conform definiției, avem: $5^{-3} = \frac{1}{125}$ (egalitate adevărată)

În același mod, să se arate că egalitățile următoare sînt adevărate:

a) $\log_3 81 = 4$; b) $\log_4 2 = \frac{1}{2}$; c) $\lg \frac{1}{10} = -1$; d) $\log_2 \frac{1}{64} = -6$;

e) $\log_{\sqrt{2}} 8 = 6$; f) $\log_{\frac{1}{3}} 729 = -6$; g) $\log_{31} \frac{1}{3} = -\frac{1}{4}$.

2°. Găsiți numărul N , al cărui logaritm în baza 5 este egal cu -2 .

R: Conform definiției, avem $\log_5 N = -2 \Leftrightarrow N = 5^{-2} \Leftrightarrow N = \frac{1}{25}$.

În același mod, să se găsească N știind că:

a) $\log_5 N = 2$; b) $\log_{\frac{1}{2}} N = -1$; c) $\lg N = -2$; d) $\log N = 3$; e) $\lg N = 0$;

f) $\log_6 N = 1,5$; g) $\log_{\sqrt{7}} N = -\frac{2}{3}$; h) $\log_{64} N = \frac{5}{6}$; k) $\log_{2,25} N = 1,5$.

3°. Găsiți baza a , a logaritmului numărului 64, dacă $\log_a 64 = -4$

R: $\log_a 64 = -4 \Leftrightarrow a^{-4} = 64 \Leftrightarrow 1 = (64)^{-\frac{1}{4}} = (2^6)^{-\frac{1}{4}} = 2^{-\frac{3}{2}} = \frac{1}{\sqrt{2^3}} = \frac{1}{2\sqrt{2}}$.

În același mod, să se găsească baza a știind că:

a) $\log_a 27 = 3$; b) $\log_a 1000 = 1,5$; c) $\log_a 27 = -6$; d) $\log_a 64 = \frac{6}{7}$;

e) $\log_a 4\sqrt{2} = -2,5$; f) $\log_a (27\sqrt[3]{3}) = \frac{5}{6}$; g) $\log_a \frac{1}{9} = -1$;

h) $\log_a \sqrt[3]{3} = \frac{1}{6}$.

4°. Găsiți logaritmul numărului 8 în baza $\sqrt{2}$.

R: $\log_{\sqrt{2}} 8 = x \Leftrightarrow (\sqrt{2})^x = 8 \Leftrightarrow 2^{\frac{x}{2}} = 2^3 \Leftrightarrow \frac{x}{2} = 3 \Leftrightarrow x = 6$. Deci $\log_{\sqrt{2}} 8 = 6$.

În același mod, să se determine numerele:]

a) $\log_3 \frac{1}{9}$; b) $\lg 0,01$; c) $\log_{\frac{1}{8}} 25$; d) $\log_{\frac{1}{\sqrt{3}}} 27$; e) $\log_{0,2} 25$;

f) $\log_4 (8\sqrt{2})$; g) $\log_{27} \frac{\sqrt{3}}{9}$.

5°. Găsiți valoarea expresiei $10^{12} \cdot 10^{12}$.

R: $10^{12} \cdot 10^{12} = 10^{12+12} = 10^{24}$.

În același mod, să se găsească valorile expresiilor:

a) $10^{-12} \cdot 5$; b) $10^{12} \cdot 100$; c) $10^{12} \cdot 0,1$; d) $100^{12} \cdot 7$.

6°. Care din expresii are sens ?

a) $\log_4 0$; b) $\lg 0,01$; c) $\sqrt{\lg 0,9}$; d) $\lg(\lg 9,6)$; e) $\log_3(-7)^2$;

f) $\log_3(-3)^3$; g) $\log_4(-4)^{-4}$; h) $\sqrt{\lg 2,5}$.

7. Rezolvați ecuația $\log_3 x = -1$

R: $\log_3 x = -1 \Leftrightarrow x = 3^{-1} \Leftrightarrow x = \frac{1}{3}$.

În același mod, să se rezolve ecuațiile:

a) $\log_{\frac{1}{2}} x = 3$; b) $\log_{0,3} x = 2$; c) $\log_3(\log_6 x) = 0$.

12.3. Programe utile în procesul de învățămînt

În cele ce urmează se prezintă o serie de programe, care pot fi utilizate în procesul de învățămînt. Ele pot fi completate cu unele din programele date în paragrafele anterioare ale acestui capitol.

Programele de mai jos sînt utile în cadrul orelor de algebră, geometrie, fizică, pentru ilustrarea, prin reprezentări grafice, a unor funcții sau fenomene fizice. De asemenea, ele pot fi folosite pentru verificarea cunoștințelor unor grupuri de elevi, în privința despărțirii cuvintelor în silabe, a cunoașterii regiunilor de județe sau pe alte teme. La sfîrșitul examinării candidații sînt ordonați autonom după mediile obținute.

12.4. Program pentru trasarea cercului trigonometric.

Programul trasează cercul trigonometric și calculează funcțiile sin, cos tg, pentru un unghi dat. Unghiul se dă sub forma unui multiplu k, de 5 grade - (PI/36).

```

1  TITLE
2  REM ***** TRIGONOMETRIC
3  REM **MULTIPLU DE 5 GRADE
4  INPUT K
5  GOTO 10
6  MOVE 50,50
7  FOR J=0 TO 2*PI STEP PI/36
8  DRAW 25*(COS(J)+25*(SIN(J)+50
9  NEXT J
10 MOVE 50,50
11 DRAW -25,0
12 DRAW 25,25
13 DRAW 0,-25
14 MOVE 50,50
15 DRAW 25*(COS(K),25*(SIN(K)
16 DRAW 0,-25*(SIN(K)
17 P=0.188/R1
18 PRINT AT(17,20) "A=1,61"
19 PRINT AT(16,2) "B=0,11"
20 PRINT AT(17,1) "C=-1,01"
21 PRINT AT(16,0) "D=0,-1"
22 PRINT AT(15,1) "U=1,0"
23 PRINT AT(15,2) "V=0,1"
24 PRINT AT(16,1) "SIN(K)=SIN(K)
25 PRINT AT(17,1) "COS(K)=COS(K)
26 PRINT AT(13,1) "TAN(K)=TAN(K)
27 END
    
```

12.5. Program pentru vizualizarea pozițiilor unor drepte care trec prin originea axelor de coordonate.

Programul are un caracter conversațional, solicitînd inițial numărul k de drepte, care urmează a fi reprezentate, coeficienții unghiulari, $M(k)$, ai acestor drepte și limitele pe orizontală ale ferestrei de afișare ($X1$, $X2$).

```

5 INIT P
10 REM"VIZUALIZAREA POZITIEI
20 REM"UNEI DREPTE CE TRECE
30 REM"PRIN O(0,0), IN FUNCTIE
40 REM"DE COEFICIENTUL UNGIULAR-
50 REM"LAR M=TAN(U).UNDE
60 REM"U=α
70 REM"VARIABLE UTILIZATE
80 REM"K = NUMAR DE DREPTE
90 REM"R(K) = VECTORUL ALE
100 REM"CARUI COMPONENTE SINT
110 REM"COEFICIENTII UNGIULARI.
120 REM"LIMITELE FERESTREI
130 REM"AFISATE : I1=LIMITA
140 REM"STINGEA.Y2=LIMITA DREAP
150 REM"TA, Y1 = LIMITA JOS.Y2=
160 REM"LIMITA SUS.
200 PRINT "INTRODUCETI K"
210 INPUT K
220 PRINT "INTRODUCETI COEFICIENTII"
230 DIM R(K)
240 FOR I=1 TO K
240 PRINT "M(I)=?"
250 INPUT R(I)
260 NEXT I
270 PRINT "INTRODUCETI LIMITELE X1,X2"
280 INPUT X1,X2
290 Y1=X1
300 Y2=X2
310 REM"AFISAREA GRAFICULUI
315 INIT P
320 VIEWPORT 40,120,20,100
330 WINDOW X1,X2,Y1,Y2
340 REM"TRASARE AXE SI CONTUR
350 MOVE X1,Y1
360 DRAW X2,Y1
370 DRAW X2,Y2
380 DRAW X1,Y2
390 DRAW X1,Y1
400 MOVE X1,0
410 DRAW X2,0
420 MOVE 0,Y1
430 DRAW 0,Y2
440 PRINT AT (1,20) "Y"
450 PRINT AT(15,30) "X"
500 R=(X2-X1)/20
510 FOR I=1 TO Y
515 PRINT AT(2+I,11) "M":I;"=":R:I
520 R=R+R
530 MOVE (1,R(1)+K1
540 DRAW (2,R(1)+K2
550 MOVE R1,0
555 U=ATN(R(1))
556 IF U=0 THEN 600
557 IF U<0 THEN 560
558 U=PI+U
560 FOR J=0 TO U STEP U/10
570 DRAW R1=COS(J),R1=SIN(J)
580 NEXT J
600 NEXT I
END

```

12.6. Graficul funcției de gradul doi.

Programul afișează într-o fereastră dată ($X1$, $X2$) o familie de parabole de gradul doi ($Y=A*X^2$) cu diferiți coeficienți. Limitele verticale ale ferestrei se calculează automat.

```

5 INIT P
10 REM*GRAFICUL FUNCTIEI
20 REM*DE GRADUL DOI
30 REM*Y=AXK^2
40 REM*VARIABLE UTILIZATE
50 REM*K=NUMAR DE PARABOLE
60 REM*A(K)=VECTORUL ALE
70 REM*CARUI ELEMENTE SINT
80 REM*COEFICIENTII LUI X^2
90 REM*LIMITELE FERESTEI
100 REM*AFISATE I X1=LIMITA
110 REM*STINGA, X2=LIMITA
120 REM*DREAPTA, Y1=LIMITA JOS
130 REM*Y2=LIMITA SUS
140 REM*FEREASTRA SE CONSIDERA
150 REM*BIMETRICA, DEC);
160 REM*ABS(X1)=ABS(X2)=X 91
170 REM*ABS(Y1)=ABS(Y2)=Y^2
200 PRINT "INTRODUCETI K:"
210 INPUT K
220 PRINT "INTRODUCETI COEFICIENTII"
230 DIM A(K)
240 FOR I=1 TO K
250 PRINT "A(";I;")=";
260 INPUT A(I)
270 NEXT I
280 PRINT "INTRODUCETI X:"
290 INPUT X
300 X1=-X
310 X2=X
320 Y1=-X^2
330 Y2=X^2
340 REM*AFISARE GRAFICA
350 INIT P
360 VIEWPORT 40,120,20,100
370 WINDOW X1,X2,Y1,Y2
380 REM*TRABARE AXE SI CONTIUR
400 MOVE X1,Y1
410 DRAW X2,Y1
420 DRAW X2,Y2
430 DRAW X1,Y2
440 DRAW X1,Y1
450 MOVE X1,0
460 DRAW X2,0
470 MOVE 0,Y1
480 DRAW 0,Y2
540 PRINT AT(1,20) "Y"
550 PRINT AT(15,30) "X"
560 PRINT AT(1,11) "Y=A*X^2"
600 P=X/20
610 FOR J=1 TO K
615 PRINT AT(2+J,11) "A(";J;")=";A(J)
620 MOVE X1,A(J)=-X1^2
630 FOR J=X1 TO X2 STEP P
640 DRAW J,A(J)=J^2
650 NEXT J
660 NEXT J
700 END

```

12.7. Graficul funcției de gradul N

Programul reprezintă graficele unei familii de parabole ($Y = A(k) \cdot X^N$), unde $A(k)$ este coeficientul parabolei cu numărul k , iar $N \geq 0$. Programul solicită numărul k de parabole, coeficienții $A(k)$ și limitele orizontale $X1$, $X2$ ale ferestrei de afișare.

```

5 INIT P
10 REM DRAPICUL FUNCȚIEI
20 REM DE GRADUL N
30 KLN Y=A+I*N
40 REM VARIABILE UTILIZATE
50 REM I=NUMAR DE PARABOLE
60 REM A(K)=VECTORUL ALE
70 REM C=CAPII ELEMENTE SINT
80 REM COEFICIENTII LUI X^N
90 REM LIMITELE FERESTREI
100 REM AFISATE: X1=LIMITA
110 REM STINGA, X2=LIMITA
120 REM DREAPTA, Y1=LIMITA JOS
130 REM Y2=LIMITA SUS
140 REM FEREAȘTRA SE CONSIDERA
150 REM SIMETRICA, DEC)
160 REM ABS(X)=ABS(X2)+I, S)
170 REM ADS(X)=ABS(X2)+I*2
200 PRINT "INTRODUCETI K:"
210 INPUT K
220 PRINT "INTRODUCETI COEFICIENTII"
230 DIM A(K)
235 DIM N(K)
240 FOR I=1 TO K
250 PRINT "A(I):"
260 INPUT A(I)
265 PRINT "N(I):"
270 INPUT N(I)
275 NEXT I
280 PRINT "INTRODUCETI X:"
290 INPUT X
300 X1=X
310 X2=X
320 Y1=X^2
330 Y2=X^2
340 REM AFISARE GRAFICA
350 INIT P
360 VIEWPORT 40,120,20,100
370 WINDOW X1,X2,Y1,Y2
380 REM AXE SI CENTR)
400 MOVE X1,Y1
410 DRAW X2,Y1
420 DRAW X2,Y2
430 DRAW X1,Y2
440 DRAW X1,Y1
450 MOVE 0,0
460 DRAW X2,0
470 MOVE 0,0
480 DRAW 0,0
490 PRINT AT(1,20) "Y"
500 PRINT AT(15,30) "X"
510 PRINT AT(1,1) "Y0=X^2"
520 P=X/20
530 FOR I=1 TO P
540 PRINT AT(24,1) "A(I):"
550 MOVE X1,A(I)+I*N(I)
560 FOR J=X1 TO X2 STEP P
570 DRAW (A(I)+I*N(I))
580 NEXT J
590 NEXT I
600 PRINT "DORITI ALT F7, SA-J"
610 INPUT D
620 IF D=1 THEN 330
630 END

```

12.8. Graficul funcției logaritmice.

Programul afișează graficul funcției logaritmice în bazele 2, 3, 5 și 10. Inițial se solicită limitele orizontale de afișare.

```

10 INIT F
11 REM "GRAFICUL FUNCTIEI
12 REM "LOGARITMICE IN
13 REM "DABELE 1,2,3,5,11
14 REM "LIMITELE INTERVALULUI
15 REM "DE AFISARE
16 REM "X1 = LIMITA STINGEA
17 REM "X2 = LIMITA DREAPTA
18 REM "Y1 = LIM. INF. Y2 = LIM. SUP.
19 REM "LIMITA SI VA FI CALCULATA
20 REM "PENTRU Y = LOG(X1)/LOG(10)
21 REM "DACA Y > 1 LIMITA X2 SE
22 VA CALCULA
23 REM "PENTRU Y = LOG(X2)/LOG(10)
24 PRINT "DENUMERATI LIMITILE F
25 PRINT "X1 SI X2"
26 PRINT "Y1 SI Y2"
27 INPUT "X1 = "
28 INPUT "X2 = "
29 PRINT "Y2 = "
30 IF X1 < 0 THEN GOTO 34
31 Y1 = LOG(X1)/LOG(10)
32 IF X2 < 1 THEN GOTO 34
33 Y2 = LOG(X2)/LOG(10)
34 GOTO 36
35 Y2 = LOG(X2)/LOG(10)
36 REM "AFISARE GRAFICUL
37 INIT F
38 VIEWPORT 40,120,0,70
39 WINDOW 1:XD,Y1,Y2
40 REM "TRAGARE ALE SI CURBIL
41 MOVE 41,Y1
42 DRAW 42,Y2
43 DRAW 41,Y2
44 DRAW 41,Y1
45 MOVE 41,0
46 DRAW 42,0
47 MOVE 0,Y2
48 DRAW 0,Y2
49 PRINT AT(10,10) "X1="
50 PRINT AT(10,20) "X2="
51 PRINT AT(10,30) "Y1="
52 PRINT AT(10,40) "Y2="
53 PRINT AT(10,50) "VALORILE LOG(10)"
54 PRINT AT(10,60) "X1="
55 PRINT AT(10,70) "X2="
56 PRINT AT(10,80) "Y1="
57 PRINT AT(10,90) "Y2="
58 F = 1
59 GOSUB 600
60 K = 10
61 GOSUB 600
62 PRINT "DORITI ALTE LIMITI > (A=1)"
63 INPUT D
64 IF D=1 THEN GOTO 59
65 GOTO 700
66 PRINT "LIMITA SI INCORRECTA"
67 STOP
68 F=(K2-K1)/100
69 FOR J = X1 TO X2 STEP F
70 DRAW J:LOG(J)/LOG(10)
71 NEXT J
72 MOVE X1,Y1
73 RETURN
74 END

```

12.9. Program pentru studiul aruncării corpurilor sub un unghi dat

Programul solicită unghiul sub care are loc aruncarea, viteza inițială și viteza vântului.

Pentru încadrarea graficului în limitele ecranului, programul execută o scalare pe verticală, calculând inițial înălțimea maximă la care ajunge corpul. În acest scop solicită numărul de pași pentru a calcula valorile parabolei în punctele respective, reținând valoarea maximă.

Se afișează înălțimea maximă și bătaia.


```

1 INIT I
2 REM PROGRAM PENTRU STUDIUL APOA-
  CARI COEFICIENTI
3 PRINT "INTRODUCETI UNghiUL U,V,INITI=U,V,VINTE
  M"
4 INPUT U,V,W
5 PRINT "INTRODUCETI NUMARUL DE PASI N"
7 INPUT N
30 INIT P
31 L=P1+U/180
32 I=V*(COS(L)-W
33 B=V*SIN(L)/W
34 A=-4.905/N^2
35 D=K+V*SIN(L)*2/9.81
36 R2=(1/19.62)*V*SIN(L)*11^2
37 PRINT AT(20,11)*N,MAX = "162
38 PRINT AT(20,11)*N,MAX = "172
45 GOSUB 1000
50 STOP
60 END
1000 IF X1 < X2 THEN 1020
1010 PRINT "LIMITE ORESIVE"
1020 STOP
1030 M=(X2-X1)/N
1040 VIEWPORT 40,100,20,100
1050 X1=0
1070 R1=0
1100 IF X2=R2 THEN 1150
1110 X2=R2
1120 GOTO 1150
1140 R2=X2
1190 WINDOW X1,X2,R1,R2
1200 MOVE X1,0
1201 DRAW X2,0
1202 MOVE 0,R1
1203 DRAW 0,R2
1204 Y=11
1220 GOSUB 1300
1230 MOVE 1,F
1240 FOR I=1 TO M
1250 X=X+M
1260 GOSUB 1300
1270 DRAW 1,F
1280 NEXT I
1290 RETURN
1300 F=A*X^2+B*X
1310 RETURN

```

12.10. Calculul punctului de intersecție a două drepte

O dreaptă poate fi definită prin specificarea coordonatelor unui punct $M(x, y)$ și a unghiului pe care îl face cu direcția pozitivă a axei Ox . Fie două drepte $D1$ și $D2$ de ecuații parametrice:

$$\begin{aligned}
 D1: \quad & X = X_1 + L_1 \cos U_1 \\
 & Y = Y_1 + L_1 \sin U_1 \\
 D2: \quad & X = X_2 + L_2 \cos U_2 \\
 & Y = Y_2 + L_2 \sin U_2
 \end{aligned}$$

Fie (X_0, Y_0) coordonatele punctului de intersecție. Înlocuind în ecuațiile parametrice, se obține sistemul:

$$\begin{aligned}
 X_0 &= X_1 + L_1 \cos U_1 \\
 Y_0 &= Y_1 + L_1 \sin U_1
 \end{aligned}$$

$$X_0 = X_2 + L_2 \cos U_2$$

$$Y_0 = Y_2 + L_2 \sin U_2$$

De aici rezultă :

$$L_1 \cos U_1 - L_2 \cos U_2 = X_2 - X_1$$

$$L_1 \sin U_1 - L_2 \sin U_2 = Y_2 - Y_1$$

De unde : $L_1 = ((X_1 - X_2) \sin U_2 - (Y_1 - Y_2) \cos U_2) / \sin (U_1 - U_2)$.

Având determinat L_1 , coordonatele punctului de intersecție vor fi :

$$X_0 = X_1 + L_1 \cos U_1$$

$$Y_0 = Y_1 + L_1 \sin U_1$$

```

1  REM "#####"
2  REM "A CALCULUL PUNCTULUI DE "
3  REM "A INTERSECȚIE A DOUA DREPTI "
4  REM "#####"
5  DIM B(1)
6  REM "CITIREA DATELOR"
7  PRINT "INTRODUCETI PARABTRII DREPTELOR"
10 PRINT "X1,Y1,U1(GRADE):"
15 INPUT X1,Y1,U1
20 PRINT "X2,Y2,U2(GRADE):"
25 INPUT X2,Y2,U2
27 REM "TESTARE PARALLELISM DREPTI"
30 N=ABS(U1-U2)/180
35 IF INT(N)<>N THEN 50
40 PRINT "DREPTILE SINT PARALELE"
45 STOP
47 REM "CALCULUL COORDONATELOR"
48 REM "PUNCTULUI DE INTERSECȚIE"
50 Q1=U1-P1/180
55 U2=Q2-P1/180
60 B1=(X1-X2)*SIN(U2)-(Y1-Y2)*COS(U2)
65 B=SIN(Q1-U2)
70 L=B1/B
75 X0=X1+L*COS(U1)
80 Y0=Y1+L*SIN(U1)
85 PRINT "COORDONATELE PUNCTULUI"
86 PRINT "DE INTERSECȚIE SINT:"
90 PRINT "X0";X0
95 PRINT "Y0";Y0
100 PRINT "DORITI AFISARE GRAFICA? (DA,NU):"
105 INPUT B(1)
110 IF B(1)=-1 THEN 120
115 STOP
118 REM "DETERMINAREA SPATIULUI DE AFISAT"
120 T1=ABS(X0)
125 T2=ABS(Y0)
130 IF T1>T2 THEN 140
135 T1=T2
140 T1=2*T1
145 IF T1<>0 THEN 149
146 T1=30
149 J=1
150 VIEWPORT 10,90,10,90
155 WINDOW -T1,T1,-T1,T1
157 REM "TRASAREA AXELOR"
160 MOVE -T1,0
165 DRAW T1,0
170 MOVE 0,-T1
175 DRAW 0,T1
178 REM "TRASAREA DREPTELOR"
180 MOVE X1-2*T1+COS(U1),Y1-2*T1+SIN(U1)
185 DRAW X1+2*T1+COS(U1),Y1+2*T1+SIN(U1)
190 MOVE X2-2*T2+COS(U2),Y2-2*T2+SIN(U2)
195 DRAW X2+2*T2+COS(U2),Y2+2*T2+SIN(U2)
200 MOVE X0,T0
205 DRAW X0,0
210 MOVE X0,T0
215 DRAW 0,T0
220 END

```

Dacă dreptele sînt paralele numitorul expresiei pentru calculul lui L , este nul. Deci se va testa condiția de neparalelism $U_1 - U_2 \neq K\pi$.

Programul funcționează astfel :

1. Citește parametrii celor două drepte ; X_1, Y_1, U_1 și X_2, Y_2, U_2 .
2. Testează dacă cele două drepte sînt paralele. Dacă dreptele sînt paralele se oprește execuția programului.
3. Calculează coordonatele punctului de intersecție și apoi le afișează.
4. Afișează grafic (eventual) rezultatul.

La definirea spațiului de afișaj în linia 155, s-a avut în vedere ca originea și axele sistemului de coordonate să apară pe ecran.

12.11. Calculul punctelor de intersecție a două cercuri

Se consideră cercurile definite prin centru și rază : $C_1 (X_1, Y_1, R_1)$ și $C_2 (X_2, Y_2, R_2)$. Distanța dintre centrele celor două cercuri este :

$$D = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

Se va nota $G = X_2 - X_1$ și $H = Y_2 - Y_1$.

Pentru a se intersecta, distanța dintre cercuri trebuie să fie mai mică decît suma razelor și mai mare decît diferența lor : $|R_2 - R_1| \leq D \leq R_1 + R_2$.

În cazul egalității, cercurile vor fi tangente. Se notează cu U_1 unghiul format de dreapta ce unește centrele cercurilor cu orizontala și U_2 unghiul format de aceeași dreaptă cu raza primului cerc dusă în punctul de intersecție dorit. Coordonatele punctului de intersecție vor fi :

$$X = X_1 + R_1 \cos (U_1 + F \cdot U_2)$$

$$Y = Y_1 + R_1 \sin (U_1 + F \cdot U_2)$$

Unde : $F = 1$ pentru punctul din stînga dreptei ce unește centrele (privit din C_1) și -1 pentru cel din dreapta.

Pentru calculul unghiurilor U_1 și U_2 folosim relațiile :

$$U_1 = \text{ATAN} (G/H), \text{ sau } \pi/2 \text{ dacă } H = 0$$

și

$$U_2 = \arccos (R_1^2 + D^2 - R_2^2) / (2 \cdot R_1 \cdot D)$$

însă \arccos va fi calculat prin funcția ATAN .

Programul funcționează în felul următor :

1. Citește coordonatele centrului și raza X_1, Y_1, R_1 pentru primul cerc.
2. Citește coordonatele centrului și raza X_2, Y_2, R_2 pentru al doilea cerc.

3. Citește valoarea lui F (1 = punctul din stînga, -1 = cel din dreapta)
4. Calculează valorile G, H și D. Dacă nu e verificată condiția : $|R2-R1| \leq D \leq R1+R2$ execuția se oprește.
5. Calculează unghiurile U1 și U2.
6. Calculează coordonatele X, Y ale punctului de intersecție dorit.
7. Măsură grafică (eventuală) a rezultatului.
8. Stop

```

001 REM "....."
002 REM "CALCULUL PUNCTELOR DE INTERSECȚIE A DOUA CERCI"
003 REM "....."
004
005 DIM R(1)
006
007 PRINT "INTRODUCETI COORDONATELE CENTRILOR SI RAZA"
008 PRINT "PENTRU PRIMA CERCE (X1,Y1,R1)"
009
010 INPUT X1,Y1,R1
011 IF R1=0 THEN 070
012
013 PRINT "PENTRU AL DOILUA CERCE (X2,Y2,R2)"
014
015 INPUT X2,Y2,R2
016 IF R2=0 THEN 070
017 PRINT "DAZA NEGATIVA SAU NUL?"
018 GO TO 070
019
020 REM "CALCULUL MARIMILOR G,H,D"
021
022 G=X2-X1
023 H=Y2-Y1
024 D=SQR(G*G+H*H)
025
026 REM "VERIFICARE CONDIȚIILOR DE INTERSECȚIE"
027
028 IF D=0 THEN 145
029 IF D<R2 THEN 155
030 PRINT "CERCII IDENTICE"
031 GO TO 070
032 IF D>R1+R2 THEN 155
033 IF D<ABS(R1-R2) THEN 160
034 PRINT "CERCII NU SE INTERSECȚEAZĂ"
035 GO TO 070
036
037 REM "CALCULUL UNGIULUI U1"
038
039 U1=PI/2
040 IF G<0 THEN 205
041 IF H=0 THEN 225
042 U1=0
043 GO TO 235
044 U1=ATN(H/G)
045 IF G=0 THEN 225
046 U1=PI
047
048 REM "CALCULUL UNGIULUI U2"
049
050 L1=(R1+R1+D+D-R2)/2*(2*R1+D)
051 U2=PI/2
052 IF L1=0 THEN 275
053 L2=SQR(1-L1*L1)
054 U2=ATN(L2)
055 IF L1<0 THEN 275
056 U2=2*PI
057
058 PRINT "PUNCTELE DE INTERSECȚIE AU COORDONATELE"
059
060 P=1

```

```

290 X=X1+R1*COS(II+P/NU2)
295 Y=Y1+R1*SIN(II+P/NU2)
300 PRINT "X=";X,"Y=";Y
305 F=F+P
310 IF F<N THEN 290
315
320 PRINT "DORITI AFISARE GRAFICA(D/N)?";
325
330 INPUT R2
335 IF R2<"D" THEN 590
340
345 REM "STABILIREA CONDIȚIILOR DE AFISARE"
350
355 INIT
360 VIEWPORT 40,140,0,100
365 AX1
370 B=Y2
375 GO SUB 545
380 T1=T
385 A=Y1
390 B=Y2
395 GO SUB 545
400 IF T>=T1 THEN 410
405 T=T1
410 WINDOW -1,T,-T,T
415 MOVE -1,-T
420 DRAW T,-T
425 DRAW T,T
430 DRAW -T,T
435 DRAW -T,-T
440
445 REM "TRASAREA CERCURILOR"
450
455 J=2*PI/P1/NU1
460 K=PI/10
465 FOR I=0 TO J STEP I
470   DRAW X1+R1*COS(I),Y1-R1*SIN(I)
475 NEXT I
480 MOVE X2+R2,Y2
485 FOR I=0 TO J STEP K
490   DRAW X2+R2*COS(I),Y2+R2*SIN(I)
495 NEXT I
500
505 REM "TRASAREA AXELOR"
510
515 MOVE -T,0
520 DRAW T,0
525 MOVE 0,-T
530 DRAW 0,T
535 MOVE -T,-T
540 GO TO 590
545
550 REM "SUBROUTINA DE CALCUL A COORDONATEI NASTINE"
555 REM "PE O DIRECTIE DATA"
560
565 T=ABS(A-R1)
570 IF T=ABS(B-R2) THEN 580
575 T=ABS(B-R2)
580 IF T=ABS(A-R1) THEN 570
585
590 T=ABS(A+R1)
595 IF T=ABS(B+R2) THEN 580
600 T=ABS(B+R2)
605 RETURN
610
615 PRINT "DORITI ALTE CERCI(D/N)?";
620
625 INPUT R3
630 IF R3="I" THEN 620
635 END

```

12.12. Calculul tangențelor dintr-un punct la un cerc

Fie o dreaptă D de ecuații parametrice :

$$X = X_1 + I \cdot \cos T$$

$$Y = Y_1 + I \cdot \sin T$$

unde: X_1, Y_1 sînt coordonatele unui punct precizat al dreptei, T direcția dreptei, iar L este distanța de la punctul (X_1, Y_1) la punctul curent.

Fie un cerc C , de centru (X_0, Y_0) și rază R , [de ecuație:]

$$(X - X_0)^2 + (Y - Y_0)^2 = R^2$$

Pentru a se obține tangentele din punctul (X_1, Y_1) la cercul C , se vor înlocui X și Y dați de ecuația dreptei D în ecuația cercului. Se obține:

$$1) \quad L^2 + 2L(H \cos T + V \sin T) + V^2 + H^2 - R^2 = 0$$

unde s-a notat: $H = X_1 - X_0$ și $V = Y_1 - Y_0$.

Se presupune că punctul (X_1, Y_1) este exterior cercului C , deci: $\sqrt{H^2 + V^2} > R$. Condiția ca dreapta D să fie tangentă la cerc este ca ecuația (1) de gradul doi în L , să aibă soluție unică. Pentru aceasta trebuie ca:

$$(2) \quad (H \cos T + V \sin T)^2 = V^2 + H^2 - R^2$$

În rezolvarea acestei ecuații se disting două cazuri:

a) $H \neq 0$. Din (2) se obține:

$$\cos T + \frac{V}{H} \sin T = \pm \frac{\sqrt{V^2 + H^2 - R^2}}{H}$$

Fie $F = \arctg\left(\frac{V}{H}\right)$. Înlocuind mai sus se obține:

$$\cos(T - F) = \pm \sqrt{\frac{V^2 + H^2 - R^2}{V^2 + H^2}}$$

$$\tan(T - F) = \pm \frac{R}{\sqrt{V^2 + H^2 - R^2}}$$

și deci $T_{1,2} = \arctg \frac{V}{H} + \arctg \frac{\pm R}{\sqrt{V^2 + H^2 - R^2}}$.

b) $V \neq 0$. Din ecuația (2), împărțind cu V rezultă:

$$\sin(T + F) = \pm \sqrt{\frac{V^2 + H^2 - R^2}{H^2 + V^2}}$$

unde F este unghiul pentru care $\tan F = \frac{H}{V}$.

Procediul ca la cazul precedent, se obține:

$$T_{1,2} = \arctg \frac{H}{V} + \arctg \left(\pm \frac{\sqrt{V^2 + H^2 - R^2}}{R} \right).$$

Se observă că în ambele cazuri se obțin două unghiuri T , corespunzătoare celor două tangente la cerc. Avînd direcțiile tangențelor determinate printr-una din cele două metode, soluția ecuației (1) corespunzătoare unghiurilor

T_1 și T_2 va fi: $L_{1,2} = -(H \cos T_{1,2} + V \sin T_{1,2})$. Coordonatele punctelor de tangență se vor obține înlocuind valorile obținute pentru T și L în ecuația dreptei D. S-au prezentat cele două cazuri de rezolvare a ecuației (2), deoarece este posibil ca punctul (X_1, Y_1) să se afle pe aceeași verticală sau orizontală cu centrul cercului.

În aceste condiții va rezulta $H=0$ sau $V=0$. (Evident, H și V nu pot fi simultan nuli). Programul va testa dacă $H \neq 0$. În acest caz, direcțiile tangențelor vor fi determinate prin prima metodă; altfel se va alege cealaltă soluție.

Funcționarea programului:

1. Citirea datelor

a) centrul și raza cercului (X_0, Y_0, R)

b) punctul din care se duc tangentele (X_1, Y_1)

2. Testează dacă punctul dat este în interiorul sau pe conturul cercului. În caz afirmativ se va afișa un mesaj corespunzător și se va opri execuția.

3. Se determină direcțiile celor două tangente și coordonatele punctelor de tangență și se afișează rezultatele.

1. Afășează (după dorință) grafic soluția găsită.

```

1 REM "*****"
2 REM " CALCULUL DREPTELOR DE TANGENTA "
3 REM " DINTR-UN PUNCT DAT LA UN CERC DAT "
4 REM "*****"
5 REM "CITIREA DATELOR"
6 DIM D(4)
7 PRINT "INTRODUCETI COORDONATELE CENTRULUI"
10 PRINT "SI RAZA CERCULUI (X0,Y0,R)"
15 INPUT X0,Y0,R
20 PRINT "INTRODUCETI COORDONATELE PUNCTULUI (X1,Y1)"
25 INPUT X1,Y1
30 LET H=X1-X0
35 LET V=Y1-Y0
40 LET D=SOR((V*V+H*H))
45 REM "SE TESTEAZA DACA PUNCTUL (X1,Y1)"
47 REM "ESTE IN INTERIORUL CERCULUI"
48 IF D>R THEN 75
50 IF D=R THEN 65
55 PRINT "PUNCTUL ESTE INTERIOR CERCULUI"
60 STOP
65 PRINT "PUNCTUL ESTE PE CERC"
70 STOP
75 REM "PUNCT EXTERIOR CERCULUI"
80 REM "CALCULUL PUNCTELOR DE TANGENTA"
90 LET R1=SOR((V*V+H*H)-R*R)
95 IF H=0 THEN 100
105 LET T1=ATN(V/H)+ATN(R1/R)
110 LET T2=ATN(V/H)-ATN(R1/R)
115 LET L1=-H*COS(T1)+V*SIN(T1)
120 LET L2=-H*COS(T2)+V*SIN(T2)
125 LET X1=X0+L1*COS(T1)
130 LET X2=X0+L2*COS(T2)
135 LET Y1=Y0+L1*SIN(T1)
140 LET Y2=Y0+L2*SIN(T2)
145 PRINT "COORDONATELE PUNCTELOR DE TANGENTA SINT:"
150 PRINT "X1=";X1;"Y1=";Y1
155 PRINT "X2=";X2;"Y2=";Y2
160 PRINT "DORITI AFISARE GRAFICA ? (DA,NU)"
165 INPUT D(1)
170 IF D(1)="D" THEN 200
175 REM "DEFINIREA SPATIULUI UTILIZATOR"
180 REM "SI A PORTIUNII DE ECRAI PENTRU AFISARE"
185 LET L1=ABS(X0)
190 IF L1>ABS(Y0) THEN 195
195 LET L1=ABS(Y0)

```

12.13. Calculul cu polinoame

Se va studia împărțirea unui polinom: $a_1x^n + a_2x^{n-1} + \dots + a_{n+1}$ printr-un polinom ireductibil și unitar peste R , anume cazul împărțirii prin $x+a$ și x^2+px+q . Coeficienții polinomului cit: $b_1x^n + b_2x^{n-1} + \dots + b_{n+1}$, vor fi determinați prin identificare în relația:

$$(b_1x^n + b_2x^{n-1} + \dots + b_{n+1})(x+a) + b_{n+2} = a_1x^n + a_2x^{n-1} + \dots + a_{n+1}$$

unde cu b_{n+2} s-a notat restul. Rezultă:

$$b_1 = 0 \text{ și } b_{i+1} = a_i - ab_i, \text{ pentru } i = 1, 2, \dots, n+1.$$

Programul de calcul al coeficienților citului și restului este dat mai jos.

Similar, se obține și programul ce determină citul și restul împărțirii polinomului $a_1x^n + a_2x^{n-1} + \dots + a_{n+1}$ prin polinomul x^2+px+q obținându-se citul și restul $rx+s$.

Programele de împărțire se pot utiliza în descompunerea în factori a polinoamelor în găsirea rădăcinilor unui polinom, în calculul valorii unui polinom.

```

5 REM *****
10 REM "IMPARTIREA UNUI POLINOM DE "
20 REM "GRAD N PRINTR-UN POLINOM "
30 REM "DE GRADUL 1 : x+a "
40 REM "*****
60 PRINT "N=";
70 INPUT N
80 DIM A(N+1),B(N+2)
90 FOR I = 1 TO N+1
100 PRINT "A(");I;")=";
110 INPUT A(I)
120 NEXT I
130 PRINT "A=";
140 INPUT A
150 B(1)=0
160 FOR I = 2 TO N+2
170 B(I)=A(I-1)-A*B(I-1)
180 NEXT I
190 REM "AFISAREA REZULTATULI"
200 REM
210 FOR J=2 TO N+1
220 PRINT "B(");J;")=";B(J)
230 NEXT J
240 PRINT "R=";B(N+2)
250 END
    
```

```

5 REM "*****
10 REM " IMPARTIREA UNUI POLINOM DE "
15 REM " GRAD N PRINTR-UN POLINOM "
20 REM " DE GRADUL 001 "
25 REM "*****
30 PRINT "N=";
40 INPUT N
45 DIM A(N+1),B(N+2)
50 PRINT "COEFICIENTII POLINOMULUI "
60 FOR I=1 TO N+1
70 PRINT "A(");I;")=";
80 INPUT A(I)
90 NEXT I
100 PRINT "COEF. DIVIZORULUI:"
110 PRINT "X^2+P*X+Q"
120 PRINT "P=";
130 INPUT P
140 PRINT "Q=";
150 INPUT Q
200 B(1)=0
210 D(2)=0
220 FOR J=3 TO N+2
230 B(J)=A(J-2)-P*B(J-1)-Q*B(J-2)
240 NEXT J
250 REM "AFISAREA REZULTATELOR"
260 FOR I=3 TO N+1
270 PRINT "B(");I;")=";B(I)
280 NEXT I
290 PRINT "AFISARE REST: RX+S "
300 PRINT "R=";B(N+2)
310 PRINT "S=";B(N+3)+P*B(N+2)
320 END
    
```

12.14. Rezolvarea ecuațiilor algebrice prin metoda Bairstow

Ecuațiile algebrice, $f(x)=0$, unde f este un polinom pot fi rezolvate prin metode aproximative. În continuare se va descrie o metodă care constă în descompunerea polinomului în produs de polinoame de gradul doi (metoda

Hirstow). Fie polinomul $f(x) = a_1 x^n + a_2 x^{n-1} + \dots + a_{n+1}$ și (p, q) două numere. Împărțind polinomul prin $x^2 + px + q$ obținem:

$$\sum_{l=1}^{n+1} a_l x^{n+1-l} = \left(\sum_{l=3}^{n+1} b_l x^{n+1-l} \right) (x^2 + px + q) + rx + s$$

Prin identificare se obțin coeficienții b_l , r și s în funcție de p și q : $b_1 = b_2 = 0$ și

$$(1) \quad b_i = a_{i-2} - pb_{i-1} - qb_{i-2}, \quad \text{pentru } 3 \leq i \leq n+3$$

$$(2) \quad \text{iar, } r = b_{n+3} \text{ și } s = b_{n+3} + pb_{n+2}$$

Va trebui să se găsească p și q astfel încît: $r(p, q) = s(p, q) = 0$. Acest lucru revine la rezolvarea sistemului:

$$r + \Delta p \frac{\partial r}{\partial p} + \Delta q \frac{\partial r}{\partial q} = 0$$

$$s + \Delta p \frac{\partial s}{\partial p} + \Delta q \frac{\partial s}{\partial q} = 0$$

Pentru calculul lui $\frac{\partial r}{\partial p}$, se vor calcula $\frac{\partial b_l}{\partial p}$ prin recurență din expresiile obținute prin identificare (1).

$$\frac{\partial b_i}{\partial p} = -b_{i-1} - p \frac{\partial b_{i-1}}{\partial p} - q \frac{\partial b_{i-2}}{\partial p}$$

Notînd $c_i = \frac{\partial b_i}{\partial p}$ pentru $1 \leq i \leq n+3$, șirul c_i va fi: $c_1 = c_2 = 0$ și $c_i = -b_{i-1} - pc_{i-1} - qc_{i-2}$, pentru $3 \leq i \leq n+3$.

Din formulele (2) se obține:

$$\frac{\partial r}{\partial p} = c_{n+3} \text{ și } \frac{\partial s}{\partial p} = -q c_{n+2}$$

Pe de altă parte, derivînd formulele (1) se obține:

$$\frac{\partial b_i}{\partial q} = c_{i-1}, \text{ deci } \frac{\partial r}{\partial q} = c_{n+1} \text{ și } \frac{\partial s}{\partial q} = c_{n+2} + pc_{n+1}$$

Sistemul (3) se va rezolva prin metoda Cramer:

$$d = c_{n+2}^2 + c_{n+1}(c_{n+2} + b_{n+2})$$

$$\Delta p = \frac{c_{n+1}b_{n+2} - b_{n+2}c_{n+2}}{d} \quad \text{și} \quad \Delta q = \frac{-b_{n+1}(qc_{n+1} + pc_{n+2}) - b_{n+2}c_{n+2}}{d}$$

Se definește apoi un șir de puncte (p', q') care, în principiu, converge către un punct (p, q) încât: x^2+px+q divide pe $f(x)$. Practic, numărul de puncte (iterații) este limitat la $k=100$, în program, utilizându-se ca condiție de oprire:

$$\frac{|\Delta p| + |\Delta q|}{|p| + |q|} < \varepsilon$$

(ε fiind luat 10^{-3} în program).

Ecuația $x^2+px+q=0$ este apoi rezolvată în \mathbb{C} folosind formulele uzuale.

Programul continuă cu polinomul cît $\sum_{i=0}^{n+1} b_i x^{n+1-i}$, dacă gradul său este mai mare decît 2.

```

40 REM *****
45 REM " REZOLVARBA ECUATIILOR ALGE- "
70 REM " BRICE-METODA BAIRSTOW- "
75 REM *****
80 PRINT "GRADUL N=";
90 INPUT N
95 DIM A(N+1),B(N+1),C(N+1)
100 FOR I=1 TO N+1
110 PRINT "A(I);";
120 INPUT A(I)
130 NEXT I
150 P=0
155 Q=0
160 X=100
165 H=.001
170 REM " TEST ASUPRA GRADULUI
180 REM
210 IF N <= 2 THEN GOTO 570
220 REM "CAUTARE P SI Q
230 REM
240 REM "INITIALIZARE CONTOR CALCULI
260 J=0
280 IF J>K THEN GOTO 590
290 J=J+1
300 REM "CALCULUL COEFICIENTILOR
310 REM "B(I) SI C(I)"
320 B(1)=0
325 B(2)=0
328 C(1)=0
325 C(2)=0
330 FOR I=3 TO N+1
340 B(I)=A(I)*(-21-P*B(I-1))-B(I*(I-2)
350 C(I)=A(I)*(-11-P*C(I-1))-B(I*(I-2)
360 NEXT I
370 REM "CALCUL DP SI DD"
380 REM
390 X=B(N+1)
392 Y=D(N+1)
394 Z=C(N+1)
396 T=C(N+1)
398 U=C(N+1)
400 D=7*Z-2*(U*X)
410 IF D=0 THEN GOTO 590
420 A=12*(Y-X*T)/D
430 B(1)=Y*(D+Z*A)-Y*T/D
440 REM "NOILE P SI Q"
450 P=P-A
455 Q=Q-B
460 F=(ABS(A)+ABS(B))/ABS(P)+ABS(Q)
470 IF F>E THEN GOTO 280
480 REM "S-A GASIT POLIN. FACTOR"
485 REM
490 GOSUB 670
500 REM "INLOCUIREA POLINOMULUI "
510 REM
520 N=N-2
530 FOR I=1 TO N+1
540 A(I)=B(I+1)
550 NEXT I
560 GOTO 170
570 REM "POLINOM FACTOR"
580 REM
590 IF N=0 THEN GOTO 670
600 X=-A(2)/A(1)
610 Y=0
620 GOTO 620
630 P=A(2)/A(1)
635 Q=A(3)/A(1)
640 GOSUB 670
650 STOP
670 REM "REZOLVARBA EC. X^2+P+Q=0"
680 REM
690 D=P*P-4*Q
700 IF D=0 THEN GOTO 750
710 D=SQRT(D)
715 Y=0
720 X=-P+D/2
725 GOSUB 690
730 X=-P-D/2
735 GOSUB 690
740 RETURN
750 C=SQRT(-D)/2
755 X=-P/2
760 Y=0
765 GOSUB 690
770 Y=D
775 GOSUB 690
780 RETURN
800 REM "AFISAREA REZULTATULUI"
810 IF Y=0 THEN GOTO 850
820 PRINT "Y=";Y;";"
830 RETURN
840 PRINT Y
850 RETURN
870 PRINT "PROCES DIVERGENT"
880 END

```

12.15. Metoda celor mai mici pătrate

Metoda celor mai mici pătrate este o tehnică curentă de determinare a unei curbe $y=f(x)$ ce aproximează o mulțime de puncte date:

$$(Y_1, X_1), (Y_2, X_2), \dots, (Y_M, X_M).$$

Metoda constă în minimizarea sumei patratelor „distanțelor”: $d_1^2 + d_2^2 + \dots + d_M^2$, unde $d_i = y_i - f(x_i)$ este distanța între ordonata punctului dat și cea aproximată. Metoda este aplicată frecvent pentru funcții exponențiale sau polinomiale. În toate cazurile, metoda necesită rezolvarea unor sisteme de ecuații algebrice liniare, necunoscutele fiind coeficienții ecuației curbei. De exemplu, se dorește aproximarea a M puncte date prin curba $y=ax^b$. Pentru aceasta trebuie rezolvat un sistem de două ecuații cu necunoscutele a și b . Ecuațiile se obțin prin logaritmarea ecuației $y=ax^b$ și anularea derivatelor parțiale în raport cu $\log a$ și b :

$$M \cdot \log a + \sum_{i=1}^M (\log x_i) b = \sum_{i=1}^M \log y_i$$

$$\left(\sum_{i=1}^M \log x_i \right) \log a + \left(\sum_{i=1}^M (\log x_i)^2 \right) b = \sum_{i=1}^M (\log x_i) (\log y_i)$$

Aceste ecuații sînt liniare în $\log a$ și b .

Dacă se dorește aproximarea prin curba $y=ae^{bx}$, atunci trebuie să se rezolve sistemul:

$$M \cdot \log a + \left(\sum_{i=1}^M x_i \right) b = \sum_{i=1}^M \log y_i$$

$$\left(\sum_{i=1}^M x_i \right) \log a + \left(\sum_{i=1}^M x_i^2 \right) b = \sum_{i=1}^M x_i \log y_i$$

Pentru aproximarea printr-o funcție polinom: $Y=C_1+C_2X+\dots+C_{n+1}X^n$, coeficienții C_i se vor obține prin rezolvarea sistemului:

$$M \cdot C_1 + \left(\sum_{i=1}^M x_i \right) C_2 + \dots + \left(\sum_{i=1}^M x_i^n \right) C_{n+1} = \sum_{i=1}^M y_i$$

$$\left(\sum_{i=1}^M x_i \right) C_1 + \left(\sum_{i=1}^M x_i^2 \right) C_2 + \dots + \left(\sum_{i=1}^M x_i^{n+1} \right) C_{n+1} = \sum_{i=1}^M x_i y_i$$

$$\left(\sum_{i=1}^M x_i^n \right) C_1 + \left(\sum_{i=1}^M x_i^{n+1} \right) C_2 + \dots + \left(\sum_{i=1}^M x_i^{2n} \right) C_{n+1} = \sum_{i=1}^M x_i^n y_i$$

Toate aceste exemple conduc la rezolvarea unor sisteme de ecuații liniare pentru care vom utiliza instrucțiunile matriceale disponibile în limbajul BASIC.

Exemplu de program

Se dorește găsirea unei funcții (putere, exponențială, sau un polinom de grad mai mic decât b), care aproximează cel mai bine un număr de puncte date (maximum 100). Coordonatele x_i, y_i ale punctelor vor fi convertite în logaritmi lor de către program, dacă va fi necesar. Numărul de puncte date și tipul curbei de aproximare fiind variabile, se vor utiliza facilitățile de redimensionare a tablourilor.

Programul va imprimă ecuația curbei de aproximare cu coeficienții calculați, lista coordonatelor punctelor date și lista valorilor funcției ($y(x_i)$) cu care aproximează, precum și eroarea medie (suma pătratelor). Valoarea erorii va fi utilizată pentru alegerea tipului funcției de aproximare. Variabilele utilizate de program sînt următoarele:

X = vector de 100 de elemente care va conține valorile x_i introduse
 Y = vector de 100 de elemente care va conține valorile y_i introduse
 A = matrice de 10 linii și 10 coloane care va conține coeficienții necunoscuților sistemului de ecuații liniare

B = inversa matricei A

C = vector de 10 elemente conținând necunoscuțele sistemului.

D = vector de 10 elemente care conține al doilea membru al sistemului

M = numărul de puncte

N = variabilă care indică metoda de aproximare, dărită:

$N=0$ pentru funcție putere, $y=ax^b$

$N=1$ pentru funcție exponențială, $y=ae^{bx}$

$N=2, 3, \dots, 10$ pentru funcție polinom, $y = \sum_{i=1}^N C_i x^{i-1}$

$K1$ = numărul de ecuații simultane

$K1=2$ dacă $N=0$ sau 1

$K1=N$ dacă $N=2, 3, \dots, 10$

I = Programul funcționează astfel

1. Citește datele

a) Citește M valori pentru y_i și M valori pentru x_i

b) Citește valoarea lui N care dă tipul curbei de utilizat

2. Calculează $\log x_i$ și $\log y_i$ dacă $N=0$, sau $\log y_i$ dacă $N=1$, pentru $i=1, 2, \dots, M$.

3. Calculează elementele tablourilor $[A]$ și $[D]$, utilizând formulele corespunzătoare curbei date

4. Rezolvă sistemul de ecuații

5. Afișează ecuația curbei

6. Calculează valorile $y(x_i)$ pentru $i=1, 2, \dots, M$

7. Calculul erorii, suma pătratelor distanțelor $d_1^2 + d_2^2 + \dots + d_M^2$

8. Afișarea coordonatelor x_i, y_i și a valorii funcției $y(x_i)$, pentru $i=1, 2, \dots, M$, apoi afișarea erorii.

De notat că dacă $N=0$ sau 1, trebuie reconvertite valorile $\log y_i$ și $\log x_i$ în y_i și respectiv în x_i .

9. Se reprezintă grafic punctele (x_i, y_i) și apoi se trasează curba care unește punctele de interpolare calculate.
10. Dacă se dorește o nouă rulare cu aceleași date, se revine la punctul 1.b).

Instrucțiunile MAT ZER din liniile 200, 205 realizează redimensionarea matricei A și a vectorului D la fiecare nouă execuție a programului. În liniile 370 și 375 tablourile B și C sînt implicit redimensionate pentru a corespunde matricei A și vectorului D. Rezolvarea sistemului de ecuații este substanțial simplificată prin utilizarea instrucțiilor matriceale (liniile 370 și 375).

```

1 REM "*****"
2 REM "A APROXIMARE PRIN METODA N°"
3 REM "A CELOR MAI MARI PĂTRATE"
4 REM "*****"
5 DIM X(100), Y(100), Z(100)
6 REM "CITIREA DATELOR"
7 PRINT "CITE PUNCTE DEFINITE?"
8 PRINT "N="
9 INPUT N
10 PRINT "INTRODUCETI COORDONATELE Y="
11 MAT INPUT Y(N)
12 PRINT "INTRODUCETI ABSCISELE X="
13 MAT INPUT X(N)
14 REM "CITIREA DATELOR"
15 PRINT "INTRODUCETI N°0 PENTRU PUTEREA"
16 PRINT "      N=1 PENTRU EXPONENTIALA"
17 PRINT "      N=NK PENTRU POLINOM"
18 PRINT "N="
19 INPUT N
200 REM "CALCULUL LOGICII SI LOGICII"
210 REM "DACA E NECESAR"
220 IF N0=2 THEN 170
230 FOR I=1 TO N
240   Y(I)=LOG(Y(I))
250 NEXT I
260 IF N1 THEN 170
270 FOR I=1 TO N
280   X(I)=LOG(X(I))
290 NEXT I
170 REM "CALCULUL MATRICEI A"
175 REM "SI A VECTORULUI D"
180 N1=N
190 IF N1=2 THEN 200
195 N1=2
200 MAT A=ZER(N1,N1)
205 MAT D=ZER(N1)
210 A(1,1)=N
215 N2=2*N1-1
220 FOR I=1 TO N
230   S=0
240   FOR K=1 TO N
250     S=S+X(K)*(D1-I)
260   NEXT K
270   FOR J=1 TO N1
280     S2=D1-J+1
290     IF J=N1 THEN 285
300     IF S2=N1 THEN 285
310     A(I,S2)=S
320   NEXT J
330   IF D1=K1 THEN 205
340   FOR K=1 TO N
350     D(I)=D(I)+Y(K)*X(K)*(D1-I)
360   NEXT K
370 NEXT I
375 D1=D1-Y(N)
380 REM "REZOLVAREA SISTEMULUI"
385 REM

```

```

270 MAT D=INVIAS
275 MAT C=END
280 REM
285 REM "TIPARIREA ECUATIEI"
290 REM
295 IF N=1 THEN 450
300 C1=EXP(C(1))
305 IF N=1 THEN 450
310 PRINT "EUIEREA Y=";C1;"*X";C(2)
315 GOTO 450
320 PRINT "EXPONENTIALA Y=";C1;"*EXP(C(2));*X"
325 GOTO 450
330 IF C(2)=0 THEN 445
335 PRINT "POLINOM Y=";C(1);C(2);"X"
340 GOTO 450
345 PRINT "POLINOM Y=";C(1);"X";C(2);"X"
350 IF N=2 THEN 465
355 FOR I=3 TO N
360 IF C(I)=0 THEN 475
365 PRINT C(I);"X";I-1
370 GOTO 480
375 PRINT "+";C(I);"X";I-1
380 NEXT I
385 PRINT
390 REM
395 REM "AFISAREA REZULTATELOR"
400 REM
405 IF N=2 THEN 545
410 FOR I=1 TO M
415 Y(I)=EXP(C(1))
420 NEXT I
425 IF N=1 THEN 545
430 FOR I=1 TO M
435 X(I)=EXP(X(I))
440 NEXT I
445 PRINT
450 PRINT "EACATI" "Y=CALCULATI"
455 REM
460 FOR I=1 TO M
465 IF N=2 THEN 565
470 IF N=1 THEN 585
475 YI=C1*X(I)*C(2)
480 GOTO 515
485 YI=C1*EXP(C(2)*X(I))
490 GOTO 515
495 YI=C(I)
500 FOR J=2 TO N
505 YI=YI+C(J)*X(I)^(J-1)
510 NEXT J
515 S=S+(Y(I)-YI)^2
520 PRINT Y(I),YI
525 Z(I)=YI
530 NEXT I
535 PRINT
540 PRINT "ERORAREA=";S
545 REM "CALCULUL VALORILOR MAXIME"
550 REM "PENTRU CELE DOUA COORDONATE"
555 U1=ABS(X(1))
560 U2=ABS(X(2))
565 FOR I=2 TO N
570 IF U1>ABS(X(I)) THEN 575
575 U1=ABS(X(I))
580 IF U2>ABS(Y(I)) THEN 585
585 U2=ABS(Y(I))
590 NEXT I
595 WINDOW =J1,U1,-U2,U2
600 T=U2/SQ
605 INIT
610 FOR I=1 TO M
615 MOVE X(I),Y(I)+T
620 DRAW X(I),Y(I)
625 NEXT I
630 MOVE X(1),Z(1)
635 FOR I=2 TO M
640 DRAW X(I),Z(I)
645 NEXT I
650 PRINT "INTRODUCETI VALOAREA S PENTRU"
655 PRINT "GPRAREA EXECUTIET"
660 INPUT S
665 IF S<>1 THEN 15
670 END

```

12.10. Transformata Fourier rapidă

12.10.1.

Transformata Fourier constituie de mai mulți ani un instrument matematic utilizat în numeroase domenii, ca de exemplu în optică, acustică, fizică cuantică, telecomunicații, teoria sistemelor și a proceselor aleatoare etc.

Fourier a demonstrat că orice semnal periodic poate fi considerat ca o combinație de oscilații sinusoidale de frecvențe $f_0, 2f_0, 3f_0 \dots$. Astfel, dacă $x(t)$ reprezintă amplitudinea semnalului în funcție de timp, $x(t)$ se poate scrie ca suma unui număr oarecare de funcții sinusoidale. Deoarece pentru fiecare semnal, fazele inițiale nu sînt identice, sîm ωt devine $\sin(\omega t + \varphi)$:

$$\sin(\omega t + \varphi) = \sin \omega t \cos \varphi + \sin \varphi \cos \omega t$$

unde $\cos \varphi$ și $\sin \varphi$ pot fi considerați coeficienți.

Astfel, toate funcțiile periodice se pot descrie în felul următor:

$$\begin{aligned} x(t) = & A_0 + a_1 \sin \omega t + b_1 \cos \omega t \\ & + a_2 \sin 2\omega t + b_2 \cos 2\omega t \\ & + a_3 \sin 3\omega t + b_3 \cos 3\omega t \\ & + \dots \end{aligned}$$

unde: $\omega = \frac{2\pi}{T} = 2\pi f$ (T este perioada și f frecvența semnalului).

$a_1, a_2, \dots, b_1, b_2, \dots$ sînt constante care reprezintă amplitudinea fiecăreia din componente;

A_0 este valoarea medie.

Această relație reprezintă descompunerea în serie Fourier a funcțiilor periodice $x(t)$. În continuare, problema principală este calcularea coeficienților $A_0, a_1, b_1, a_2, b_2, \dots$, care se numesc coeficienții Fourier.

Deoarece A_0 este valoarea medie a semnalului periodic $x(t)$, acest coeficient se calculează imediat cu relația:

$$A_0 = \frac{1}{T} \int_0^T x(t) dt.$$

Fourier a demonstrat că ceilalți coeficienți se obțin din formulele

$$a_n = \frac{2}{T} \int_0^T x(t) \sin n\omega t dt$$

$$b_n = \frac{2}{T} \int_0^T x(t) \cos n\omega t dt.$$

Se definește transformata Fourier a unui semnal continuu $x(t)$ prin integrala:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-2\pi i f t} dt$$

Parametrii t și f reprezintă în general timpul și frecvența, dar foarte bine acest calcul se poate aplica la o mare varietate de fenomene unde t și f pot reprezenta alți parametri.

Transformata Fourier inversă se definește prin relația:

$$x(t) = \int_{-\infty}^{+\infty} X(f) e^{2\pi i f t} df$$

Prin utilizarea dispozitivelor numerice se obține adesea un semnal discret $x(n)$, printr-o eșantionare a semnalului inițial $x(t)$. În acest caz integrala Fourier definită mai sus poate fi înlocuită prin transformata Fourier discretă:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-2\pi i n k / N}$$

unde $k=0, 1, \dots, N-1$.

Mulțimea celor N valori $x(n)$ constituie o reprezentare discretă a funcției $x(t)$, iar cele N valori $X(k)$ o reprezentare a spectrului $X(f)$:

$$x(n) \Leftrightarrow x(t)$$

$$X(k) \Leftrightarrow X(f)$$

Algoritmul transformatei Fourier rapide permite reducerea timpului de execuție a unei transformate Fourier discrete.

Transformata Fourier discretă a unei secvențe finite de valori $\{x(n)\}$, $0 \leq n \leq N-1$ poate fi prezentată într-o manieră mai practică:

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot w^{nk}$$

unde

$$w^{nk} = e^{-2\pi i n k / N}$$

Această relație arată că pentru o secvență de N numere o evaluare directă necesită $(N-1)^2$ înmulțiri și $N \cdot (N-1)$ adunări. Pentru valori ale lui N , de exemplu 1000, ca ordin de mărime, un calcul direct implică o cantitate foarte mare de calcule, chiar și pentru un sistem puternic.

Principiul transformatei Fourier rapide constă în separarea secvenței inițiale de N valori în două secvențe mai scurte ale căror transformate discrete pot fi combinate pentru a produce transformata discretă a secvenței de N puncte.

Astfel, dacă N este par și secvența originală a fost împărțită în două secvențe de $N/2$ puncte, este necesar un număr de înmulțiri de ordinul $N^2/2$ pentru a evalua transformata discretă pe N puncte. Dacă $N/2$ este la rândul său par, atunci secvența de $N/2$ valori se poate de asemenea împărți în două secvențe de câte $N/4$ valori fiecare, pentru care se calculează independent transformatele și procedeu poate continua, obținându-se de fiecare dată creșterea vitezei cu un factor de aproximativ 2. Dacă N este o putere a lui 2, procesul se poate repeta pînă la calcularea în final a transformatelor pentru două puncte.

Pornind de la secvența de N valori $x(n)$, se generează două secvențe de $N/2$ puncte fiecare $x_1(n)$ și $x_2(n)$, prima conținând termenii de rang par din secvența inițială, iar a doua termenii de rang impar :

$$x_1(n) = x(2n)$$

$$x_2(n) = x(2n+1)$$

unde $n=0, 1, 2, \dots, N/2-1$

Se poate arăta că transformata Fourier discretă a secvenței inițiale se scrie făcând compunerea transformărilor Fourier pentru cele două secvențe inițiale $x_1(n)$ și $x_2(n)$:

$$X(k) = X_1(k) + W_N^k X_2(k)$$

unde $X(k)$, $X_1(k)$ și $X_2(k)$ sînt transformatele Fourier discrete ale secvențelor $x(n)$, $x_1(n)$ și $x_2(n)$ și :

$$W_N^k = e^{-j(2\pi/N)k}$$

Datorită proprietăților de periodicitate ale transformatei Fourier, relația precedentă se poate scrie de asemenea :

$$X(k) = X_1(k-N/2) - W_N^k X_2(k-N/2)$$

pentru k cuprins între $N/2$ și $N-1$, iar :

$$W_N^{k+N/2} = W_N^k \cdot W_N^{N/2} = -W_N^k$$

12.17. Simularea salturilor unei mingi

Se va descrie mișcarea unei mingi de cauciuc care sare sus-jos sub efectul gravitației proprii și în același timp se deplasează pe orizontală cu viteză constantă. Se presupun date : înălțimea inițială de la sol (H), viteza orizontală (V) și numărul de salturi ale mingii (N). Se dă de asemenea coeficientul de restituire (C) definit ca raportul vitezelor imediat după și înainte de salt.

Pentru a calcula poziția mingii în timp se dă un increment mic de timp (D) și se aplică legile fizicii pe acest interval :

$$T(I+1) = T(I) + D$$

$$X(I+1) = X(I) + V \cdot D$$

$$Z(I+1) = Z(I) - G \cdot D$$

$$Y(I+1) = Y(I) + 0.5 \cdot (Z(I) + Z(I+1)) \cdot D$$

Unde :

X este deplasarea pe orizontală (inițial nulă), Z este viteza pe verticală (de asemenea nulă la plecare), Y este înălțimea deasupra solului, G este accele-

rația gravitațională ($9,81 \text{ m/s}^2$). Indicii 1 și 1+1 corespund valorilor diferitelor variabile la începutul și la sfîrșitul incrementului de timp.

Dacă are loc o ciocnire pe durata incrementului de timp, formulele ce se aplică sînt puțin modificate. Prezența unei ciocniri este semnalată printr-o valoare negativă a lui $Y(1+1)$, ce ar fi imposibilă fizic. Cînd se produce aceasta, $Z(1+1)$ și $(1+1)$ se recalculează cu mai jos.

Mai întîi se calculează timpul necesar mingii pentru a atinge pămîntul, plecînd din poziția sa de la începutul incrementului de timp. Dacă se notează acest timp $D1$, atunci:

$$D1 = D * Y(1) / (Y_c(1) - Y_c(1+1))$$

Se calculează apoi viteza pe verticală imediat înainte de ciocnire:

$$Z = Z(1) - G * D1$$

Viteza pe verticală imediat după ciocnire va fi deci:

$$Z1 = -C * (Z(1) - G * D1)$$

Înălțimea deasupra solului la sfîrșitul incrementului de timp este:

$$Y(1+1) = 0.5 * (Z1 + Z_c(1+1)) * (D - D1)$$

ar viteza pe verticală la sfîrșitul incrementului de timp:

$$Z(1+1) = Z1 - G * (D - D1)$$

Programul funcționează astfel:

1. Citește valorile H , V , N , C și D și testează validitatea lor

2. Inițializează parametrii:

$$I = 1 \text{ (contor de incrementare)} \quad X(1) = 0$$

$$B = 0 \text{ (contor de salturi)} \quad Z(1) = 0$$

$$T(1) = 0 \quad Y(1) = H$$

3. Calculează deplasarea pe orizontală și pe verticală, și viteza pe verticală cu ajutorul formulelor de mai sus.

4. Dacă mingia lovește pămîntul pe parcursul incrementului de timp se testează dacă trebuie calculat saltul următor sau se termină programul.

a) Dacă $B < N$ se recalculează viteza pe verticală și deplasarea pe verticală cu formulele modificate, se incrementează contorul de salturi ($B = B + 1$) și se trece la incrementul de timp următor.

b) Dacă $B = N$ se calculează timpul și deplasarea pe orizontală la momentul ciocnirii.

5. Afășează valorile lui X și T urmate de o tabelare completă pentru T , X , Y și Z .

6. Trasează grafic Y funcție de T .

7. Se reîntoarce la pasul 1. Dacă se introduce $H=0$ se iese din program. Programul acceptă 100 de incrementări de timp. D trebuie ales astfel încât să existe între 8 și 20 de puncte de fiecare salt.

Calculul valorilor $T(I+1)$, $X(I+1)$, $Z(I+1)$ și $Y(I+1)$ este efectuat de un subprogram.

Se observă că de fapt programul integrează ecuația diferențială $d^2y/dt^2 = -g$ cu ajutorul metodei Euler modificată.

```

100 REM "*****"
105 REM "SIMULAREA SAJURII UNEI MINGI"
110 REM "*****"
115
120 REM "INTROUCEREA DATEI OR SI VERIFICAREA LOR"
125
130 DIM Z(100), Y(100), X(100), T(100), D(10)
135 PRINT "INALTIME INITIALA SI STOP:"
140 INPUT H
145 IF H=0 THEN GOTO
150 PRINT "INALTIME NEGATIV-ERORARE!"
155 GO TO 105
160 IF H<0 THEN GOTO
165 PRINT "VITEZA ORIZONTALA SI STOP:"
170 INPUT V
175 PRINT "VALOR DE SALTURI:"
180 INPUT N
185 IF N=0 THEN GOTO
190 PRINT "VALOR DE SALTURI NEGATIV-ERORARE!"
195 GO TO 105
200 PRINT "COEFICIENT DE RESISTINTA:"
205 INPUT C
210 IF C=1 THEN GOTO
215 PRINT "COEFICIENT 0 PARAMETRAR-ERORARE!"
220 GO TO 105
225 PRINT "INCREMENT DE TIMPU:"
230 INPUT D
235 IF D=0 THEN GOTO
240 PRINT "INCREMENT NEGATIV SAU NULL-ERORARE!"
245 GO TO 105
250 PRINT
255 REM "INITIALIZAREA PARAMETRELOR"
260
265 D=110-Y(1)/2110
270 Y(1)=H
275 G=9.8
280
285 REM "CALCULUL VITEZEI SI AL DEPLASARII"
290 REM "PENTRU FIECARE INCREMENT"
295
300 FOR I=1 TO 99
305   GOTO 310
310   IF Y(I)+D<0 THEN GOTO
315   IF D=0 THEN GOTO
320   Z(I)=V/(X(I)-Y(I)+1)
325   DZ=D
330   ZI=D*(Z(I)-G*D)
335   Z(I+1)=Z(I)+DZ
340   Y(I+1)=Y(I)+D*(Z(I)-G)
345   Y(I+1)=Y(I)+D*(Z(I)+Z(I+1)+D*(Z(I)-G))
350   B=Z(I)
355   IF Y(I)+D<0 THEN GOTO
360   IF D=0 THEN GOTO
365   DZ=D*(Z(I)+Z(I+1)+D*(Z(I)-G))
370   ZI=D*(Z(I)-G)
375   GO TO 310
380 NEXT I
385 T=99
390 GO TO 300
395
400 REM "ULTIMUL SALT"
405
410 D=D*(Y(I)/Y(I)-Y(I)+1)
415 GO SUB 220
420 T=T+1
425 T=T*(1)
430 ZI=X(I)
435
440 REM "AFISAREA REZULTATELOR NUMERICE"
445
450 PRINT "DISTANTA ORIZONTALA PARCURSA:"T(1) M"
455 PRINT "TIMPUL CERUT:"T(1) S"
460 PRINT
465 PRINT "Y(100), "DIST. ORIZ.", "INALTIME", "VIT. VORC."
470 PRINT "*****"
475 FOR I=1 TO 11
480   PRINT "Z=";Z(I), "X=";X(I), "Y=";Y(I), "Z=";Z(I)
485 NEXT I
490 PRINT
495 PRINT "SOLUTIA GRAFICA A PROBLEMEI(2/100):"
500
505 INPUT G
510 IF G=0 THEN GOTO
515 GOTO
520 WINDOW=20,100,10,50
525 LT=11/10
530 R=11*11/10
535 G=-G/10
540 T=110/10
545 WINDOW=L.P.D.
550 MOVE L,B
555 DRAW R,P
560 DRAW R,T
565 DRAW L,T
570 DRAW L,D
575
580 REM "TRASAREA AXELOR"
585
590 MOVE L,D
595 DRAW R,D
600 MOVE L,D
605 DRAW L,T
610
615 REM "TRASAREA CURBEI"
620
625 MOVE L,H
630 FOR I=2 TO 11
635   DRAW I,Z(I)
640 NEXT I
645 MOVE L,D
650 GO TO 405
655
660 REM "SUBPROGRAM DE CALCUL AL VITEZEI SI AL
665 REM "DEPLASARII LA SFIRSITUL INCREMENTULUI"
670
675 T(I+1)=T(I)+D
680 Y(I+1)=Y(I)+D*V
685 Z(I+1)=Z(I)+D*G
690 Y(I+1)=Y(I)+D*(Z(I)+G*(Z(I)+Z(I+1))+D*(Z(I)-G))
695 RETURN
700 END
    
```

12.18. Exerciții de despărțire a cuvintelor in silabe

Programul constituie un exemplu simplu de aplicare a învățării programate. Se afișează o regulă de despărțire in silabe, urmată de un exemplu. In continuare snt date mai multe exerciții de verificare pentru regula respec-

tivă. Regulile și exercițiile sînt codificate prin DATA începînd cu linia 500 din program, putîndu-se adăuga noi reguli și exerciții.

```

10 DIN EX102:
20 M = 1
30 INIT
40 PRINT "DESPARTIREA CUVINTELOR IN "
45 PRINT "-----"
50 PRINT "SILABA "
55 PRINT "-----"
60 PRINT "REGULA "IN
70 PRINT "-----"
80 GOTO 340
90 READ EX103
100 PRINT "EXEMPLU "
110 PRINT E$
120 PRINT
130 PRINT "EXERCITII "
140 READ EX104
150 IF EX104 = "0" THEN 180
160 PRINT E$
170 GOTO 340
180 READ R
190 I = 0
200 PRINT "RĂSPUNS "
210 INPUT E
220 IF E = K THEN 260
230 I = I + 1
240 PRINT "REVEDUTI REGULA "
250 GOTO 340
260 IF I < 4 THEN 280
270 PRINT "FELICITARI "
280 READ EX105
290 IF EX105 = "1" THEN 340
300 IF EX105 = "2" THEN 400
310 N = M + 1
320 IF N < 4 THEN 30
330 STOP
340 READ M
350 FOR I = 1 TO M
360 READ EX106
370 PRINT E$
380 NEXT I
390 RETURN
400 READ EX107
410 PRINT E$
420 GOTO 80
500 DATA 3
510 DATA "DACA VOCALA G URNATA DE O "
520 DATA "SINGURA CONSONANTA ACEASTA "
530 DATA "TRECE LA SILABA URMATOARE "
540 DATA "LG - GC ; O - RA "
550 DATA "OU - MA - RE = 1 "
560 DATA "DUN - A - RE = 2 "
570 DATA "DUN - AR - E = 3 "
580 DATA "D - UR - A - RE = 4 "
590 DATA "0",1,"1"
600 DATA "HOT - A - RI - RE = 1 "
610 DATA "HO - TAR - RE = 2 "
620 DATA "HOT - AR - (RE = 3 "
630 DATA "HO - TA - RI - RE = 4 "
640 DATA "0",4,"2"
650 DATA 4
660 DATA "DINTRE DOUA VOCALE SUCCESIVE "
670 DATA "CARE NU FORMEAZA DIPHTONG "
680 DATA "PRIMA APARTINE SILABEI DIMPAINTE "
690 DATA "A DOUA CELEI URMATOARE "
700 DATA "CE - RE - A - LO ; LU - A "
710 DATA "IH - DI - UI - DUA - LI - TA - TR = 1 "
720 DATA "IN - DI - VI - DU - A - LI - TA - IG = 2 "
730 DATA "0",2,"1"
740 DATA "PO - DO - A - UA = 3 "
750 DATA "PO - DE - UA = 2 "
760 DATA "0",2,"2"
770 DATA 3
780 DATA "DACA G URNATA DE DOUA "
790 DATA "SAU MAI MULTA CONSONANTA "
800 DATA "PRIMA CONSONANTA TRECE LA SILABA "
810 DATA "DIMPAINTE/CEALALTA (CELELALTE) "
820 DATA "LA SILABA URMATOARE "
830 DATA "E - XIE - TA - SU - MCC - TA "
840 DATA "MU - NTE - NE - N = 1 "
850 DATA "MUNT - E - N - TA = 2 "
860 DATA "MUN - TE - R - TA = 3 "
870 DATA "MUN - TE - VE - A = 4 "
880 DATA "0",1,"1"
890 DATA "PER - SUN - A - LI - TA - TE = 1 "
900 DATA "PER - UN - A - LI - TA - TE = 2 "
910 DATA "PER - SUN - AL - I - TA - TE = 3 "
920 DATA "PER - SU - NA - LI - TA - TE = 4 "
930 DATA "0",4,"3"
940 DATA "EXISTA EXCEPTII LA ACEASTA REGULA "
950 DATA 3
960 DATA "CIND PRIMA CONSONANTA ESTE "
970 DATA "R,C,D,G,P,T,K,F (AR A DOUA L,R "
980 DATA "ANIMACIUA TRECE LA SILABA URMATOARE "
990 DATA "A - NRE - VI - A ; A - CRU "
1000 DATA "CU - RU = 1 "
1010 DATA "CU - DRU = 2 "
1020 DATA "0",2,"1"
1030 DATA "SU - LU = 1 "
1040 DATA "SU - PLU = 2 "
1050 DATA "0",2,"1"
1060 DATA "DE - MCC - RAT = 1 "
1070 DATA "DE - NO - CRAT = 2 "
1080 DATA "0",2,"1"
1090 DATA "NI - EGT - RU = 1 "
1100 DATA "NI - SB - TRU = 2 "
1110 DATA "0",2,"2"
2000 END

```

12.19. Verificarea cunoștințelor de geografie

Exemplul ales în cadrul programului este axat pe verificarea cunoștințelor de geografie a reședințelor de județ. Examinatului i se solicită să răspundă la 10 întrebări, afișînd un număr de 10 județe alese aleator și cerînd să se introducă numele reședinței județului respectiv. După parcursirea celor 10 întrebări, se afișează numărul de răspunsuri corecte. Dacă se răspunde greșit la o întrebare, se indică și răspunsul corect.

Programul poate fi extins fără dificultăți pentru orice gen de teste asemănătoare (verificarea cunoștințelor regulelor de circulație, formule, definiții etc.)

```

10 PRINT " VERIFICAREA CUNOSTINTELOR "
15 PRINT " LA GEOGRAFIE "
20 PRINT " DENUMIRE JUDEȚ - BESEȘINTA "
25 DIM M$(50),S$(50),V$(50)
30 M = 0
35 T = 30
40 FOR J = 1 TO 10
45 FOR I = 1 TO INTERNUAL(I)-1
50 READ M$(I),S$(I),V$(I)
55 NEXT I
60 PRINT " DATI BESEȘINTA JUDEȚULUI "
65 INPUT V$(T)
70 IF V$ = S$ THEN 85
75 M = M+1
80 GOTO 70
85 PRINT " RĂSPUNS CORECT NR FI "
90 RESTORE
95 NEXT I
100 PRINT " DIN 10 ÎNTREBĂRI "
110 STOP
120 DATA " JUDEȚ " " BESEȘINTA "
130 DATA .

```

```

400 DATA " JUDEȚ N " ; " BESEȘINTA "
410 END

```

12.20. Verificarea cunoștințelor unui grup de candidați

Inițial, programul solicită prin dialog crearea tabelului cu întrebări. Fiecărei întrebări, operatorul îi asociază 3 răspunsuri posibile cu indicarea răspunsului corect. Se cere apoi numărul total de persoane de examinat. Pentru

```

2 DIM B(100)
3 DIM U(100,30),T(100),S(100)
4 MAT B = ZER
5 INIT
6 PRINT " INTRODUCETI ÎNTREBĂRILE? (DA/NU) "
7 INPUT AS
8 IF AS = "DA" THEN 1000
10 PRINT " DATI NUMERE DE RĂSPUNSURI CORECTE? (DA/NU) "
11 INPUT AS
12 IF AS = "DA" THEN 1170
13 IF B(1) < 0 THEN 30
14 PRINT " FARA RĂSPUNSURI INTRODUSE "
15 STOP
20 PRINT " VERIFICAREA 30 CUNOSTINTE "
30 PRINT " CITE PERSOANE EXAMINATE "
40 INPUT K
50 MAT U = ZER
60 MAT T = ZER
70 REM " ÎNCEPE VERIFICAREA "
80 FOR I = 1 TO M
90 FOR J = 0 TO M-1
95 IF U = 0 THEN 140
100 PRINT G+(J+1)
110 PRINT R+(J+1)
120 PRINT R+(J+2)
130 PRINT R+(J+3)
140 PRINT " DATI RĂSPUNSUL (1,2,3) "
150 INPUT R
160 IF U+(J+1) = R THEN 210
170 PRINT " RĂSPUNS ERONAT "
180 PRINT " RĂSPUNS CORECT ESTE "
185 IF U = 0 THEN 195
190 PRINT R+(J+1)+(370)
192 GOTO 230
195 PRINT R+(J+1)
200 GOTO 230
210 PRINT " RĂSPUNS CORECT "
220 U[(J+1)] = 1
230 NEXT J
240 B = 0
250 FOR X = 1 TO M
260 B = B+U(X,K)
270 NEXT X
280 T(1) = 0
290 PRINT " DIN "N" ÎNTREBĂRI CUNOȘTE "
295 PRINT 0
300 NEXT I
310 J = 0
320 FOR I = 1 TO M
330 J = J+T(I)
340 NEXT I
350 K = J/M
370 PRINT " ÎN MODUL CUNOȘC "

```

```

380 FOR I = 1 TO N
390 S(I) = (T(I)*100)/M
400 NEXT I
405 REM * GRAFICUL. X=NR. INTREBARI, Y=CITI STIU *
410 INITP
420 WINDOW -10,N*10,-10,110
430 MOVE -10,0
440 DRAW N*10,0
450 MOVE 0,-10
460 DRAW 0,110
470 MOVE -1,25
480 DRAW 1,25
490 MOVE -2,50
500 DRAW 2,50
510 MOVE -1,75
520 DRAW 1,75
530 MOVE -2,100
540 DRAW 2,100
550 FOR I = 1 TO N
560 MOVE 1,0
570 DRAW 1,S(I)
580 NEXT I
590 STOP
1000 REM * INITIALIZARE *
1005 U = 1
1010 DIM Q$(20,30),R$(3*20,30),B(20)
1020 PRINT * CITE INTREBARI INTRODUCETI *
1030 INPUT N
1040 PRINT * DATI INTREBAREA SI CELE *
1050 PRINT * TREI RASPUNSURI *
1060 FOR I = 0 TO N-1
1070 PRINT * INTREBAREA ",I+1
1080 INPUT Q$(I+1)
1090 FOR J = 1 TO 3
1100 PRINT * RASPUNSUL ",J
1110 INPUT R$(I+1,J)
1120 NEXT J
1130 PRINT * DATI NUMARUL RASPUNSULUI *
1135 PRINT * CORECT *
1140 INPUT B(I+1)
1150 NEXT I
1160 GOTO 20
1170 PRINT * CITE INTREBARI AVETI? *
1175 INPUT N
1175 FOR I = 0 TO N-1
1175 PRINT * RASPUNSUL LA INTREBAREA ",I+1
1180 INPUT B(I+1)
1190 NEXT I
1200 GOTO 20

```

fiecare persoană în parte, se parcurge tot setul de întrebări, cel examinat indicând de fiecare dată răspunsul pe care îl consideră ca fiind corect. La răspuns incorect, programul afișează pe cel corect, în vederea însușirii lui. După parcurgerea setului de întrebări, se afișează totalul răspunsurilor corecte date de examinat.

La epuizarea examinării tuturor persoanelor, se afișează numărul mediu de răspunsuri corecte, caracteristic grupului examinat, și se trasează graficul cu numărul de răspunsuri corecte pentru fiecare persoană în parte.

12.21. Ordonarea candidaților după mediile obținute

Exemplul de față realizează ordonarea candidaților după mediile obținute la 5 probe. Se introduce numărul total al candidaților, urmat de numele fiecărui candidat și calificativele obținute la cele 5 probe. Se afișează numele candidaților și media obținută, în ordinea descrescătoare a mediilor.

Prin modificarea liniei 17 se poate modifica numărul de probe.

```

5  PRINT * ORDONARACANDIDATILOR DUPA NR020 *
10 PRINT * DATI NUMARUL CANDIDATELOR *
15 INPUT N
17 P = 5
20 DIM A(N),C(0),N(N),CIP>
22 N(1) = 2EM
25 PRINT * DATI NUMELE SI CALIFICATIVUL *
30 FOR I = 1 TO N
32 INPUT A$(I),C(I)
40 N(I) = C
42 N(1) = 0
44 N(1) INPUT C
46 FOR J = 1 TO P
48 N(I) = N(I)+C(J)
50 NEXT J
52 N(I) = N(I)/P
54 NEXT I
60 I = 1
80 I = N+1
90 I = 1+1
95 I = 1
100 IF N(I) <= 0 THEN 115
105 C = N(I)
110 K = 1
115 IF J = N THEN 130
120 I = 1+1
125 GOTO 100
130 REM * INVERSAREA ORDINII *
135 B(1) = A$(I),C(I)
140 A$(1) = A$(K),C(K)
145 A$(K) = B(1)
150 C = N(I)
155 N(I) = N(K)
160 N(K) = C
165 I = 1+1
170 IF I < N THEN 85
175 REM * IMPARIREA *
180 PRINT * NUMELE *
185 FOR I = 1 TO N
190 PRINT A$(I),C(I)
195 NEXT I
200 STOP
210 END

```


13.1. Trasarea strofoidei

Programul trasează strofoida, conform ecuației:

$$Y^2 = \frac{X^3(X+a)}{a-X}$$

Se solicită prin dialog distanța punctului A față de origine. (fig. 13.1)

```

5  PRINT "STROFOIDA"
10 PRINT "DAȚI VALOAREA A"
15 INPUT A
20 X = A/3
25 Y = SQR(X^2*(X+A)/(A-X))
30 INIT
35 WINDOW -Y,Y,-Y,Y
40 MOVE -Y,0
45 DRAW Y,0
50 MOVE 0,-Y
55 DRAW 0,Y
60 MOVE X,Y
65 FOR X = A/3 TO -A STEP -1
70 Y = SQR(X^2*(X+A)/(A-X))
75 IF X <= 0 THEN 90
80 DRAW X,Y
85 GOTO 95
90 DRAW X,-Y
95 NEXT X
100 FOR X = -A TO A/3
105 Y = SQR(X^2*(X+A)/(A-X))
110 IF X >= 0 THEN 125
115 DRAW X,Y
120 GOTO 130
125 DRAW X,-Y
130 NEXT X
135 STOP
140 END
    
```

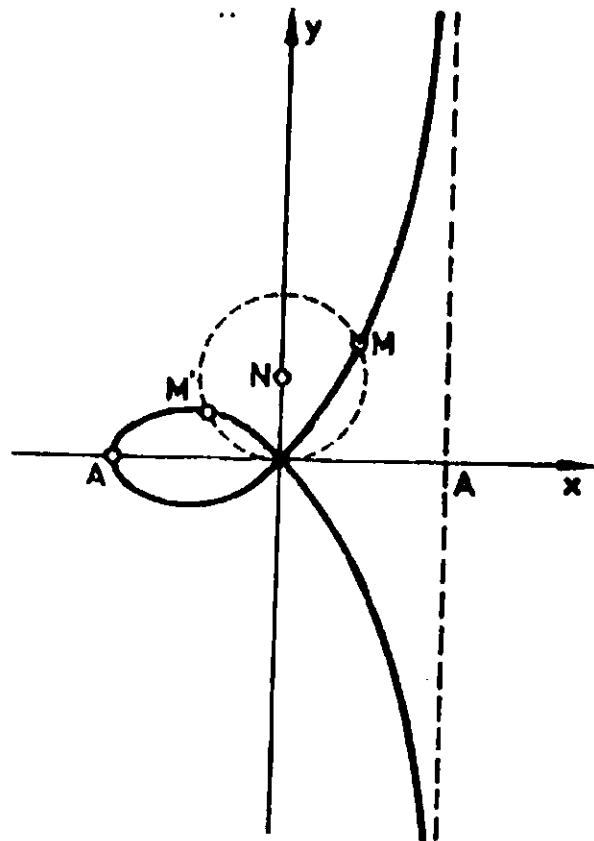


Fig. 13.1. Strofoida.

13.2. Trasarea cicloidei

Programul trasează cicloida, conform ecuației:

$$X = r(t - \sin t)$$

$$Y = -r(1 - \cos t)$$

unde: (fig. 13.2)

r este raza cercului,

t este unghiul de rotație,

λ este parametru:

λ este 1 punct pe cerc

$\lambda > 1$ punct exterior cercului

$\lambda < 1$ punct interior cercului

Se solicită prin dialog valorile pentru r , λ precum și numărul total de cicluri de trasat.

```

5 PRINT " CICLOIDA "
10 PRINT " DATI RAZA CERCULUI "
15 INPUT R
20 PRINT " DATI PARAMETRUL L "
30 INPUT L
40 PRINT " CITE CICLURI? (1,2,3) "
50 INPUT N
55 IF N <= 3 THEN 65
60 N = 3
65 A = -2*R
70 B = N*2*PI*10
75 INITP
80 WINDOW A,B,A,B
85 REN " TRASAREA AXELOR "
90 MOVE A,0
95 DRAW B,0
100 MOVE 0,A
105 DRAW 0,B
110 REN " TRASAREA CERCULUI "
115 MOVE R,R
120 FOR J = 0 TO 2*PI STEP PI/10
125 DRAW R*COSE(J),R*R*SENE(J)
130 NEXT J
135 X = 0
140 Y = R*(1-L)
145 MOVE X,Y
150 FOR J = 0 TO 2*PI STEP PI/10
155 X = R*(J-L*SENE(J))
160 Y = R*(1-L*COSE(J))
165 DRAW X,Y
170 NEXT J
175 STOP
180 END

```

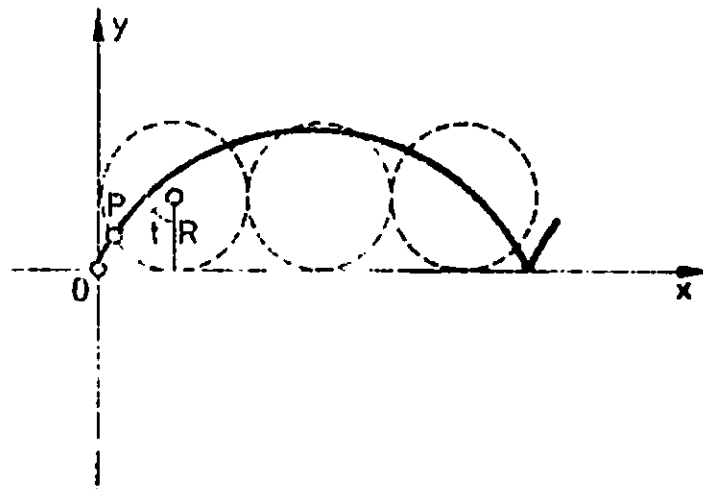


Fig. 13.2. Cicloida

13.3. Trasarea epicicloidei

Programul trasează epicicloida, după ecuațiile:

$$X = (R+r) \cos \frac{r}{R} t - r \cos \frac{R+r}{R} t$$

$$Y = (R+r) \sin \frac{r}{R} t - r \sin \frac{R+r}{R} t$$

unde : (fig. 13.3)

R — raza cercului fix

r — raza cercului mobil

l — unghiul format de dreapta care unește centrul celor 2 cercuri cu axa X.

Forma curbei depinde de raportul r/R ; dacă $r=R$ se obține graficul cardioidii.

Se solicită prin dialog valorile pentru r și R și se trasează epicycloida.

```

5  PRINT " EPICYCLOIDA "
10 PRINT " DATI RAZA CERCULUI FIX "
15 INPUT R
20 PRINT " DATI RAZA CERCULUI MOBIL "
25 INPUT L
30 INITP
35 A = R+2*L
40 WINDOW -A,A,-A,A
45 MOVE -A,0
50 DRAW A,0
55 MOVE 0,-A
60 DRAW 0,A
65 MOVE R,0
70 FOR X = 0 TO 2*PI STEP PI/10
75 DRAW R=COS(X),R=SIN(X)
80 NEXT X
85 MOVE R,0
90 A = R+l.
95 B = L/R
100 C = A/R
105 H = 2*PI*R/L
110 FOR X = 0 TO H STEP PI/10
115 Z = A*COS(B*X)-L*COS(C*X)
120 Y = A*SIN(B*X)-L*SIN(C*X)
125 DRAW Z,Y
130 NEXT X
135 STOP
140 END

```

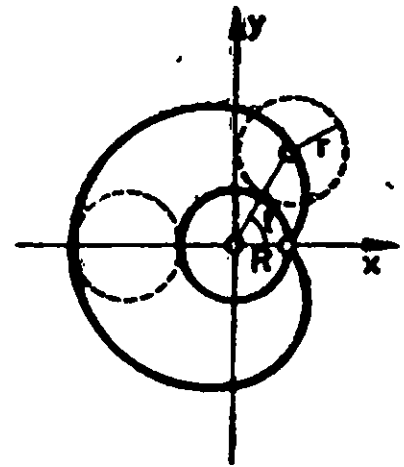


Fig. 13.3. Epicycloida.

13.4. Trasarea melcului lui Pascal

Programul trasează melcul lui PASCAL pe baza ecuațiilor :

$$X = 2R \cos^2 \varphi + a \cos \varphi$$

$$Y = 2R \cos \varphi \sin \varphi + a \sin \varphi$$

unde (fig. 13.4) :

R este raza cercului

φ este unghiul dreptei care se rotește față de axa X

a este parametrul de care depinde forma curbei
(dacă $a/R = 2$ se obține graficul cardioidii)

Se solicită prin dialog valorile pentru R și pentru raportul a/R (nu se dă valoarea direct pentru a).

Din coordonatele centrului cercului și coordonatele unui vîrf al triunghiului se poate determina raza cercului.

$$R = \sqrt{|B - X_1|^2 + |C - Y_1|^2}$$

Se inițializează ecranul și se determină spațiul utilizator: -100, 100 - -100, 100. Se trasează triunghiul și cercului circumseris. Se afișează coordonatele centrului cercului și mărimea razei.

```

10 PRINT " CERCUL CIRCUNSCRIS "
20 DIM A(2),B(2),C(2)
30 PRINT " DATI COORDONATELE VÂRFURILOR "
40 MAT INPUT A,B,C
50 MAT D = A
60 MAT E = B
70 GOSUB 600
80 IF Y = 0 THEN 400
90 S = R
100 T = B
110 MAT B = C
120 MAT E = B
130 GOSUB 600
140 IF Y = 0 THEN 460
150 Y = (3*B-R*T)/(5-R)
160 IF S > 0 THEN 190
170 X = (Y-B)/R
180 GOTO 200
190 X = (Y-T)/S
200 B = ABS(X-A(1))
210 C = ABS(Y-A(2))
220 R = SQRT(B^2+C^2)
230 INITF
240 WINDOW -100,100,-100,100
250 MOVE -100,0
260 DRAW 100,0
270 MOVE 0,-100
280 DRAW 0,100
290 MOVE A(1),A(2)
300 DRAW B(1),B(2)
310 DRAW C(1),C(2)
320 DRAW A(1),A(2)
330 MOVE X+R,Y
340 FOR I = 0 TO 2*PI STEP PI/10
350 DRAW X+R*COS(I),Y+R*SIN(I)
360 NEXT I
370 PRINT AT(1,1);"RAZA = ";R
380 PRINT AT(2,1);"X = ";X,"Y = ";Y
390 STOP
400 MAT D = C
410 MAT E = B
420 GOSUB 600
430 IF Y = 0 THEN 510
440 S = R
450 T = B
460 MAT D = A
470 MAT E = C
480 GOSUB 600
490 IF Y = 0 THEN 510
500 GOTO 190
510 PRINT " PUNCTE COLINIARE "
520 GOTO 30
600 Y = 0
610 I = D(2)-E(2)
620 IF I = 0 THEN 730
630 R = -(B(1)-E(1))/I
640 X = E(1)
650 IF B(1) < D(1) THEN 670
660 X = B(1)
670 X = X+ABS((B(1)-E(1))/2)
680 Y = B(2)
690 IF B(2) < D(2) THEN 710
700 Y = B(2)
710 Y = Y+ABS((B(2)-E(2))/2)
720 B = R*(0-X)+Y
730 RETURN
740 END

```

13.6. Graficul funcției polinomiale

Se introduce gradul polinomului și valorile tuturor coeficienților (zero pentru coeficienții care lipsesc). Se introduce apoi intervalul de definiție al funcției și numărului de pași în care se face trasarea graficului. Un număr mai mare de pași face trasarea mai exactă, dar mai lentă. Se va corela acest număr cu mărimea intervalului de definiție.

Subprogramul de la linia 500 calculează valoarea funcției pentru o valoare dată a variabilei. La linia 170 se apelează cu limita inferioară a intervalului de definiție. În bucla 200—280 se reține valoarea minimă și maximă a funcției (în P și R) pe intervalul de definiție. Spațiul utilizator se definește în funcție de limitele de definiție și de valorile P și R în linia 330. Se trasează axele OX și OY și forma funcției pe acest interval.

```

10 PRINT "GRAFICUL PUNCTIEI POLINOMIALE "
15 PRINT " DAT PRIN GRAD SI COEFICIENTI "
20 PRINT " DATI GRADUL PUNCTIEI "
30 INPUT G
40 DIM C(G+1)
50 PRINT " DATI COEFICIENTII "
60 MAT INPUT C
70 DIM L(2)
80 PRINT " DATI DOMENIUL DE DEFINITIE "
90 MAT INPUT L
100 PRINT " DATI NUMARUL DE PASI "
110 INPUT H
120 IF L(1) < L(2) THEN 150
130 PRINT " LIMITE SRODATE "
140 GOTO 90
150 H = (L(2)-L(1))/H
160 X = L(1)
170 GOSUB 300
180 P = P
190 R = P
200 FOR K = 1 TO H
210 X = X+H
220 GOSUB 300
230 IF P >= P THEN 240
240 P = P
250 GOTO 200
260 IF P <= R THEN 280
270 R = P
280 NEXT K
290 IF R > P THEN 320
300 PRINT " PUNCTIE CU VALOARE CONSTANTA "
305 PRINT " IN INTERVALUL DAT = ",R
310 STOP
320 INITP
330 WINDOW L(1),L(2),P,R
340 MOVE L(1),0
350 DRAW L(2),0
360 MOVE 0,P
370 DRAW 0,R
380 X = L(1)
390 GOSUB 300
400 MOVE X,P
410 FOR K = 1 TO H
420 X = X+H
430 GOSUB 300
440 DRAW X,P
450 NEXT K
460 STOP
500 REM " CALCULUL VALORII PUNCTIEI "
510 F = C(1)
520 FOR I = 2 TO G
530 F = F*X+C(I)
540 NEXT I
550 RETURN
560 END

```

13.7. Suma grafică a mai multor vectori

Se solicită prin dialog numărul total N al vectorilor de însumat, mărimea fiecăruia, și unghiul lor față de orizontală.

Masivul L (N) conține lungimile, iar A (N) unghiurile în grade.

Pentru fiecare vector se calculează coordonatele vârfului vectorului considerând vectorul din originea axelor :

$$X(I) = L(I) \cdot \cos(A(I) \cdot \pi / 180)$$

$$Y(I) = L(I) \cdot \sin(A(I) \cdot \pi / 180)$$

Coordonata vârfului vectorului sumă se obține din formulele :

$$A = \sum_{I=1}^N X(I)$$

$$B = \sum_{I=1}^N Y(I)$$

iar lungimea vectorului sumă și unghiul cu orizontala :

$$L = \sqrt{A^2 + B^2}$$

$$C = (\arctg(B/A)) \cdot 180/\pi$$

```

10 PRINT "ADUNAREA VECTORILOR "
20 PRINT "NR. VECTORI DE ADUNAT "
30 INPUT N
40 DIM L(N),A(N),X(N),Y(N)
50 PRINT "DATI MARIIMEA SI UNGIUL "
55 PRINT "CU AXA X A VECTORILOR "
60 FOR I = 1 TO N
70 INPUT L(I),A(I)
80 NEXT I
90 FOR I = 1 TO N
100 X(I) = L(I)*COS(A(I)*PI/180)
110 Y(I) = L(I)*SIN(A(I)*PI/180)
120 NEXT I
130 A = X(1)
140 B = Y(1)
150 FOR I = 2 TO N
160 A = A+X(I)
170 B = B+Y(I)
180 NEXT I
185 INITP
190 L = SQR(A^2+B^2)
200 C = ATN(B/A)
210 PRINT AT(1,1): "LUNGIMEA " ; L
220 PRINT AT(2,1): "UNGIUL " ; C*180/PI
230 WINDOW -1,A*1,-1,B*1
240 MOVE -1,0
250 DRAW A*1,0
260 MOVE 0,-1
270 DRAW 0,B*1
280 MOVE 0,0
290 FOR I = 1 TO N
300 DRAW X(I),Y(I)
310 NEXT I
320 MOVE 0,0
330 DRAW A,B
340 STOP
350 END

```

După inițializarea ecranului, spațiul utilizator se stabilește în funcție de coordonatele A, B, se trasează axele și prin bucla 290—310 utilizând instrucțiunea RMOVE vectorii de adunat avînd coordonatele relative X (I), Y (I).
Se trasează și vectorul sumă.

13.8. Mișcarea unui punct material într-un cîmp gravitațional

Se consideră un punct material care pleacă cu viteza inițială V_0 dintr-un punct A și trebuie să atingă un punct B aflat la o distanță D față de A. Mișcarea punctului este dată de ecuația:

$$Y = X \operatorname{tg} \omega - X^2 \frac{g}{V_0^2 \cos^2 \omega}$$

unde :

V_0 este viteza inițială

ω este unghiul de aruncare față de orizontală.

Se consideră de asemenea că între punctele A și B poate exista o diferență de nivel N (punctele nu se află obligatoriu ambele pe axa orizontală).

Programul solicită prin dialog distanța între cele două puncte, diferența de nivel între cele două puncte, viteza inițială a mobilului și unghiului său de aruncare.

```

10 PRINT " BALLISTIC "
20 PRINT " DISTANTA DE TRAGERE "
30 INPUT B
40 PRINT " DIFERENTA DE NIVEL "
45 PRINT " POZITIV SAU NEGATIV "
50 INPUT N
60 M = N*44/D
70 INITP
80 X = 10
90 Y = 10
100 U = 54
110 Z = 10
120 IF N <> 0 THEN 150
130 Y = Y*ABSIN(
140 GOTO 140)
150 Z = Z*N
160 WINDOW 0,64,0,44
170 PLOT X,Y
180 PLOT U,Z
190 MOVE X,Y
200 PRINT AT(29,21)," VITEZA INITIALA "
210 INPUT U
220 PRINT AT(30,21)," UNGIUL IN GRAD "
230 INPUT U
240 U = PI*U/180
250 I = TAN(U)
260 C = 5/1000/COS(U)**2
270 I = 0
280 FOR K = 0 TO 44
290 A = (X+Y-C*K**2)
300 DRAW X+K,Y+A
310 IF K < 44 THEN 300
320 IF ABSIZ-A) < 1 THEN 350
330 PRINT AT(30,27)," LOUI "
340 I = I
350 NEXT K
360 IF I < 1 THEN 10
370 GOTO 190
380 END

```


Se trasează graficul traiectoriei punctului, specificându-se la sfârșit dacă s-a atins sau nu punctul B.

Dacă nu a fost atins punctul B se cere introducerea noilor valori de viteză inițială a mobilului și a unghiului de aruncare pînă se găsește combinația adecvată.

13.9. Generarea și modificarea unei figuri

Programul generează desenul unui scaun văzut în perspectivă (fig. 13.5), solicitînd prin dialog dimensiunile l_1 , l_2 , l_3 și l_4 . După generare figura poate fi modificată succesiv, schimbînd una sau mai multe dimensiuni. Se pot obține astfel o serie de reprezentări ale aceluiași obiect, cu diferite proporții între dimensiunile principale.

Programul poate fi extins cu ușurință pentru generarea de figuri mai complicate, generarea mai multor figuri la diferite distanțe între ele, permutarea lor etc. ajutînd la proiectarea corpului respectiv.

```

5 PRINT " PROIECTARE ASISTATA "
10 PRINT " DATI LUNGIME, LĂTIMĂ ȘEZUT SCAUN "
15 INPUT L1,L2
20 PRINT " DATI ÎNĂLȚIME SPATAR "
25 INPUT L3
30 PRINT " DATI ÎNĂLȚIME PICIOR "
35 INPUT L4
40 X1 = 50R(L1*2/2)
50 Y1 = X1
60 X2 = 50R(4*L2*2/5)
70 Y2 = X2/3
80 X3 = 50R(L3*2/4)
90 Y3 = 3*X3
100 A = X2-X1-X3+20
110 B = L4*Y2-Y1+Y3+20
120 INITP
130 WINDOW 0,A,0,B
140 MOVE 10,10
150 AHOVE 0,Y2
160 RBRAU 0,L4
170 RBRAU X1,Y1
180 RBRAU X2,-Y2
190 RBRAU -X1,-Y1
200 RBRAU 0,-L4
210 RBRAU 0,L4
220 RBRAU -X2,Y2
230 RBRAU X1,Y1
240 RBRAU X3,Y3
250 RBRAU X3,-Y2
260 RBRAU -X3,-Y3
270 RBRAU 0,-L4
280 PRINT " CONTINUATE (DA=1) "
290 INPUT C
300 IF C = 1 THEN 30
310 STOP
320 END

```

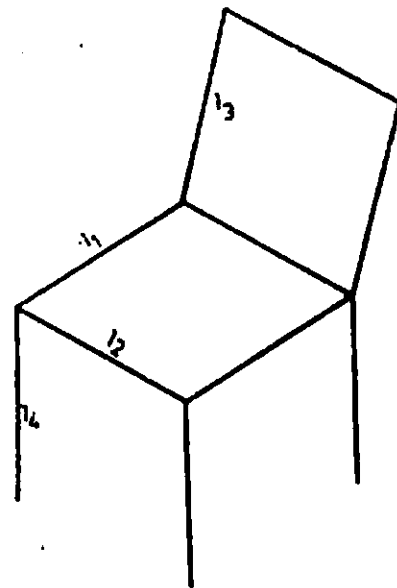


Fig. 13.5. Scaun văzut în perspectivă.

13.10. Generarea de figuri tridimensionale conform legilor perspectivei

Programul iese din sfera exemplelor demonstrative, fiind un instrument util pentru proiectarea asistată de calculator. Programul permite generarea de obiecte cu muchii drepte în spațiul tridimensional marcat de axele OX,

OY, OZ. Obiectele se compun prin asocierea așa-numitor segmente de dreaptă unitare. Lungimea unui segment unitar poate fi modificată pe parcurs. Indicația direcției de trasare se face prin tastele asociate deplasărilor pe cele 3 axe de coordonate. Odată încheiată faza de generare a obiectului, acesta poate fi privit din orice punct din exteriorul său (ca și cum privitorul s-ar roti în jurul obiectului). După dorință, obiectele generate pot fi completate sau modificate ulterior.

Imaginea în perspectivă a unui obiect a fost creată simulând (aproximativ) fenomenul de creare a unei imagini pe retina ochiului. S-au luat în considerare următoarele ipoteze simplificatoare:

- a) privirea este îndreptată totdeauna spre originea axelor de coordonate
 - b) cristalinul este o lentilă subțire convergentă, iar retina un plan ortogonal pe direcția privirii, la distanța de 3 cm față de centrul cristalinului.
- S-a utilizat formula lentilelor convergente:

$$\frac{1}{o} + \frac{1}{i} = \frac{1}{f}$$

unde (fig. 13.6):

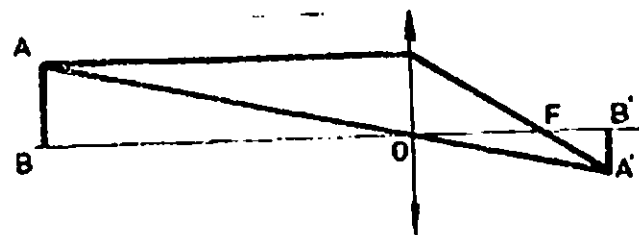
- O este distanța între obiect și lentilă (OB)
- i este distanța între obiect și imagine (OB')
- f este distanța focală (OF)
- A este punctul de proiectat
- P este proiecția punctului A pe planul retinei
- \bar{i} , \bar{j} sint versorii axelor de coordonate în planul retinei:

$$\bar{i} = \frac{\bar{n} \times \bar{e}_3}{\|\bar{n} \times \bar{e}_3\|}; \quad \bar{e}_3 = (0,0,1)$$

$$\bar{j} = \frac{\bar{n} \times \bar{i}}{\|\bar{n} \times \bar{i}\|}$$

Imaginea obținută prin lentilă fiind răsturnată, la proiecția pe ecran ea va fi din nou răsturnată pentru obținerea imaginii corecte.

Fig. 13.6. Lentile convergente.



Poziția punctului P în spațiu se obține intersectând planul retinei cu dreapta AP:

$$\begin{cases} \bar{b} = t_b \cdot \bar{n} + \bar{a} \\ \bar{n} \cdot \bar{b} - \|\bar{n}\|^2 = 0 \end{cases}$$

Punctul B se află la intersecția planului lentilei cu raza de lumină paralelă cu direcția privirii ce pornește din C. Vom avea :

$$\begin{cases} \bar{b} = t_b \cdot \bar{n} + \bar{c} \\ \bar{n} \cdot \bar{b} - \|\bar{n}\|^2 = 0 \end{cases}$$

de unde rezultă :

$$\bar{b} = \frac{\|\bar{n}\|^2 - \bar{n} \cdot \bar{c}}{\|\bar{n}\|^2} \cdot \bar{n} + \bar{c}.$$

```

5  INITP
10  DIM G(254,12),R(254)
30  PRINT AT(2,1); " N = FIGURA NOUA "
40  PRINT AT(3,1); " U = FIGURA VECHIE MODIFICATA "
50  PRINT AT(4,1); " H = FIGURA VECHIE MODIFICATA "
60  INPUT F$
70  J = 1
75  E = 1
80  I = 0
82  U = 0
84  V = 0
86  W = 0
90  GOSUB 5000
100 GOSUB 700
110 GOSUB 9000
120 GOSUB 300
130 GOSUB 3000
140 GOSUB 2000
150 MOVE C=30,D=50
160 GOSUB 9000
165 GOSUB 300
166 K$ = G$(1,JTOJ)
170 IF K$ = "A" THEN 140
180 IF K$ = "B" THEN 110
190 IF K$ = "S" THEN 250
200 IF K$ = "L" THEN 300
210 GOSUB 3000
220 GOSUB 2000
230 DRAW C=30,D=30
240 GOTO 140
250 PRINT AT(1,1); " R = RELUATI "
260 PRINT AT(2,1); " S = STOP "
270 INPUT F$
280 IF F$ = "R" THEN 20
290 STOP
300 IF F$ = "N" THEN 330
310 G = R(E)
320 GOTO 370
330 PRINT AT(1,1); "
340 PRINT AT(1,1); "
350 INPUT G
360 R(E) = G
370 E = E+1
380 GOTO 160
500 IF F$ (<) N THEN 610
510 PRINT AT(1,1); "
520 PRINT AT(1,1); "
530 A$ = INKEY$
540 IF A$ = " " THEN 530
550 IF A$ = "C" THEN 610
560 IF A$ = "E" THEN 600
570 IF A$ = "A" THEN 620
580 IF A$ = "B" THEN 640
590 GOTO 530
600 GOSUB 9040
610 RETURN
620 G$(1,JTOJ+2) = "AAA"
630 RETURN
    
```

```

640 F4 = "N"
650 RETURN
700 INITP
710 PRINT AT(27,26):"5*2"
720 PRINT AT(28,26):"6*2"
730 PRINT AT(29,26):"7*X"
740 PRINT AT(30,26):"8*X"
750 PRINT AT(31,26):"9*Y"
760 PRINT AT(32,26):"0*Y"
770 IF F4 <> "N" THEN GOTO
780 PRINT AT(22,26):"C=CON"
790 PRINT AT(24,26):"I=IML"
800 PRINT AT(25,26):"A=ASN"
810 PRINT AT(23,26):"D=BEZ"
820 RETURN
2000 REM * CALCULE
2010 J3 = L+L+M+N
2020 IF J3 = 0 THEN 2080
2030 I1 = -M
2040 I2 = L
2050 J1 = -L+M
2060 J2 = -M+N
2070 GOTO 2120
2080 I1 = 0
2090 I2 = -1
2100 J1 = -1
2110 J2 = 0
2120 N1 = L+L+M+M+M+N
2130 IF N1 = 0 THEN 2290
2140 N1 = SQR(N1)
2150 N2 = I1*2+I2*2
2160 N2 = SQR(N2)
2170 N3 = J1*2+J2*2+J3*2
2180 N3 = SQR(N3)
2190 N4 = L+U+M+V+N+U
2200 N5 = N1*2-N4
2210 IF N5 = 0 THEN 2260
2220 T = 3*N1*2/N5
2230 C = -1*(U+I1+V+I2)/N2
2240 B = T*(U+J1+V+J2+U+J3)/N3
2250 RETURN
2260 C = 0
2270 B = 0
2280 RETURN
2290 PRINT " ERDARE"
2300 STOP
3000 FOR K = 0 TO 3
3010 K5 = GS(I, J+K*10)+K1
3020 IF K5 = "5" THEN 3090
3030 IF K5 = "6" THEN 3110
3040 IF K5 = "7" THEN 3130
3050 IF K5 = "8" THEN 3150
3060 IF K5 = "9" THEN 3170
3070 IF K5 = "0" THEN 3190
3080 GOTO 3200
3090 U = U-G
3100 GOTO 3200
3110 U = U+G
3120 GOTO 3200
3130 U = U-G
3140 GOTO 3200
3150 U = U+G
3160 GOTO 3200
3170 V = V+G
3180 GOTO 3200
3190 V = V-G
3200 NEXT K
3210 RETURN
5000 IF F4 = "N" THEN 5030
5010 G = P(L)
5020 GOTO 5060
5030 PRINT AT(9,1):" MARETINA "
5040 INPUT U
5050 R(L) = G
5060 B = E+I
5070 PRINT AT(10,1):" COORDONATELE "
5080 INPUT L,N,M
5090 RETURN
9000 I = I+1
9010 IF F4 = "N" THEN 9040
9020 REM * LA U SI N EXISTA IN MEMORIA *
9030 GOTO 9070
9040 PRINT AT (1,1):"
9050 PRINT AT(1,1):"
9060 INPUT G(I, J*10)+2)
9070 IF I < 254 THEN 9130
9080 I = 0
9090 J = J+3
9100 IF J < 9 THEN 9130
9110 PRINT " MEMORIA PLENA "
9120 STOP
9130 RETURN
9140 END

```

Poziția punctului F o aflăm din formula lentilei și din faptul că F se află pe direcția de privire:

$$\vec{f} = t_r \cdot \vec{n}$$

$$\|\vec{n} - \vec{f}\| = \frac{3 \|\vec{n}\|}{3 + \|\vec{n}\|}$$

de unde rezultă:

$$\vec{f} = \left(1 + \frac{3}{3 + \|\vec{n}\|} \right) \cdot \vec{n}.$$

Imaginea punctului C (punctul A) se obține intersectând cele două raze de lumină:

$$\begin{cases} \vec{a} = t_a (\vec{f} - \vec{b}) + \vec{f} \\ \vec{a} = t_a (\vec{u} - \vec{c}) + \vec{u} \end{cases}$$

Sistemul este compatibil determinat dacă vectorii $\{\bar{f}-\bar{b}, \bar{n}-\bar{c}\}$ sînt liniari independenți. Acest fapt este echivalent cu $\bar{c}(\bar{b}-\bar{n}) \neq 0$.

Deoarece $(\bar{b}-\bar{n}) \perp \bar{n}$ și $(\bar{b}-\bar{n}) \perp \bar{f}$, avem :

$$\bar{a} = t(\bar{n}-\bar{c}) + \bar{n}; \text{ unde :}$$

$$t = \frac{3 \|\bar{n}\|}{\|\bar{n}\|^2 - \left(\frac{3}{\|\bar{n}\|} + 1\right) \bar{n} \cdot \bar{c}}$$

Folosind ecuațiile de mai sus, se creează imaginea obiectului care va trebui apoi proiectată pe planul retinei (fig. 13.8)

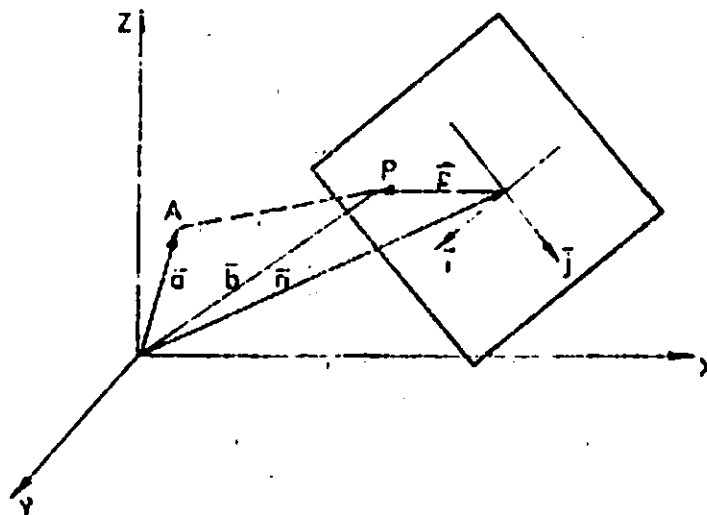


Fig. 13.8. Imaginea obiectului proiectată pe planul retinei.

Vom da în continuare instrucțiunile de exploatare ale programului. După lansarea sa în lucru, programul solicită operatorului una din opțiunile de bază :

- a) N — generarea unei figuri noi
- b) V — privirea figurii existente din diferite puncte de vedere
- c) M — modificarea figurii existente

Opțiunea N Se cere lungimea segmentului unitar, ce va fi utilizat în trasările ulterioare. Se recomandă valori cuprinse în domeniul 5—20. Aceste valori derivă din faptul că întreg ecranul este considerat ca un spațiu de 100 × 100 unități. O valoare mai mică pentru segmentul unitar permite trasarea de figuri mai complexe. O valoare mai mare permite trasarea mai expeditivă a figurilor, riscîndu-se însă depășirea limitelor ecranului. În acest caz se modifică punctul de vedere al privitorului, plasîndu-l la o distanță mai mare față de obiect.

Se solicită apoi coordonatele punctului de vedere (în care se consideră că se află privitorul). Generarea figurii pornește implicit din centrul ecranului, considerat ca avînd coordonatele 0, 0, 0. Prin comanda D (deplasare), se pot

modifică coordonatele de pornire a trasării. Aceeași comandă poate fi utilizată pentru părăsirea coordonatelor curente de trasare și începerea trasării dintr-un alt punct. Coordonatele indicate în comanda D se raportează totdeauna relativ la coordonatele ultimului punct de trasare anterior.

Trasarea efectivă a cîte unui segment unitar se execută cu ajutorul următoarelor taste :

- 5— pentru deplasare (trasare) pe axa OZ în sens pozitiv
- 6— pentru deplasare (trasare) pe axa OZ în sens negativ
- 7— pentru deplasare (trasare) pe axa OX în sens pozitiv
- 8— pentru deplasare (trasare) pe axa OX în sens negativ
- 9— pentru deplasare (trasare) pe axa OY în sens pozitiv
- 0— pentru deplasare (trasare) pe axa OY în sens negativ

La apăsarea tastei, se trasează un segment unitar în direcția specificată. În afară de trasare simplă (apăsarea unei singure taste de „direcție”) se permit și trasări combinate din maxim 3 direcții de deplasare. Combinația de direcții indică poziția punctului care va fi unit printr-o linie cu ultimul punct unde s-a oprit anterior trasarea. Astfel, dacă se indică o deplasare XYZ combinată, se va trasa direct diagonala cubului cu lungimea laturii egală cu segmentul unitar. De remarcat faptul că prin opțiunea de trasare combinată se pot face trasări în orice direcții, nu numai paralele cu cele 3 axe de coordonate.

Dacă se dorește modificarea lungimii segmentului unitar se utilizează comanda L. Trasarea va continua cu noua lungime.

Sfârșitul trasării figurii se indică prin comanda S.

Opțiunea V Permite schimbarea poziției privitorului față de obiectul generat. Se introduc coordonatele noului punct de vedere, apoi programul trasează fără întrerupere imaginea vechiului obiect, văzut din noul punct dat.

Opțiunea M Permite modificarea facilă a imaginii unui obiect deja generat. Trasarea figurii se face pas cu pas, iar după fiecare pas se așteaptă o comandă, care poate fi :

- C — se trece la pasul următor ;
- I — se înlocuiește comanda de trasare următoare cu ceea ce dorește operatorul ;
- A — se abandonează comanda de trasare următoare (salt peste un pas) ;
- D — dezvoltarea figurii — se consideră din acest moment trasarea ca și la opțiunea N.

La terminarea trasării, operatorul poate relua ciclul celor trei opțiuni.

Domeniile de aplicație pot fi multiple, începînd cu exersarea vederii în spațiu a diferitelor obiecte, și terminînd cu activități de design în arhitectură, interioare de locuințe, construcții de mașini etc.

13.11. Trasare de labirint

Se trasează un labirint de lățime și lungime dată. Generarea se face utilizând caracterele I, —, :, .. Astfel se poate genera labirintul fără utilizarea de instrucțiuni grafice. Labirintul are doar o singură intrare și ieșire și un singur drum de trecere. Fiind trasat pe baza generării de numere aleatoare se obțin forme diferite de labirint.

```

20 PRINT " LABIRINT "
30 DIM M(50,50),V(50,50)
40 PRINT " LĂȚIME , LUNGIME = "
50 INPUT M , V
60 IF M < 1 THEN 100
62 IF M < 50 THEN 100
64 IF V < 1 THEN 100
66 IF V < 50 THEN 100
68 PRINT " DIMENSIUNI ERONATE = "
69 GOTO 30
140 M = 0
150 V = 0
160 X = INT( RND(X) * M + 1 )
165 FOR I = 1 TO M
170 IF I = 1 THEN 172
171 PRINT " : "
172 GOTO 130
173 PRINT " "
180 NEXT I
190 PRINT " "
195 K = 1
196 M(R,1) = 0
197 L = C+1
200 R = X
205 S = 1
205 GOTO 210
210 IF R = M THEN 240
215 IF V(R,S) THEN 250
220 R = 1
225 S = 1
225 GOTO 230
230 R = 1
235 S = C+1
235 GOTO 230
240 R = R+1
250 IF M(R,C) = 0 THEN 210
260 IF R+1 = 0 THEN 530
265 IF M(R-1,S) < 0 THEN 530
270 IF S-1 = 0 THEN 530
280 IF M(R,S+1) < 0 THEN 530
290 IF R = M THEN 330
300 IF M(R+1,S) < 0 THEN 330
310 X = INT( RND(X) * S + 1 )
320 ON X GOTO 790 , 820 , 840
330 IF S < V THEN 340
334 IF Z = 1 THEN 370
338 Q = 1
339 GOTO 350
340 IF M(R,S+1) < 0 THEN 370
350 X = INT( RND(X) * S + 1 )
360 ON X GOTO 790 , 820 , 910
370 X = INT( RND(X) * 2 + 1 )
380 ON X GOTO 790 , 820
390 IF R = M THEN 470
400 IF M(R+1,S) < 0 THEN 470
405 IF S < V THEN 420
410 IF Z = 1 THEN 450
415 Q = 1
416 GOTO 430
420 IF M(R,S+1) < 0 THEN 450
430 X = INT( RND(X) * S + 1 )
440 ON X GOTO 790 , 840 , 910
450 X = INT( RND(X) * 2 + 1 )
460 ON X GOTO 790 , 860
470 IF S < V THEN 490
480 IF Z = 1 THEN 520
485 Q = 1
486 GOTO 500
490 IF M(R,S+1) < 0 THEN 520
500 Z = INT( RND(X) * 2 + 1 )
510 ON X GOTO 790 , 910
520 GOTO 790
530 IF S-1 = 0 THEN 670
540 IF M(R,S-1) < 0 THEN 670
545 IF R = M THEN 610
547 IF M(R+1,S) < 0 THEN 610
550 IF S < V THEN 560
552 IF Z = 1 THEN 590
554 Q = 1
556 GOTO 570
560 IF M(R,S+1) < 0 THEN 590
570 X = INT( RND(X) * S + 1 )
580 ON X GOTO 820 , 840 , 910
590 X = INT( RND(X) * 2 + 1 )
600 ON X GOTO 620 , 840
610 IF S < V THEN 630
620 IF Z = 1 THEN 660
625 Q = 1
627 GOTO 640
630 IF M(R,S+1) < 0 THEN 660
640 X = INT( RND(X) * 2 + 1 )
650 ON X GOTO 820 , 910
660 GOTO 820
670 IF R = M THEN 740
680 IF M(R+1,S) < 0 THEN 740
685 IF S < V THEN 700
690 IF Z = 1 THEN 730
695 Q = 1
697 GOTO 630
700 IF M(R,S+1) < 0 THEN 730
710 X = INT( RND(X) * 2 + 1 )
720 ON X GOTO 840 , 910
730 GOTO 860
740 IF S < V THEN 740
750 IF Z = 1 THEN 780
755 Q = 1
757 GOTO 770
760 IF M(R,S+1) < 0 THEN 780
770 GOTO 910
780 GOTO 1000
790 M(R-1,S) = C
800 C = C+1
802 V(R-1,S) = 2
804 R = R-1
810 IF C = H*V + 1 THEN 1010
815 Q = 0
816 GOTO 260
820 M(R,S-1) = C
830 C = C+1
840 V(R,S-1) = 1
842 S = S-1
844 IF C = H*V + 1 THEN 1010
850 Q = 0
854 GOTO 260
860 M(R+1,S) = C
870 C = C+1
872 IF V(R,S) = 0 THEN 880
875 V(R,S) = 3
877 GOTO 890
880 V(R,S) = 2
890 R = R+1
900 IF C = H*V + 1 THEN 1010
905 GOTO 530
910 IF Q = 1 THEN 940
920 M(R,S+1) = C
922 L = C+1
924 IF V(R,S) = 0 THEN 940
930 V(R,S) = 3
932 GOTO 950
940 V(R,S) = 1

```


13.12. Mastermind

Jocul este o variantă adaptată a celebrului MASTERMIND, care solicită determinarea unei combinații de 4 culori într-o ordine anumită, din 6 culori posibile. Adaptarea s-a făcut pentru determinarea unei combinații de 4 cifre din 6 posibile. Setul de cifre posibile este : ..
0, 1, 2, 3, 4, 5. Se introduce o combinație de 4 cifre ; programul compară această combinație cu propria sa combinație stabilită prin randomizare la începutul jocului. Se afișează un "*" pentru cifră corectă în poziție corectă, și un "+
pentru cifră corectă în poziție incorectă. Se admite ca aceeași cifră să apară de mai multe ori în cadrul combinației de 4 cifre. Scopul jocului este de a determina combinația corectă din cât mai puține încercări. Se afișează la sfârșit numărul total de încercări.

```

8  INITP
10  PRINT " M A S T E R M I N D "
15  PRINT " GASITI PATRU CIFRE DIN "
20  PRINT " SASB POSIBILE 0,1,2,3,4,5 "
25  PRINT " RASPUNSUL LA O INCERCARE "
30  PRINT " * = NR. BUN LA LOC BUN "
35  PRINT " + = NR. BUN LA ALT LOC "
40  L = 0
45  DIM A(4),B(4)
50  FOR Z = 1 TO 4
55  Y = INT(RND(X)*6)
60  B(Z) = Y
65  NEXT Z
70  L = 0
75  PRINT " DATE O INCERCARE "
80  L = L+1
85  MAT INPUT A
90  K = 0
95  J = 0
100 FOR Z = 1 TO 4
105 IF A(Z) <= 5 THEN 115
110 J = 1
115 NEXT Z
120 IF J = 0 THEN 135
125 PRINT " COMBINATIE ERONATA "
130 GOTO 85
135 FOR Z = 1 TO 4
140 B(Z) = ABS(B(Z))
145 IF A(Z) <> B(Z) THEN 165
150 K = K+1
155 A(Z) = 7
160 B(Z) = -B(Z)
165 NEXT Z
170 FOR Z = 1 TO 4
175 G = 0
180 FOR N = 1 TO 4
185 IF A(N) = 7 THEN 215
190 IF A(N) <> B(Z) THEN 215
195 IF G <> 0 THEN 215
200 J = J+1
205 A(N) = 7
210 G = 1
215 NEXT N
220 NEXT Z
230 IF K = 0 THEN 250
235 FOR Z = 1 TO K
240 PRINT "*" " "
245 NEXT Z
250 IF J = 0 THEN 270
255 FOR Z = 1 TO J
260 PRINT "+" " "
265 NEXT Z
270 PRINT
275 IF X < 4 THEN 80
280 PRINT " ATI GASIT DIN " ; L ;
285 PRINT " INCERCARI "
290 PRINT " BORITI ALT JOC (DA/NU) "
300 INPUT B$
305 IF B$ = "DA" THEN 30
310 STOP
320 END

```

13.13. Vânătoarea de vulpi

Programul trasează un careiaj de $N \times N$ dimensiuni ($N=10-30$) în care plasează în mod aleator 3 vulpi. Scopul jocului este de a determina coordonatele vulpilor din cât mai puține încercări. O încercare se specifică prin coordonatele punctului (X, Y), după care programul afișează un "/" pentru punct liber, sau un "X" pentru vulpe găsită. Se totalizează la sfârșit numărul de încercări.

```

5 DIM A(6),C(3)
10 INITP
20 N = INT(RND(X)*30)
30 IF N > 10 THEN 50
40 N = 10
50 FOR I = 1 TO 4
60 A(I) = INT(RND(X)*30)+10
70 IF A(I) < N THEN 100
80 A(I) = A(I)-5
90 GOTO 70
100 N = 0
110 IF I < 2 THEN 170
120 FOR J = 1 TO I-1 STEP 2
130 IF A(I) <> A(J) THEN 150
140 N = 1
150 NEXT J
160 IF N <> 0 THEN 60
170 NEXT I
180 NAT C = 2BR
190 N = 0
200 U = 0
205 V = 0
210 INITP
220 WINDOW -10,N+10,-10,N+10
230 PRINT AT(1,1); " VANA TOARE DE VULPI "
240 PRINT AT(2,1); " GASITI TREI VULPI ";
245 PRINT "INTR-O ZONA DE ";N;"/";N
250 FOR I = 0 TO N
260 MOVE 0,I
270 DRAW N,I
280 MOVE I,0
290 DRAW I,N
300 NEXT I
310 PRINT AT(20,1); "
320 PRINT AT(20,1); " COORDONATA ";
330 NAT INPUT B(2)
340 V = V+1
350 FOR J = 1 TO 5 STEP 2
360 IF A(J) <> B(1) THEN 330
370 IF A(J+1) <> B(2) THEN 330
380 I = INT(J/2)+1
390 IF C(I) = 0 THEN 420
400 PRINT AT(20,1); " VULPE DEJA GASITA "
410 GOTO 520
420 PRINT AT(20,1); " ATI GASIT O VULPE "
430 C(I) = 1
440 MOVE B(1),B(2)
450 DRAW B(1)+1,B(2)+1
460 MOVE B(1),B(2)+1
470 DRAW B(1)+1,B(2)
480 V = V+1
490 IF V <> 3 THEN 520
500 PRINT " ATI GASIT O(N ";V;
505 PRINT " INCERCARI "
510 N = 1
520 J = 3
530 NEXT J
540 IF N <> 0 THEN 20
550 MOVE B(1),B(2)
560 DRAW B(1)+1,B(2)+1
570 GOTO 310
580 END

```

13.14. Verificarea vitezei de reacție

Programul afișează într-o poziție aleatoare pe ecran o cifră (0—9) și așteaptă un timp determinat introducerea aceleiași cifre de către operator.

Se solicită inițial viteza de joc (1—5), viteza 1 fiind cea mai mare. Se afișează permanent scorul pe ecran. Din start, jucătorul primește 5 puncte. Pe parcurs, modificarea scorului decurge astfel:

- pentru o cifră introdusă corect, se adaugă un punct;
- pentru o cifră incorectă, se scade un punct;
- pentru nici o cifră introdusă în intervalul de așteptare, se scad 2 puncte.

Oprirea programului are loc la atingerea scorului nul.

```

5  INITP
10 PRINT "R E P L E X U L "
15 PRINT " SE AFISEAZA ALEATOR O CIFRA "
20 PRINT " PE ECRAN "
25 PRINT " DATETI CIT NAI REPEDE TASTA "
30 PRINT " RESPECTIVA "
35 PRINT " DIN START PRIMITI 5 PUNCTE "
40 PRINT " +1 PUNCT LA TASTA BUNA "
42 PRINT " -1 PUNCT LA TASTA BRONATA "
44 PRINT " -2 PUNCTE DACA NU TASTATI "
45 L = 5
50 PRINT " DATI VITEZA DE JOC (1,2,3,4,5) "
55 INPUT M
60 INITP
65 PRINT AT(1,1); " R E P L E X U L "
70 PRINT AT(2,1); " SCORUL: ";L
75 IF L < 1 THEN 220
80 A = INT(RND(X)*9)+40
85 B$ = INKEY$
90 X = INT(RND(X)*26)+4
95 Y = INT(RND(X)*29)+1
100 PRINT AT(X,T);CHR$(A)
105 A$ = INKEY$
110 FOR I = 1 TO 100*M
115 IF A$ <> B$ THEN 125
120 B$ = INKEY$
125 NEXT I
130 IF B$ = A$ THEN 195
135 B = 0
140 FOR I = 0 TO 9
145 IF B$ = STR$(I) THEN 155
150 GOTO 165
155 IF I <> A-40 THEN 165
160 B = I
165 NEXT I
170 IF B = 1 THEN 105
175 L = L-1
180 GOTO 200
185 L = L+1
190 GOTO 200
195 L = L-2
200 FOR I = 1 TO N=100
205 REN " PAUSE "
210 NEXT I
215 GOTO 70
220 STOP
225 END

```

13.15. Perspico

Jocul constă în alinierea a trei 0-uri în linie sau diagonală, în cadrul unui careu de 3×3. Fiecare căsuță din careu primește un număr (1—9), utilizat pentru indicarea locului în care se plasează 0-ul jucătorului. Ca răspuns, programul plasează un X într-o poziție defavorabilă pentru jucător, căutând la rândul său să alinieze trei X-uri. La sfârșit se afișează câștigătorul.

```

5 PRINT " P E R S P I C O "
10 DIM B(9),P(9)
15 INITP
20 FOR A = 1 TO 9
25 B(A) = A
30 NEXT A
35 NAT P = ZBR
40 B = 0
45 B = 0
50 N = 0
55 X = 4
60 PRINT AT(X,3);
65 FOR A = 1 TO 9
70 IF A = X THEN 90
75 IF B(A) = 0 THEN 110
80 IF B(A) = 10 THEN 120
85 GOTO 130
90 X = X+3
95 PRINT
100 PRINT AT(X,3);
105 GOTO 75
110 PRINT "0"; " ";
115 GOTO 135
120 PRINT "X"; " ";
125 GOTO 135
130 PRINT B(A); " ";
135 NEXT A
140 IF N = 1 THEN 210
145 IF B (<) 0 THEN 160
150 PRINT AT(15,1); " AN CISTIGAT "
155 GOTO 445
160 PRINT AT(15,1); " MUTAREA "
165 INPUT I
170 IF B(I) (<) 2 THEN 165
175 N = 1
180 B = B+1
185 B(I) = 0
190 P(I) = 1
195 IF 0 (= 0 THEN 55
200 PRINT AT(15,1); " RENIZA "
205 GOTO 475
210 C = 0
215 C = 1
220 B = 7
225 P = 7
230 GOSUB 340
235 C = 2
240 B = 8
245 GOSUB 340
250 C = 3
255 B = 9
260 GOSUB 340
265 C = 1
270 F = 4
275 GOSUB 340
280 B = 3
285 P = 1
290 GOSUB 340
295 C = 4
300 B = 6
305 GOSUB 340
310 C = 7
315 B = 9
320 GOSUB 340
325 C = 3
330 B = 7
335 P = 2
340 GOSUB 340
345 C = 8+1
350 IF 0 = 5 THEN 50
355 GOTO 215
360 B = 0
365 U = 1
370 FOR A = C TO B STEP P
375 B = B+P(A)
380 NEXT A
385 IF B = 3 THEN 440
390 IF B = 0 THEN 435
395 IF B = 6 THEN 470
400 IF B = 1 THEN 435
405 IF B = 2 THEN 470
410 IF B = 2 THEN 435
415 IF B = 5 THEN 470
420 IF B = 3 THEN 435
425 IF B = 1 THEN 470
430 IF B = 4 THEN 470
435 RETURN
440 PRINT AT(15,1); " ATI CISTIGAT "
445 PRINT AT(17,1); " SCRITI ALT JOC (BA=1) "
450 INPUT B
455 IF B = 1 THEN 15
460 STOP
470 FOR A = C TO B STEP P
475 IF B(A) (<) A THEN 500
480 IF U = 0 THEN 500
485 B(A) = 10
490 P(A) = 4
495 U = 0
500 NEXT A
505 GOTO 50
510 END

```

13.16. Cursa de obstacole

Se conduce un mobil printr-o configurație de obstacole generată aleator pe ecran. Deplasarea mobilului are loc de la dreapta la stînga ; jucătorul are posibilitatea de a muta cu un rînd mai sus sau mai jos direcția de înaintare, apăsînd tasta S, respectiv J. Dacă nu se apasă nici o tastă, mobilul își continuă deplasarea în linie dreaptă. La lovirea unui obstacol, se oprește înaintarea și se afișează scorul. Valoarea scorului indică distanța parcursă pînă la lovirea obstacolului. După parcurgerea unui ecran complet, se reia mișcarea mobilului, modificînd locul de apariție al acestuia și combinația de obstacole,

```

10 PRINT " CONDUCTI VEHICOLUL PRINTRU "
15 PRINT "OBSTACOLE "
20 PRINT " VEHICOLUL SE DIRIGEAZA CU "
25 PRINT "TASTELE "
30 PRINT " S = S U S , J = J O S "
40 DIM B(30,30)
50 S = 0
60 MAY B = ZER
70 INITP
80 FOR A = 1 TO 30
90 X = INT(RND(X)*29)+1
100 Y = INT(RND(X)*29)+1
110 PRINT AT(X,Y); " "
120 B(X,Y) = 1
130 NEXT A
140 P = INT(RND(X)*29)+1
150 FOR A = 30 TO 1 STEP -1
160 PRINT AT(P,A); "<"
170 IF B(P,A) = 1 THEN 310
180 PRINT AT(P,A); " "
190 Z = INKEY$
200 IF Z = " " THEN 200
210 IF Z = "J" THEN 240
220 IF Z <> "S" THEN 280
230 IF P = 1 THEN 280
240 P = P-1
250 GOTO 280
260 IF P = 30 THEN 280
270 P = P+1
280 S = S+1
290 NEXT A
300 GOTO 40
310 FOR Z = 1 TO 20
320 PRINT AT(P,A); "S"
330 PRINT AT(P,A); " "
340 NEXT Z
350 PRINT AT(29,1); " SCORUL " : S
360 FOR I = 1 TO 100
370 REM " TIME "
380 NEXT I
390 GOTO 30
400 END

```

13.17. Tragerea la țintă

Pe ecran se afișează o țintă, considerată ca fiind în fața privitorului. Se introduce viteza inițială a proiectilului și unghiul înălțătorului. Programul consideră ținta ca fiind la o distanță aleatoare față de trăgător. Din datele introduse, pe baza formulei:

$$S = V_0^2 \sin \frac{\omega}{g} \quad (1)$$

unde:

S este distanța parcursă de proiectil;

V_0 este viteza inițială;

ω este unghiul înălțătorului;

g este accelerația gravitațională.

Se determină dacă s-a atins sau nu ținta.

Dacă ținta nu a fost atinsă, se afișează distanța la care a căzut proiectilul față de țintă („+” pentru lovitură prea lungă, „-” pentru lovitură prea scurtă). Se caută lovirea ținte din cît mai puține încercări, pe baza coroborării datelor făcute pentru loviturile în gol.

```

10 INITP
20 PRINT AT(1,10); " T I N T A "
30 PRINT AT(3,13); " "
40 PRINT AT(4,10); " "
50 PRINT AT(5,10); " "
60 PRINT AT(6,10); " "
70 PRINT AT(8,1); " VITEZA UNGHIIUL "
75 PRINT " DISTANTA "
80 PRINT AT(9,L); "-----";
85 PRINT "-----";
90 K = 0
100 B = 1
110 X = INT(RND(X)*13000+30)
120 K = K+1
122 IF K <= 32 THEN 130
125 K = 32
130 PRINT AT(15,22); " " ; B ; " < "
140 PRINT AT(10+K,1); " " ; " " ;
150 INPUT V
160 IF V < 10 THEN 140
170 PRINT AT(10+K,10); " " ;
180 INPUT B
190 IF B < 3 THEN 170
200 IF B > 89 THEN 170
210 Z = INT(V*2*SIN(P/180/90)/9.81)
220 B = 2-X
230 PRINT AT(10+K,20); B
240 IF Z = X THEN 270
250 B = B+1
260 GOTO 120
270 PRINT AT(13,1); " MINERIT "
280 PRINT AT(15,1); " BORITI ALT JOC (BA+1) "
290 INPUT B
300 IF B = 1 THEN 10
310 STOP
320 END

```

13.18. Ecranul magic

Programul permite trasarea oricăror figuri compuse din drepte orizontale și verticale, pornind dintr-un punct dat inițial. Deplasarea punctului este coordonată de tastele :

1— dreapta	3— jos
2— stînga	4— sus

Punctul se deplasează continuu, oprirea trasării făcîndu-se prin apăsarea oricărei taste în afara celor de mai sus. Reluarea trasării se face prin apăsarea oricărei taste de comandă a direcției de deplasare.

Direcția de deplasare poate fi completată, pentru a avea posibilitatea trasării sub un unghi oarecare față de orizontală. Dacă se dorește o trasare la un unghi de 45° în direcția dreapta-sus, se introduce linia 105 de recunoaștere a tastei "5" :

```
105 IF A$ = "5" THEN 500
```

La linia 500 se introduce secvența de trasare :

```
500 REM "DEPLASARE 45 DREAPTA SUS"
```

```
510 B$=INKEY$
```

```
520 IF Y=100 THEN 570
```

```
530 IF X=100 THEN 570
```

```
540 X=X+1
```

```
550 Y=Y+1
```

```
560 DRAW X, Y
```

```

570 A $ = INKEYS
580 IF A $ = B $ THEN 520
590 GOTO 60

```

Programul poate fi generalizat pentru trasări pe alte direcții, trasări de linii curbe etc.

```

5 INITP
10 PRINT AT(1,1); " ECRAN BASIC "
15 PRINT AT(2,1); " 1= DREAPTA "
20 PRINT AT(3,1); " 2= STINGA "
25 PRINT AT(4,1); " 3= JOS "
30 PRINT AT(5,1); " 4= SUS "
35 PRINT AT(4,1); " COORD. DE ÎNCEPUT "
40 INPUT X,Y
45 MOVE X,Y
50 A$ = INKEYS
60 IF A$ = " " THEN 50
70 IF A$ = "1" THEN 120
80 IF A$ = "2" THEN 200
90 IF A$ = "3" THEN 280
100 IF A$ = "4" THEN 360
110 GOTO 50
120 REM " DEPL. DREAPTA "
130 B$ = INKEYS
140 IF X = 100 THEN 170
150 X = X+1
160 DRAW X,Y
170 A$ = INKEYS
180 IF A$ = B$ THEN 140
190 GOTO 50
200 REM " DEPL. STINGA "
210 B$ = INKEYS
220 IF X = 0 THEN 250
230 X = X-1
240 DRAW X,Y
250 A$ = INKEYS
260 IF A$ = B$ THEN 220
270 GOTO 50
280 REM " DEPL. JOS "
290 B$ = INKEYS
300 IF Y = 0 THEN 330
310 Y = Y-1
320 DRAW X,Y
330 A$ = INKEYS
340 IF A$ = B$ THEN 300
350 GOTO 50
360 REM " DEPL. SUS "
370 B$ = INKEYS
380 IF Y = 100 THEN 410
390 Y = Y+1
400 DRAW X,Y
410 A$ = INKEYS
420 IF A$ = B$ THEN 360
430 GOTO 50
490 END

```

13.19. Nim

Jocul Nim — originar din China antică — este cunoscut și sub numele de Fan-Tan.

Se generează într-un număr oarecare de grămezi un număr aleator de obiecte. Jucătorii (în cazul de față operatorul și calculatorul) elimină cu schimbul obiecte din grămezi, și anume cel puțin un obiect (cel mult toată grămadă), dar la un moment dat numai dintr-o singură grămadă.

Câștigă cine scoate ultimul sau ultimele obiecte din ultima grămadă rămasă.

O grămadă de obiecte se reprezintă printr-o linie de asteriscuri (un asterisc reprezintă un obiect). Numărul total de grămezi se stabilește de operator la începutul fiecărui joc. Numărul obiectelor din fiecare grămadă se generează aleator la începutul jocului.

```

10 DIM A(10),B(10,5),O(5),V(10)
15 PRINT " N = I - M "
20 PRINT " CU CITE GRAMEZI JUCATI? "
30 INPUT N
40 IF N <= 0 THEN 20
50 IF N > 9 THEN 20
55 MAT V = ZER
60 REM " DEJERAREA GRAMEZILOR "
70 FOR I = 1 TO N
80 A(I) = INT( RND(1) * 30 )
85 IF A(I) = 0 THEN 80
90 NEXT I
95 GOSUB 2050
100 PRINT " DIN CARE GRAMADA SI CITE "
105 PRINT " OBIECTE LUATI? "
110 INPUT I , K
120 IF I > N THEN 230
130 IF I < 0 THEN 230
140 IF K > A(I) THEN 230
150 IF K < 1 THEN 230
160 V(I) = K
170 GOSUB 450
180 REM " JOACA CALCULATORUL "
190 GOSUB 1300
200 GOSUB 450
210 IF C <> 0 THEN 290
220 GOTO 95
230 PRINT " ERORARE "
240 GOTO 95
290 IF C = -1 THEN 320
300 PRINT " AN CISTIGAT "
310 STOP
320 PRINT " AI CISTIGAT "
330 STOP
450 FOR I = 1 TO N
460 A(I) = A(I) - V(I)
470 V(I) = 0
480 NEXT I
490 RETURN
1300 MAT B = ZER
1350 MAT O = ZER
1360 FOR U = 1 TO N
1370 M = A(U)
1380 FOR I = 1 TO 5
1390 B(U,I) = M - INT( M/2 ) + 2
1400 M = INT( M/2 )
1410 O(I) = B(U,I) + O(I)
1420 IF O(I) > 1 THEN 1460
1430 GOTO 1470
1440 O(I) = 0
1450 NEXT I
1470 NEXT U
1480 U = 0
1490 FOR U = 5 TO 1 STEP -1
1495 IF O(U) <> 0 THEN 1510
1500 GOTO 1530
1510 M = U
1520 U = 1
1530 NEXT U
1540 IF M <> 0 THEN 1570
1550 GOSUB 1600
1560 RETURN
1570 GOSUB 1730
1580 RETURN
1600 U = 0
1610 FOR J = N TO 1 STEP -1
1620 IF A(J) <> 0 THEN 1640
1630 GOTO 1660
1640 M = J
1650 J = 1
1660 NEXT J
1670 IF M <> 0 THEN 1700
1680 C = -1
1690 RETURN
1700 C = 0
1705 J = M
1710 V(J) = A(J) - INT((A(J)-1)*RND(1))
1720 RETURN
1730 U = 0
1740 FOR I = 1 TO N
1745 IF B(I,U) <> 1 THEN 1760
1750 B(I,U) = 0
1752 U = 1
1755 I = N
1760 NEXT I
1770 FOR J = M-1 TO 1 STEP -1
1775 IF O(J) = 0 THEN 1790
1780 B(U,J) = 1 - B(U,J)
1790 NEXT J
1795 I = U
1800 FOR J = 5 TO 1 STEP -1
1810 V(I) = 2 * V(I) + B(I,J)
1820 NEXT J
1830 V(I) = A(I) - V(I)
1840 S = 0
1850 FOR J = 1 TO N
1860 S = S + A(J) - V(J)
1870 NEXT J
1880 IF S = 0 THEN 1910
1890 C = 0
1900 RETURN
1910 C = 1
1920 RETURN
2050 FOR I = 1 TO N
2060 IF A(I) = 0 THEN 2100
2070 FOR J = 1 TO A(I)
2080 PRINT "*"
2090 NEXT J
2100 PRINT
2110 V(I) = 0
2120 NEXT I
2130 RETURN

```


13.20. Turnurile din Hanoi

Jocul este cunoscut și sub numele *Acele Faraonului* și constă în a transfera un set de discuri aranjate în ordine descrescătoare a mărimii lor de pe primul ac pe al treilea ac, utilizând ca element ajutător acul din mijloc. Regula

```

100 REM " TURURILE DIN HANOI "
110 DIM T(10,3)
120 E = 0
130 NAT Y = ZER
140 PRINT " TURURILE DIN HANOI "
150 PRINT " DISCURILE SE MUTA DIN STINGA "
155 PRINT " IN DREAPTA SI NU SE PUNE DISC "
160 PRINT " MAI MARE PE DISC MAI MIC "
200 PRINT " CU CITE DISCURI JUCATI? (MAX. 10) "
210 INPUT S
230 M = 0
240 IF S > 0 THEN 260
250 GOTO 270
260 IF S <= 10 THEN 350
270 E = E + 1
280 IF E > 2 THEN 310
290 PRINT " DATI NUMAR CORECT 1 - 10 "
300 GOTO 200
310 PRINT " NU JUCATI CORECT "
320 STOP
350 PRINT " IN PROGRAM DISCURILE SINT BOTEZATE "
360 PRINT " PRIN NUMERE ZECIMALE 1 LA 10 "
370 PRINT " DACA JUCATI CU MAI PUTIN DECIT 10 "
380 PRINT " VETI JUCA CU DISCURI DE NUMERE CELE "
390 PRINT " MAI MARI "
380 PRINT " MUTATI DE PE ACUL 1 PE ACUL 3 "
400 Y = 10
410 D = 10
420 FOR X = S TO 1 STEP -1
430 T(Y,1) = D
440 D = D - 1
450 Y = Y - 1
460 NEXT X
470 GOSUB 1240
480 PRINT AT(27,1): " CARE DISC IL MUTATI? "
490 E = 0
500 INPUT D
510 IF D > 10 - S THEN 530
520 GOTO 540
530 IF D <= 10 THEN 590
540 PRINT " DISC INEXISTENT "
550 E = E + 1
560 IF E > 1 THEN 310
570 GOTO 480
590 REM " VERIFICA DACA DISCUL E SUB ALT DISC "
595 A = 10
600 B = 3
595 FOR R = 1 TO 10
600 FOR C = 1 TO 3
605 IF T(R,C) <> D THEN 630
610 A = R
615 B = C
620 R = 10
625 C = 3
630 NEXT C
635 NEXT R
640 IF T(A-1,B) <> 0 THEN 670
645 GOTO 700
670 PRINT " DISCUL E SUB ALT DISC "
680 GOTO 480
700 E = 0
710 PRINT " PE CARE AC IL MUTATI? "
720 INPUT N
730 IF (N-1)*(N-2)*(N-3) = 0 THEN 800
740 E = E + 1
750 IF E > 1 THEN 310
760 PRINT " DATI NUMAR CORECT 1 . 2 . 3 "
770 GOTO 710
800 A = 0
810 FOR R = 1 TO 10

```

```

010 IF T(R,N) = 0 THEN 020
015 A = R
020 R = 10
025 NEXT R
030 IF A = 0 THEN 070
035 REM " VERIFICA SA NU SE PUNA DISC MAI "
036 REM " MARE PE MAI MIC "
040 IF D < T(A,N) THEN 070
050 PRINT " DISC MAI MARE PE MAI MIC NU SE POATE "
060 GOTO 400
070 FOR V = 1 TO 10
075 FOR M = 1 TO 3
080 IF T(V,M) = D THEN 090
085 GOTO 910
090 A = V
095 B = M
100 U = 10
105 N = 3
110 NEXT M
115 NEXT V
120 FOR U = 1 TO 10
125 IF T(U,N) = 0 THEN 130
130 X = U-1
135 U = 10
140 GOTO 160
150 X = 10
160 NEXT U
170 T(X,N) = T(A,B)
180 T(A,B) = 0
190 REM " TIPARESTE STAREA ACTUALA "
1000 GOSUB 1240
1010 M = M+1
1015 A = 0
1020 FOR R = 1 TO 10
1030 FOR C = 1 TO 2
1040 IF T(R,C) = 0 THEN 1060
1050 A = 1
1060 NEXT C
1070 NEXT R
1080 IF A = 0 THEN 1120
1090 IF M <= 2*B THEN 400
1100 PRINT " ATI DEPASIT NUMARUL DE PASI MINIM "
1105 PRINT " NECESARI "
1110 STOP
1120 PRINT " ATI REZOLVAT IN "M" PASI "
1140 PRINT " MAI JUCATI? (DA/NU) "
1150 INPUT AS
1160 IF AS = "DA" THEN 120
1170 STOP
1240 INITP
1250 A = 26
1260 FOR I = 10 TO 1 STEP -1
1270 Z = 5
1280 FOR J = 1 TO 3
1290 IF T(I,J) = 0 THEN 1340
1300 PRINT AT(A,Z-INT(T(K,J)/2)+1);
1310 FOR V = 1 TO T(K,J)
1320 PRINT "N";
1330 NEXT V
1340 Z = Z+10
1350 NEXT J
1360 A = A-3
1375 PRINT
1390 RETURN

```

de bază a transferului specifică obligativitatea mutării unui disc mai mic peste unul mai mare, niciodată invers.

În jocul realizat pe calculator se permit maxim 10 discuri. Discurile sînt reprezentate prin asteriscuri. Numărul de asteriscuri care formează discul reprezintă mărimea discului. Pașii necesari pentru a termina jocul sînt $2^n - 1$, unde n reprezintă numărul de discuri în joc. Dacă jucătorul nu termină în număr de pași necesari, se oprește jocul.

13.21. Jocul cu trei grămezi

Se generează trei grămezi cu diferite numere de obiecte în fiecare grămadă. Jucătorul mută dintr-o grămadă oarecare într-o altă grămadă un obiect în contul unui obiect din a treia grămadă care se elimină din joc. Inițial cel puțin

```

5 INITP
10 PRINT "JOCUL CU TREI GRAMEZI "
12 PRINT "-----"
15 PRINT "SE DAU TREI GRAMEZI PRIN "
20 PRINT "MARILE DE OBIECTE DIN ELE "
25 PRINT "JOCUL CONSTA IN A MUTA UN OBIECT "
30 PRINT "DINTR-O GRAMADA IN ALTA IN "
35 PRINT "CONTUL UNUI OBIECT DIN A TREIA "
40 PRINT "CARE SE ELIMINA DIN JOC "
45 PRINT "SCOPUL ESTE DE A AJUNGE LA DOUA "
50 PRINT "GRAMEZI CALE SI IN A TREIA "
55 PRINT "SA RAMINA UN SINGUR OBIECT "
60 PRINT "GRAMEZILE SE INDICA PRIN 1 , 2 , 3 "
65 DIM T(3)
70 T(1) = INT (RND(1)) * 19 + 1
75 T(2) = INT (RND(1)) * 19 + 1
100 T(3) = INT (RND(1)) * 19 + 1
105 N = INT (RND(2)) * 2 + 1
110 M = INT (RND(2)) * 2 + 1
115 IF N = M THEN 110
120 P = 1
125 IF P = N THEN 140
130 IF P = M THEN 140
135 GOTO 150
140 P = P + 1
145 GOTO 125
150 IF INT(T(N)/2) = T(N)/2 THEN 165
155 IF INT(T(M)/2) = T(M)/2 THEN 190
160 GOTO 205
165 IF INT(T(N)/2) = T(N)/2 THEN 175
170 GOTO 205
175 IF INT(T(P)/2) <> T(P)/2 THEN 205
180 T(P) = INT (RND(2)) * 19 + 1
185 GOTO 175
190 IF INT(T(P)/2) = T(P)/2 THEN 205
195 T(P) = INT (RND(2)) * 19 + 1
200 GOTO 190
205 PRINT " INCEPEM UN JOC (DA = 1) "
210 INPUT D
215 IF D = 1 THEN 225
220 GOTO 210
225 INITP
230 I = 1
235 PRINT AT(1,5):T(1); " " : T(2); " " : T(3)
240 I = I + 1
245 IF I < 30 THEN 255
250 GOTO 225
255 PRINT AT(31,21) "
260 PRINT AT(30,21) " DIN CARE = IN CARE "
265 PRINT AT(31,21)
270 INPUT D,E
275 IF D = E THEN 295
285 IF D > 8 THEN 295
290 IF E > 3 THEN 295
295 G = 0
305 FOR J = 1 TO 3
310 IF D = J THEN 310
315 IF E <> J THEN 310
320 T(J) = T(J) + 1
325 GOTO 315
330 T(G) = T(J) - 1
335 IF T(J) = -1 THEN 390
345 NEXT J
350 IF G = 1 THEN 410
355 K = 0
360 L = 0
365 FOR J = 1 TO 3
370 IF T(J) <> 0 THEN 380
385 K = K + 1
390 GOTO 355

```

359 I - 1133
 359 NEKI J
 369 IF I C 2 TIEN 235
 369 IF I = 1 TIEN 300
 370 FRONT AT (30, 131) NU ATI AJUNS LA REZULTAT BUN -
 375 GOTO 90
 380 FRONT AT (30, 131) ATI AJUNS LA REZULTAT BUN -
 385 GOTO 90
 390 END

În două grămezi trebuie să existe obiecte. Scopul este ca în final să rămână un singur obiect într-o grămadă, celelalte să fie goale.

Se poate demonstra că dacă în toate trei grămezile există inițial obiecte în număr par sau impar (deci sînt de aceeași paritate) atunci jocul nu are soluție. Dacă la două grămezi paritatea este aceeași și a treia are altă paritate, jocul are soluție și în final acele grămezi vor fi goale care au avut aceeași paritate, rămînînd un obiect în grămada cu paritate diferită.

13.22. Ruleta

Un joc clasic de noroc este ruleta. Simularea pe calculator scoate în evidență doar faptul că jucătorul cu suma de bani inițială să rămână cît mai mult în joc, nu latura de noroc.

În cazul de față se generează aleator numărul și culoarea cîștigătoare. Jucătorul poate miza prin numere între 1 și 50 cu 5 pînă la 500 de unități de bani cu mai multe mize. Inițial jucătorul are la dispoziție 1 000 de unități de bani.

Mizele sînt după cum urmează .

— pe un număr se poate miza cu numărul respectiv (1 pînă la 36), pe 0 sau 00 se mizează cu numerele 49 respectiv 50. Cîștigul este de 35 de ori miza.

— pe numere între 1 și 12 ; 13 și 24 ; 25 și 36 se mizează cu codurile 37 ; 38 respectiv 39, cîștigul fiind dublu.

— pe numere în prima coloană (1, 4, 7, 10...) în a doua coloană (2, 5, 8, 11...) și în a treia coloană (3, 6, 9, 12...) se poate miza cu codurile 40 ; 41 respectiv 42, cîștigul fiind tot dublu.

— pe numere între 1 și 18 ; 19 și 36 se mizează cu codurile 43 respectiv 44 cîștigul fiind doar miza

— pe număr par sau impar se mizează cu codurile 45 respectiv 46 ; pe roșu sau negru cu codurile 47 respectiv 48, cîștigul fiind simplu.

Pentru fiecare miză făcută de jucător trebuie să se indice două numere :

— prima reprezintă codul (număr între 1 și 50)

— a doua reprezintă valoarea mizei).

```

10 DIM B(100),C(100),T(100),X(25),A(50)
20 MAT X = ZER
30 P = 1000
40 D = 100000
50 PRINT " CITE MIZE FACETI? "
60 INPUT Y
70 IF Y < 1 THEN 50
80 IF Y <> INT(Y) THEN 50
90 IF Y > 50 THEN 50
90 MAT A = ZER
100 FOR C = 1 TO Y
110 PRINT " M I Z A "C:
120 INPUT X,Z
130 B(C) = Z
140 T(C) = X
150 IF X < 1 THEN 110
IF X>38 THEN 110
170 IF X <> INT(X) THEN 110
180 IF Z < 5 THEN 110
190 IF Z <> INT(Z) THEN 110
210 IF Z > 500 THEN 110
220 IF A(X) = 0 THEN 230
230 PRINT " ACEST COD A MAI FOST "
240 GOTO 110
250 A(X) = 1
260 NEXT C
270 PRINT
275 PRINT "##### SE INVITIE POATA #####"
276 PRINT
280 S = 1 + INT(38*RND(X))
290 X(S) = X(S)+1
300 IF S < 37 THEN 340
310 IF S = 37 THEN 340
320 PRINT " CISTIOA 00 "
330 GOTO 490
340 PRINT " CISTIOA 0 "
350 GOTO 490
360 RESTORE
370 K = 0
380 FOR I = 1 TO 19
390 READ R
400 IF R <> S THEN 420
410 K = 1
420 NEXT I
430 IF K = 1 THEN 470
440 AS = "NEORU"
450 PRINT " CISTIOA "S: " , "I:AS
460 GOTO 490
470 AS = " ALB "
480 GOTO 450
490 PRINT
500 FOR C = 1 TO Y
510 IF T(C) < 37 THEN 1400
520 ON T(C)-36 GOTO 540,460,710,760,840
525 ON T(C)-41 GOTO 920,1000,1070,1110
530 ON T(C)-45 GOTO 1150,1190,1290
540 GOTO 1400
560 REM " 1 - 12 (COD 37) 2:1 "
570 IF S <= 12 THEN 820
580 PRINT " PIERDERE "B(C):" BANI CU MIZA "
590 D = D + B(C)
600 P = P - B(C)
610 GOTO 1540
620 PRINT " CISTIO "B(C)*2:" BANI CU MIZA "
630 D = D - B(C)*2
640 P = P + B(C)*2
650 GOTO 1540
660 REM " 13 -24 (COD 38) 2:1 "
670 IF S >12 THEN 490
680 GOTO 580
490 IF S < 25 THEN 620
700 GOTO 580
710 REM " 25 - 34 (COD 39) 2:1 "
720 IF S > 24 THEN 740
730 GOTO 580
740 IF S < 37 THEN 620
750 GOTO 580
760 REM " PRIMA COLOANA (COD 40) 2:1 "
770 K = 0
780 FOR I = 1 TO 34 STEP 3
790 IF S <> I THEN 810
800 K = 1
810 NEXT I
820 IF K = 1 THEN 620
830 GOTO 580
840 REM " COLOANA A DOUA (COD 41) 2:1 "
850 K = 0
860 FOR I = 2 TO 35 STEP 3
870 IF S <> I THEN 890
880 K = 1
890 NEXT I
900 IF K = 1 THEN 620
910 GOTO 580
920 REM " COLOANA A TREIA (COD 42) 2:1 "
930 K = 0
940 FOR I = 2 TO 36 STEP 3
950 IF S <> I THEN 970
960 K = 1
970 NEXT I
980 IF K = 1 THEN 620
990 GOTO 580
1000 REM " 1 - 19 (COD 43) 1:1 "
1010 IF S < 19 THEN 1030
1020 GOTO 580
1030 PRINT " CISTIO "B(C):" BANI CU MIZA "C
1040 D = D - B(C)
1050 P = P + B(C)
1060 GOTO 1540
1070 REM " 19 - 34 (COD 44) 1:1 "
1080 IF S < 19 THEN 580
1090 IF S < 37 THEN 1030
1100 GOTO 580
1110 REM " NR. PAR (COD 44) 1:1 "
1120 IF S/2 <> INT(S/2) THEN 580
1130 IF S < 37 THEN 1030
1140 GOTO 580
1150 REM " NR. IMPAR (COD 45) 1:1 "
1160 IF S/2 = INT(S/2) THEN 580
1170 IF S < 37 THEN 1030
1180 GOTO 580
1190 REM " ROSU (COD 47) 1:1 "
1200 RESTORE
1210 K = 0
1220 FOR I = 1 TO 18
1230 READ R
1240 IF S <> R THEN 1260
1250 K = 1
1260 NEXT I
1270 IF K = 1 THEN 1030
1280 GOTO 580
1290 REM " NEORU (COD 48) 1:1 "
1300 RESTORE
1310 K = 0
1320 FOR I = 1 TO 18
1330 READ R
1340 IF S <> R THEN 1360
1350 K = 1
1360 NEXT I
1370 IF K = 1 THEN 580
1380 IF S > 34 THEN 580

```

```

1370 GOTO 1030
1400 REM " O S1 00 (COD 49 31 50) 111 "
1410 IF T(C) < 42 THEN 1490
1420 IF T(C) = 49 THEN 1450
1430 IF T(C) = 50 THEN 1470
1440 GOTO 500
1450 IF S = 37 THEN 1510
1460 GOTO 500
1470 IF S = 38 THEN 1510
1480 GOTO 500
1490 IF T(C) = 2 THEN 1510
1500 GOTO 500
1510 PRINT " CISTIGUL " ; B(C) * 35 ; " BANI CU MIZA " ; C
1520 D = D - B(C) * 35
1530 P = P + B(C) * 35
1540 NEXT C
1550 PRINT " BANCA " ; D ; " BANI "
1560 PRINT " AVETI " ; P ; " BANI "
1570 IF P > 0 THEN 1600
1580 PRINT " NU MAI AVETI BANI "
1590 GOTO 1430
1600 IF D > 0 THEN 1630
1610 PRINT " S-A GOLIT BANCA "
1620 D = 101000
1630 PRINT " CONTINUAM? (DA/NU) "
1650 INPUT Y0
1660 IF Y0 = "DA" THEN 50
1660 STOP
1470 DATA 1,3,5,7,9,12,14,16,18,19,21,23,25,27
1675 DATA 30,32,34,36
1680 END

```

13.23. Trasarea bioritmului

Programul solicită pentru persoana căreia i se trasează bioritmul următoarele date :

- data nașterii ;
- data de la care se dorește începerea trasării.

Se verifică dacă anul, luna și ziua de naștere sînt corecte, dacă nu, se reia întrebarea. La fel se verifică data trasării și dacă data trasării este după data nașterii.

Se calculează numărul de zile trecute de la data nașterii pînă la data trasării. Diferența de ani (U-A) între trasare și naștere ; diferența de zile (X-Z) între zilele de trasare și naștere ; anii bisecți pentru care se adună cîte o zi (B) ; lunile între naștere și trasare $\Sigma (Zi)$ se iau în considerare la calculul zilelor. Formula de calcul este

$$N = (U - A) * 365 \pm \Sigma (Zi) + (X - Z) + B$$

Se trasează pe o perioadă de 15 zile, începînd cu data solicitată, bioritmul persoanei în cauză. Intervalul de 15 zile a fost ales pentru a avea o imagine suficient de bună (neînghesuită) pe ecran.

După inițializarea ecranului se trasează axa pe care se marchează cele 15 zile. Subprogramul 1000 calculează valoarea curbei sinus în funcție de variabila I cu care se apelează. Pentru variația intelectului periodicitatea este 33 zile, pentru psihic 23 zile iar pentru fizic 28 zile. Curba intelect se marchează cu "+", cea pentru psihic cu "*" iar cea pentru fizic cu ".". Dacă cele 15 zile trec dintr-o lună în luna următoare se apelează subprogramul 1300. Pe ecran se trasează 3 sinusoidă cu caracterele de mai sus.

Programul poate fi reluat pentru altă dată de lansare a trasării bioritmului.

```

5 PRINT " B I G A I T H "
10 DIM T(12)
15 T(1) = 31
20 T(2) = 28
25 T(3) = 31
30 T(4) = 30
35 T(5) = 31
40 T(6) = 30
45 T(7) = 31
50 T(8) = 31
55 T(9) = 30
60 T(10) = 31
65 T(11) = 30
70 T(12) = 31
75 PRINT " DATA NASTURII "
80 PRINT " ANUL "
85 INPUT A
90 PRINT " ZIUA "
95 INPUT Z
100 PRINT " LUNA "
105 INPUT L
110 IF A > 1999 THEN 75
115 IF A < 1900 THEN 75
120 IF Z < 1 THEN 75
125 IF Z > 31 THEN 75
130 IF L < 1 THEN 75
135 IF L > 12 THEN 75
140 PRINT " DATA LANSARII ANALIZEI "
145 PRINT " ANUL "
150 INPUT U
155 PRINT " ZIUA "
160 INPUT X
165 PRINT " LUNA "
170 INPUT Y
175 IF U < A THEN 140
180 IF U > 1999 THEN 140
185 IF X < 1 THEN 140
190 IF X > 31 THEN 140
195 IF Y < 1 THEN 140
200 IF Y > 12 THEN 140
205 R = 0
210 C = A
215 IF INT(C/4) <> C/4 THEN 225
220 B = B+1
225 C = C+1
230 IF C <= U THEN 215
235 IF INT(A/4) <> A/4 THEN 250
240 IF L <= 2 THEN 250
245 B = B-1
250 IF INT(U/4) <> U/4 THEN 265
255 IF Y > 2 THEN 265
260 B = B-1
265 N = (U-A)*365+(X-Z)*B
270 IF Y > L THEN 295
275 FOR I = Y-1 TO L-1
280 M = M-T(I)
285 NEXT I
290 GOTO 310
295 FOR I = L+1 TO Y-1
300 M = M+T(I)
305 NEXT I
310 INTP
320 PRINT AT(15,1);".....",
325 PRINT "....."
330 FOR J = 2 TO 30 STEP 2
340 PRINT AT(15,J);"1"
350 NEXT J
360 ON Y GOTO 370,380,390,400,410,420
365 ON H GOTO 430,440,450,460,470,480
370 PRINT AT(2,2);U;"X";" " ; "IANUARIE"
375 GOTO 490
380 PRINT AT(2,2);U;"X";" " ; "FEBRUARIE"
385 GOTO 490
390 PRINT AT(2,2);U;"X";" " ; "MARTIE"
395 GOTO 490
400 PRINT AT(2,2);U;"X";" " ; "APRILIE"
405 GOTO 490
410 PRINT AT(2,2);U;"X";" " ; "MAI"
415 GOTO 490
420 PRINT AT(2,2);U;"X";" " ; "Iunie"
425 GOTO 490
430 PRINT AT(2,2);U;"X";" " ; "IULIE"
435 GOTO 490
440 PRINT AT(2,2);U;"X";" " ; "AUGUST"
445 GOTO 490
450 PRINT AT(2,2);U;"X";" " ; "SEPTEMBRIE"
455 GOTO 490
460 PRINT AT(2,2);U;"X";" " ; "OCTOMBRIE"
465 GOTO 490
470 PRINT AT(2,2);U;"X";" " ; "NOIEMBRIE"
475 GOTO 490
480 PRINT AT(2,2);U;"X";" " ; "DECEMBRIE"
490 FOR K = 1 TO 21 STEP 10
500 IF X < 10 THEN 530
510 PRINT AT(16,K);X
520 GOTO 540
530 PRINT AT(16,K+1);X
540 X = X+5
550 IF X <= 28 THEN 700
560 IF Y <> 2 THEN 410
570 H = 29
580 IF INT(U/4) = U/4 THEN 490
590 H = 28
600 GOTO 490
610 IF X <= 30 THEN 700
620 H = 30
630 IF Y = 4 THEN 490
640 IF Y = 6 THEN 490
650 IF Y = 9 THEN 490
660 IF Y = 11 THEN 490
670 IF X = 31 THEN 700
680 H = 31
690 X = X-H
700 NEXT K

710 FOR S = 2 TO 30 STEP 2
720 J = 33
730 GOSUB 1000
740 PRINT AT(N,E);" "
750 J = 23
760 GOSUB 1000
770 PRINT AT(N,B);" "
780 J = 28
790 GOSUB 1000
800 PRINT AT(N,B);" "
810 N = N+1
820 X = X+1
830 IF T(Y) < 8 THEN 850
840 GOTO 840
850 GOSUB 1300
860 NEXT B
870 PRINT AT(27,2);"INTELCT = "
880 PRINT AT(28,2);"PSINIC = "
890 PRINT AT(29,2);"FIZIC = "
900 STOP
1000 P = INT(N-INT(N/J)*J)
1005 P = (P/J)*P*2
1010 P = SIN(P)
1015 N = 15-INT(P*10)
1020 RETURN
1300 IF Y <> 2 THEN 1340
1305 IF INT(U/4) <> U/4 THEN 1340
1310 IF X <> 29 THEN 1340
1315 X = X+1
1320 Y = Y+1
1325 IF Y <= 12 THEN 1340
1330 Y = 1
1335 U = U+1
1340 RETURN

```

13.24. Dicționar de sinonime

Programul permite crearea unui dicționar de sinonime în două sau trei limbi.

La creare (lansare program prin RUN) se stabilește dacă se creează în 2 sau 3 limbi și limbile în care se creează. La lansare cu GOTO 10 se utilizează dicționarul creat. În acest caz se poate căuta după oricare limbă (sau după oricare rubrică) un cuvânt obținând celelalte cuvinte (rubrici).

La lansare cu GOTO 10 mai există posibilitatea de completare a dicționarului (fișierului) creat cu cuvinte noi.

Dacă la căutare nu se găsește cuvântul cerut, se revine la secvența de căutare sau de completare a dicționarului.

Programul poate fi utilizat și ca bloc-notes, cele 3 rubrici corespunzătoare celor 3 limbi putând conține orice informație (de ex. numele unei persoane, adresa și numărul de telefon asociate).

```

10 PRINT " DICTIONAR DE SIMONIME "
20 IF M = 0 THEN 400
30 REM " DICTIONAR CREAT "
40 PRINT " DATI CUVINTE NOI (SAU) "
50 INPUT G
60 IF G = 1 THEN 400
70 REM " CAUTARE IN DICTIONAR "
80 PRINT " DUPA CARE LIMBA CAUTATI? "
90 INPUT B$(10)
100 A = 0
110 IF B$ = X$ THEN 180
120 A = L
130 IF B$ = Y$ THEN 180
140 IF M = 2 THEN 80
150 A = 2#1
160 IF B$ = Z$ THEN 180
170 GOTO 80
180 PRINT " DATI CUVINTUL IN LIMBA " ; B$
190 INPUT C$(10)
200 I = 1
210 IF C$ = D$(1, A+1 TO A+L) THEN 260
220 I = I + 1
230 IF I <= N THEN 210
240 PRINT " NU EXISTA CUVINTUL " ; C$
250 GOTO 80
260 PRINT D$(I, 1 TO L) ; " "
270 IF M = 2 THEN 290
280 PRINT D$(I, L+1 TO 2#L) ; " "
290 PRINT D$(I, 1+(M-1)*L TO L)
300 GOTO 80
400 REM " CREARE SI COMPLETARE DICTIONAR "
410 IF M <> 0 THEN 520
420 PRINT " IN 2 SAU 3 LIMBI CREATI? "
430 INPUT H
440 DIM D$(40, M+15)
450 DIM C$(15)
460 PRINT " DATI LIMBILE DICTIONARULUI "
470 INPUT X$
480 INPUT Y$
490 IF M = 2 THEN 510
500 INPUT Z$
510 L = 15
520 IF M > 40 THEN 620
530 PRINT " DATI CUVINTUL IN LIMBA " ; X$
540 INPUT D$(M, 1 TO 15)
550 PRINT " DATI CUVINTUL IN LIMBA " ; Y$
560 INPUT D$(M, 16 TO 30)
570 IF M = 2 THEN 600
580 PRINT " DATI CUVINTUL IN LIMBA " ; Z$
590 INPUT D$(M, 31 TO 45)
600 M = M + 1
610 GOTO 520
620 STOP
630 END

```


13.25. Ordonarea unui set de informații

Programul posedă un set de date (în cazul de față datele constau din numele, greutatea și înălțimea unui grup de persoane) pe care le poate ordona după un criteriu dat.

În exemplul de față, ordonarea se face pentru ordinea alfabetică, după greutate sau după înălțime. După ordonare, se afișează lista de date ordonată.

```

5 PRINT " ORDONAREA DUPA ALFABET "
7 PRINT " GREUTATE SAU INALTIME "
10 DIM A$(20,10),S(20),H(20)
15 N = 0
20 N = N+1
25 READ A$(N,10)
30 READ H(N)
35 READ S(N)
40 IF A$(N) <> "STOP" THEN 20
45 PRINT " DATI CRITERIUL DE ORDONARE "
50 PRINT " 1 = ALFABET "
55 PRINT " 2 = GREUTATE "
60 PRINT " 3 = INALTIME "
65 INPUT C
70 REM " ORDONAREA "
75 C = 0
80 FOR K = 1 TO N-1
85 ON C GOTO 90,100,110
90 IF A$(K) > A$(K+1) THEN 120
95 GOTO 170
100 IF S(K) > S(K+1) THEN 120
105 GOTO 170
110 IF H(K) > H(K+1) THEN 120
115 GOTO 170
120 S = S(K)
125 S(K) = S(K+1)
130 S(K+1) = S
135 B$ = A$(K,10)
140 A$(K) = A$(K+1,10)
145 A$(K+1) = B$(10)
150 H = H(K)
155 H(K) = H(K+1)
160 H(K+1) = H
165 C = C+1
170 NEXT K
175 IF C > 0 THEN 75
180 ON C GOTO 185,195,205
185 PRINT " NUME "; " INALTIME "; " GREUTATE "
190 GOTO 210
195 PRINT " NUME "; " GREUTATE "
200 GOTO 210
205 PRINT " NUME "; " INALTIME "
210 FOR I = 1 TO N
215 ON C GOTO 220,230,240
220 PRINT A$(I); " "; H(I); " "; S(I)
225 GOTO 245
230 PRINT A$(I); " "; S(I)
235 GOTO 245
240 PRINT A$(I); " "; H(I)
245 NEXT I
250 STOP
260 DATA "POPESCU",200,80
265 DATA "CONSTANTIN",175,65.4
270 DATA "BARNA",180,70
275 DATA "MARESCU",185,71
280 DATA "MICHID",205,91
285 DATA "STOP",0,0
300 END

```

13.26. Decodificarea numerelor romane

Programul servește la transformarea numerelor romane în numere zecimale. Se introduce numărul roman sub forma unui șir de litere. Decodificarea se face în baza regulilor definite prin DATA în liniile 440—550 din program. Regulile specifică succesiunea corectă de cifre romane; o succesiune eronată este semnalată printr-un mesaj. Numărul decodificat este apoi afișat pe ecran.

După introducerea numărului roman în șirul T\$ și a lunginii numărului (din câte litere se compune) în L, se citește prima literă și se verifică dacă e din setul de litere permise într-un număr roman (M, D, C, L, X, V, I). Prima literă poate fi oricare dintre aceste litere. Aceasta e specificată în prima linie de DATA unde de exemplu 100002 înseamnă că dacă litera e M valoarea este de 1000 iar după M pot să apară litere după regula (Ø2) care se găsește în linia următoare de DATA. Valoarea-1 în DATA înseamnă că numărul respectiv nu poate apare, deci se semnalează succesiunea eronată.

```

10  REM  DECODIFICAREA NUMERELOR ROMANE *
20  DIM A(10,7),C$(17,1)
30  MAT READ A(1,7)
40  C$(1) = "M"
50  C$(2) = "D"
60  C$(3) = "C"
70  C$(4) = "L"
80  C$(5) = "X"
90  C$(6) = "V"
100 C$(7) = "I"
110 DIM I$(20)
120 E = 1
130 H = 0
140 C = 0
150 P = 0
160 PRINT " DATA NUMARUL ROMAN"
170 INPUT T$(10)
180 L = 0
190 FOR I = 1 TO 20
200 IF T$(I0) = " " THEN 220
210 L = L+1
220 NEXT I
230 FOR J = 1 TO L
240 K = 0
250 FOR J = 1 TO 7
260 IF T$(I0J) <> C$(J) THEN 280
270 K = J
280 NEXT J
290 IF K <> 0 THEN 320
300 PRINT " NUMAR ROMAN ERONAT"
310 GOTO 120
320 X = A(I,K)
330 IF X = 0 THEN 300
340 C = (1-ABS(SGN(K-F)))*C+1
350 IF C > 2 THEN 300
360 P = K
370 H = H+INT(X/100)
380 R = X-100*INT(X/100)
390 IF R <> 0 THEN 410
400 I = L
410 NEXT I
420 PRINT H
430 GOTO 120
440 REM * DECODIFICAREA CONDIȚIILOR P1
450 REM * M      D      C      L      X      V      I
460 DATA 100002,50000,10000,5005,1009,507,110
470 DATA 100002,50003,10008,5005,1009,507,110
480 DATA -1,-1,10009,5005,1009,507,110
490 DATA -1,-1,10004,5005,1009,507,110
500 DATA -1,-1,-1,5006,1009,507,110
510 DATA -1,-1,-1,-1,1004,507,110
520 DATA -1,-1,-1,-1,-1,-1,107
530 DATA 80005,30005,10004,5006,1009,507,110
540 DATA -1,-1,6007,3007,1004,507,110
550 DATA -1,-1,-1,-1,800,300,107
560 END

```

13.27. Mira de control AMIC

Programul generează o imagine de control asemănătoare mirei TV, cuprinzând trasări de drepte orizontale și verticale, a unui cerc circumscris unui pătrat, și a setului de caractere, inclusiv cele semigrafice. Mira obținută poate fi utilizată la reglajul calității imaginii TV.

```

10 RUN * IMAGINE DE CONTROL *
20 INIT P
30 PRINT AT(2,1):"MMN IMAGINE DE CONTROL MMN"
40 WINDOW 0,150,0,160
50 MOVE 152,75
60 FOR I = 0 TO 2*PI STEP PI/10
70 DRAW 75*67*CO5(I),75*67*SIN(I)
80 NEXT I
90 MOVE 0,142
100 DRAW 140,142
110 MOVE 0,132
120 DRAW 140,132
130 PRINT AT(5,1):"00"
140 PRINT AT(5,23):"00"
150 PRINT AT(6,1):"00"
160 PRINT AT(6,23):"00"
170 PRINT AT(29,1):"00"
180 PRINT AT(29,23):"00"
190 PRINT AT(30,1):"00"
200 PRINT AT(30,23):"00"
210 MOVE 0,117
220 DRAW 140,117
230 MOVE 0,35
240 DRAW 140,35
250 MOVE 0,20
260 DRAW 140,20
270 MOVE 0,9
280 DRAW 140,9
290 PRINT AT(7,7):"XXXXXXXXXX"
300 PRINT AT(8,7):"XXXXXXXXXX"
310 PRINT AT(9,7):"XXXXXXXXXX"
320 PRINT AT(26,7):"XXXXXXXXXX"
330 PRINT AT(27,7):"XXXXXXXXXX"
340 PRINT AT(28,7):"XXXXXXXXXX"
350 MOVE 0,100
360 DRAW 140,100
370 MOVE 0,50
380 DRAW 140,50
390 MOVE 0,75
400 DRAW 25,75
410 MOVE 120,75
420 DRAW 140,75
430 MOVE 17,0
440 DRAW 17,140
450 MOVE 25,0
460 DRAW 25,140
470 MOVE 120,0
480 DRAW 120,140
490 MOVE 137,0
500 DRAW 137,140
510 PRINT AT(14,4):"11*020*(1)*+..?"
520 PRINT AT(16,5):"0123456789:;<=1?"
530 PRINT AT(18,5):"ABCDEFGHIJKLMN0P"
540 PRINT AT(20,5):"RSTUVUZ (\)"
550 MOVE 41,0
560 DRAW 41,50
570 MOVE 41,100
580 DRAW 41,140
590 MOVE 107,0
600 DRAW 107,50
610 MOVE 107,100
620 DRAW 107,140
630 REM *STOP*
640 GOT0430
650 END.

```

13.28. Ceas electronic

Programul solicită prin dialog ora exactă, după care afișează imaginea unui ceas cu arătătoarele în mișcare. În dreapta-jos a ecranului se afișează și ora sub forma HHMM. Reglajul avansului se face în linia 250 din program. Constanta T are valoarea de aprox. 11 000, o valoare mai mare ducând la "întirzirea" ceasului.

```

10 PRINT " C E A S U L "
20 PRINT " DATI ORA SE PORNIRE "
30 INPUT H
40 PRINT " MINUTUL "
50 INPUT M
60 INTP
70 PRINT AT(6,13);"="
80 PRINT AT(14,21);"="
90 PRINT AT(26,13);"="
100 PRINT AT(16,5);"="
110 MOVE S2,S2
120 G = (H*PI/30)
130 DRAW S2*30*SIN(G),S2*30*COS(G)
140 MOVE S2,S2
150 G = (H*M/60)*PI/6
160 DRAW S2*30*SIN(G),S2*30*COS(G)
170 PRINT AT(30,24);H;"'";M
180 H = H+1
190 IF H = 60 THEN 210
200 GOTO 250
210 M = 0
220 H = H+1
230 IF H <> 24 THEN 250
240 H = 0
250 T = 11000
260 FOR I = 1 TO T
270 NEXT I
280 GOTO 60
290 END

```

13.29. Anagrame

Programul solicită introducerea unui șir de max. 20 de caractere, afișând apoi combinații aleatoare ale șirului.

Cuvântul se introduce în variabila șir A\$, iar lungimea șirului în N. În masivul A (N) se aleg aleator cifre între 1 și N care determină litera a A (I)-a din cuvânt. După completarea lui A (N) se afișează literele cuvântului în ordinea din A (N). După afișare se reia ciclul. Programul se continuă până la oprire cu CTRL/C, după care se poate relansa cu alt cuvânt (șir).

```

5  AN = " A N A G R A X A "
10 DIM A$(30)
20 PRINT " DATI UN CUVINT "
30 INPUT A$(10)
40 N = 0
50 FOR I = 1 TO 20
60 IF A$(170) = " " THEN 80
70 N = N+1
80 NEXT I
90 DIM A(N)
100 FOR I = 1 TO 7
130 A(I) = INT(RND*I*N)+1
140 U = 0
150 FOR J = 1 TO I-1
160 IF A(J) <> A(I) THEN 180
170 U = J
180 NEXT J
190 IF U <> 0 THEN 130
200 NEXT I
210 AN = " "
220 FOR B = 1 TO N
230 B$ = B$+A$(A(B))
240 NEXT B
250 PRINT B$
260 NEXT X
270 GOTO 100
280 END

```

13.30. Bugetul cheltuielilor zilnice într-o familie

Programul solicită introducerea cheltuielilor efectuate pe zile; noile cheltuieli se introduc după ultima zi completată într-o rulare anterioară. La cerere, se afișează totalul cheltuielilor și histograma cheltuielilor pe zile.

Dacă se introduc cheltuielile pentru primele zile se lansează programul cu RUN. Pentru completare de cheltuieli se lansează cu GOTO 20. Histograma cheltuielilor se dă la cerere.

```

10 DIM A(31)
15 I = 1
20 PRINT " CITO ZILE INTRODUCETI? "
25 INPUT N
30 FOR J = 1 TO N
40 INPUT A(I)
50 I = I + 1
60 NEXT J
70 PRINT " VRETI HISTOGRAMA DA/NU "
90 INPUT B$
95 IF B$ <> "DA" THEN GOTO 10
100 C = A(1)
110 T = A(1)
120 FOR J = 2 TO I
130 B = C - A(J)
140 IF B >= 0 THEN GOTO 160
150 C = A(J)
160 T = T + A(J)
170 NEXT J
180 INIT$
190 PRINT AT(1,1); " HISTOGRAMA CHELTUIELILOR "
195 PRINT AT(2,1); " PE "; I; " ZILE "
200 PRINT AT(3,1); " TOTAL CHELTUIELI "; T
210 WINDOW = 10,110,10,110
220 MOVE 0,-10
230 DRAW 0,100
240 MOVE -10,0
250 DRAW 100,0
260 FOR J = 1 TO I
270 A(J) = 100 * A(J) / C
280 MOVE J * 3,0
290 DRAW J * 3, A(J)
300 NEXT J
310 STOP
320 END

```

13.31. Microfișier

Programul permite crearea și exploatarea unei microbaze de date generată sub forma unui tabel de articole. Fiecare articol poate conține mai multe câmpuri de lungimi diferite.

La creare se definește baza de date, prin următorii parametri:

- nume;
- număr maxim de articole (max. 254);
- număr câmpuri/articol;
- denumirile și lungimile fiecărui câmp.

În funcție de parametrii dați, se dimensionează tabelul ce va conține baza de date. Se verifică dubbele definiții pentru numele asociate câmpurilor.

Exploatarea bazei de date se execută cu ajutorul următoarelor comenzi:

a) introducere articol — se caută o linie (articol) vidă din tabel și se introduc valorile asociate fiecărui câmp din articol. Introducerea unui articol

se execută prin afişarea numelui fiecărui câmp și introducerea conținutului câmpului de către operator.

Dacă numai există loc pentru un nou articol (tabel plin) se emite mesajul FIȘIER PLIN, după care se poate utiliza comanda de ștergere articol și introducerea celui solicitat.

b) ștergere articole după conținut — se șterg articolele care conțin informații identice într-un câmp dat. Se solicită denumirea câmpului și conținutul său. Se caută fiecare articol care conține în câmpul respectiv informația dată și se șterge articolul respectiv.

c) listare articole — se listează articolele din cadrul fișierului.

d) modificare câmpuri după conținut — se solicită numele câmpului, vechiul conținut și noul conținut. Se listează fiecare articol ce conține câmpul cu informația identică cu „conținut vechi” și se întreabă operatorul dacă dorește sau nu modificarea articolului.

```

5 PRINT " MICROFIȘIER "
10 IF P <> 0 THEN 215
15 REM " SE CREEAZA FIȘIER NOU "
20 PRINT " DATI NUME FIȘIER "
25 INPUT P$
30 I = 1
35 PRINT " DATI NUMAR MAXIM ARTICOLE "
40 INPUT T
45 PRINT " DATI NUMAR CÂMPURI/ARTICOL "
50 INPUT M
55 DIM C$(M,10),L$(M),S$(T),S$(3,10),X$(10)
60 MAT S = ZER
65 PRINT " DATI DENUMIRE SI LUNGIME CÂMP "
70 D = 0
75 FOR I = 1 TO M
80 INPUT C$(I,10)
85 INPUT L(I)
90 D = D+L(I)
95 NEXT I
100 D = 1
105 FOR I = 1 TO M-1
115 X$ = C$(I)
120 FOR C = 1+1 TO M
125 IF X$ <> C$(C) THEN 135
130 D = -1
135 NEXT C
140 NEXT I
145 IF D = 1 THEN 160
150 PRINT " CÂMPURI CU ACELAȘI NUME "
155 STOP
160 I = 0
165 PRINT " INTRODUCETI COMANDA "
170 PRINT " 1 = INTRODUCERE ARTICOL "
175 PRINT " 2 = ȘTERGERE ART. DUPA CONTINUT "
180 PRINT " 3 = LISTARE ARTICOLE "
185 PRINT " 4 = MODIF.CÂMPURI DUPA CONTINUT "
190 PRINT " 5 = LISTARE PT. MODIFICARI "
195 PRINT " 6 = TERMINARE SESIUNE "
200 INPUT C
205 ON C GOTO 215,325,395,430,540,155
210 GOTO 165
215 PRINT " SE LUCREAZA CU FIȘIERUL ",P$
220 REM " INTRODUCERE ARTICOLE "
225 IF L < T THEN 240
230 PRINT " FIȘIER PLIN "
235 GOTO 165
240 B = 1
245 IF B > T THEN 230
250 IF S(B) = 0 THEN 265
255 B = B+1
260 GOTO 245
265 A = 0
270 FOR I = 1 TO M
275 PRINT C$(I)
280 INPUT F$(B,A+1)TOA+L(I)
285 A = A+L(I)
290 NEXT I
295 S(D) = 2
300 L = L+1
305 PRINT " MAI (INTRODUCETI) (SA=1) "
310 INPUT C
315 IF C = 1 THEN 245
320 GOTO 165
325 PRINT " SE LUCREAZA CU FIȘIERUL ",P$
330 REM " ȘTERGERE ARTICOLE DUPA CONTINUT "
335 PRINT " DATI NUME SI CONTINUT CÂMP "
340 PRINT " DUPA CARE SE ȘTERG ARTICOLELE "
345 INPUT S$(1,10)
350 INPUT S$(2,10)
355 GOSUB 1000
360 IF I < 0 THEN 165
365 D = 1
370 GOSUB 1100
375 IF D > T THEN 165
380 S(D) = 0
385 L = L-1
390 GOTO 370
395 PRINT " SE LUCREAZA CU FIȘIERUL ",P$
400 REM " LISTARE FIȘIER "
405 FOR I = 1 TO M
410 PRINT C$(I);", "
415 NEXT I
420 PRINT
425 FOR I = 1 TO T
430 IF S(I) = 0 THEN 440
435 GOSUB 1200
440 NEXT I
445 GOTO 165
450 PRINT " SE LUCREAZA CU FIȘIERUL ",P$
455 REM " MODIF.CÂMPURI DUPA CONTINUT "
460 PRINT " DATI NUME CÂMP SI CONTINUT "
465 PRINT " VECHI SI NOU "
470 INPUT S$(1,10)
475 INPUT S$(2,10)
480 INPUT S$(3,10)
485 GOSUB 1000
490 IF I < 1 THEN 165
495 D = 1
500 GOSUB 1100
510 IF D > T THEN 165
515 GOSUB 1200
515 PRINT " SE MODIFICA (SA=1) "
520 INPUT C
525 IF C = 1 THEN 335
530 F$(B,A+1)TOA+L(I) = S$(3)
535 GOTO 500
540 PRINT " SE LUCREAZA CU FIȘIERUL ",P$
545 REM " LISTARE FIȘIER PT. MODIFICARE "
550 D = 1
555 GOSUB 1200
560 IF D > T THEN 165
565 GOSUB 1300

```

```

570 B = B+1
575 GOTO 165
1000 REM * CAUTARE NUME CIMP *
1005 I = 1
1010 A = 0
1015 IF I > N THEN 1040
1020 IF C(I) = S(I) THEN 1030
1025 A = A+L(I)
1030 I = I+1
1035 GOTO 1015
1040 PRINT * NU EXISTA CIMP *;S(I)
1045 I = -1
1050 RETURN
1100 REM * CAUTARE CONTINUT *
1105 IF D > T THEN 1130
1110 IF S(I) <> 2 THEN 1170
1115 IF P(D,A+100*L(I)) = S(2) THEN 1130
1120 B = B+1
1125 GOTO 1105
1130 RETURN
1200 REM * TIPARIRE ARTICOL *
1205 B = 0
1210 FOR J = 1 TO N
1215 PRINT P(D,B+100*L(J))". ".
1220 B = B+L(J)
1230 NEXT J
1235 PRINT
1240 RETURN
1300 REM * MODIFICAREA *
1305 PRINT * AL CITELSA CIMP SE MODIF
1306 PRINT * 0 = NU SE MODIFICA *
1310 INPUT C
1315 IF C = 0 THEN 1360
1320 PRINT * BATI CONTINUT NOU *
1325 INPUT S(J,10)
1330 B = 0
1332 IF C = 1 THEN 1305
1335 FOR J = 1 TO C-1
1340 B = B+L(J)
1345 NEXT J
1355 GOTO 1305
1360 RETURN
1365 END

```

e) Listare articole și modificare — se listează pe rând fiecare articol. Se întreabă dacă se dorește modificarea unuia sau mai multor cîmpuri din articol.
 {Procedura continuă pînă la sfîrșitul fișierului.

13.32. Universul Conway

Se poate studia prin aceasta creșterea, modificarea, înmulțirea și moartea celulelor dintr-o colonie. Viața celulei este influențată de celulele înconjurătoare.

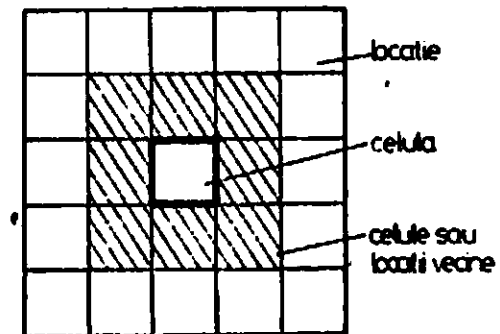
{ La început pe o zonă de $N \times M$ locații se generează celule vii conform coordonatelor date de utilizator. O locație poate avea două situații: conține sau nu o celulă vie. Din starea inițială în baza legilor geneticii se ajunge la generații următoare. Starea locațiilor în următoarea generație poate fi:

- 1 — celula vie supraviețuiește;
- 2 — celula vie moare;
- 3 — s-a generat o nouă celulă vie.

Cele trei stări se determină după următoarele reguli:

1. Supraviețuiește fiecare celulă dacă are două sau trei celule vii vecine.

Fig. 13.9. Organizarea ecranului pentru universul lui Conway.



```

5 PRINT " UNIVERSUL CONWAY "
7 PRINT " ----- "
10 PRINT " DATE CINFUL "
15 INPUT N,M
20 IF N > 30 THEN 10
25 IF M > 30 THEN 10
30 DIM U(N,M)
40 MAT U = /ER
50 PRINT " CITE CELULE VII INTRODUCETI? "
60 INPUT C
70 IF C <= N*M/2 THEN 100
80 GOTO 50
100 PRINT " DATE COORDONATELE CELULELOR VII "
110 FOR I = 1 TO C
120 INPUT X,Y
130 IF X > N THEN 120
140 IF Y > M THEN 120
150 IF X < 1 THEN 120
160 IF Y < 1 THEN 120
170 U(X,Y) = 1
180 NEXT I
185 INITP
190 PRINT AT(1,2): " STAREA INITIALA "
195 GOTO 220
200 INITF
210 PRINT AT(1,2): " GENERATIA URMATOARE "
220 FOR I = 1 TO N
230 PRINT AT(I+2,1)
240 FOR J = 1 TO M
250 IF U(I,J) = 0 THEN 290
260 PRINT "N";
270 GOTO 290
280 PRINT " ";
290 NEXT J
300 NEXT I
500 I = 1
510 FOR J = 1 TO N
520 C = 0
530 K = I + 1
540 IF K <= M THEN 560
550 K = M
560 L = I - 1
570 IF L >= 1 THEN 590
580 L = 1
590 FOR X = L TO K
600 O = J + 1
610 IF O <= M THEN 630
620 O = M
630 P = J - 1
640 IF P >= 1 THEN 660
650 P = 1
660 FOR Y = P TO O
670 IF INT( U(Y,X)/2 ) = U(Y,X)/2 THEN 690
680 C = C + 1
690 NEXT Y
700 NEXT X
710 D = 0
720 IF INT( U(J,I)/2 ) = U(J,I)/2 THEN 740
730 D = 1
740 C = C - D
750 IF C = 3 THEN 810
760 IF C <> 2 THEN 790
770 GOTO 830
790 D = D + 4
800 GOTO 820
810 D = D + 2
820 U(J,I) = D
830 NEXT J
840 I = I + 1
850 IF I <= M THEN 510
2000 FOR J = 1 TO M
2005 FOR I = 1 TO N
2010 IF U(I,J) > 3 THEN 2025
2015 IF U(I,J) > 1 THEN 2035
2020 GOTO 2045
2025 O = 0
2030 GOTO 2040
2035 G = 1
2040 U(I,J) = O
2045 NEXT I
2050 NEXT J
2055 GOTO 200
2060 END

```



2. Moare — dacă are patru sau mai multe celule vii vecine, motivul este suprapopulația,

— dacă are una sau nici-o celulă vie vecină, motivul fiind izolarea.

3. Naștere — în fiecare locație goală se naște o celulă vie dacă are trei (nici mai multe nici mai puține) celule vii vecine.

13.33. Pătratul magic

Într-un pătrat de mărime dată se așază numere în așa fel încît pe orizontală pe verticală și în diagonală suma numerelor să fie aceeași. Există diferite metode de a genera pătratele magice de diferite mărimi.

Primul exemplu generează pătrate magice de grad impar utilizînd o generare bazîndu-se de următorul algoritm :

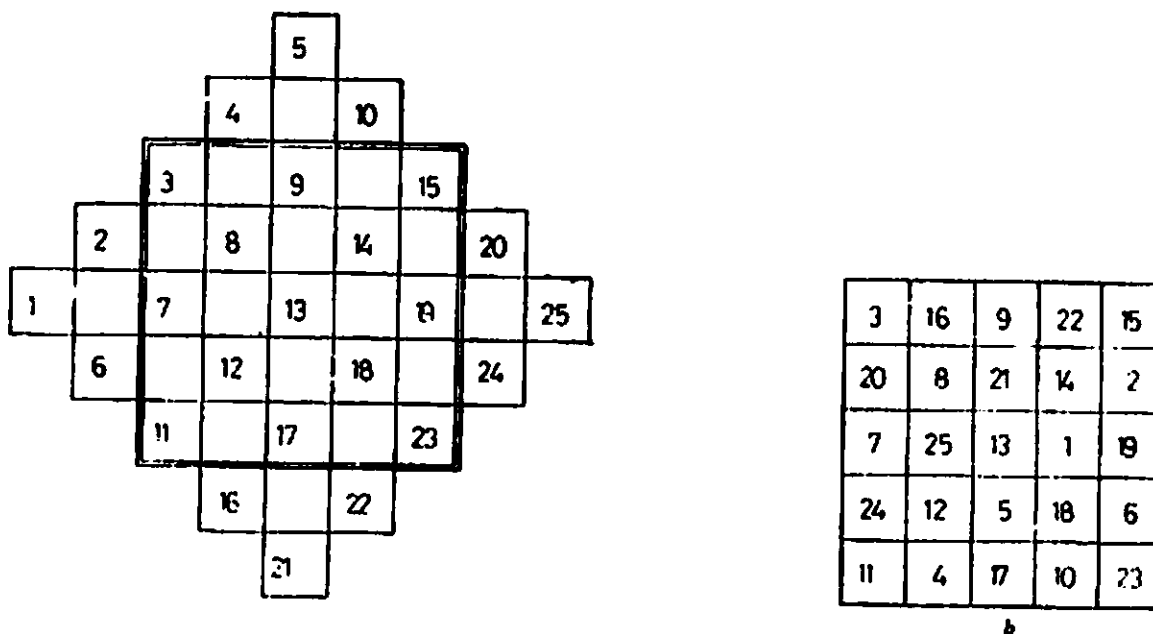


Fig. 13.10. Pătratul magic : (a) exemplu, (b) generale.

Pe pătratul de gradul dat se desenează piramide de pătrățele și în diagonalele astfel formate se trec cifrele în ordine crescătoare după care cifrele în afara pătratului se mută cu atîtea poziții cît este gradul pătratului.

Al doilea exemplu generează mai multe pătrate de gradul impar utilizînd următorul algoritm :

Din șirul de numere 1, 2, 3, 4, 5 ... pînă la gradul pătratului și 0, n, $2 \cdot n$, $3 \cdot n$... (unde n e gradul) pînă la $(n-1) \cdot n$ aleator se generează două pătrate de gradul dat sub forma :

— se alege din primul șir un șir aleator și se pune în primul rînd al pătratului ;

```

5 PRINT " PATRAT MAGIC "
7 PRINT " ----- "
10 PRINT " DATI GRADUL PATRATULUI "
20 INPUT N
30 IF N > 0 THEN 70
40 PRINT " GRAD NEGATIV "
50 GOTO 20
70 IF INT( N/2 ) <> N/2 THEN 100
80 PRINT " GRAD PAR "
90 GOTO 20
95 IF N = 1 THEN 100
95 PRINT " GRAD IMPAR "
97 GOTO 20
100 DIM M(N,N)
110 MAT M = ZER
120 J = INT(N/2)
130 I = J + 1
140 V = 1
150 IF M(I,J) = 0 THEN 210
160 IF V = N*2 + 1 THEN 460
170 I = I + 1
180 J = J - 1
190 IF I <= N THEN 150
200 GOTO 310
210 M(I,J) = V
220 IF V = N*2 THEN 400
230 V = V + 1
240 I = I + 1
250 J = J + 1
260 IF I > N THEN 300
270 IF J <= N THEN 290
280 J = J - 1
290 GOTO 150
300 IF J > N THEN 320
310 I = 1
315 GOTO 150
320 I = 2
330 J = N
340 GOTO 150
400 A = (30 - N*2)/2
402 A = A-2
410 FOR I = 1 TO N
415 D = A + I - 1
417 GOSUB 500
418 PRINT AT(B,D-1-(I-1)*3);"I"
420 FOR J = 1 TO N
425 C = D-(J-1)*3-(I-1)*3
426 IF M(I,J) >= 10 THEN 430
427 C = C + 1
430 PRINT AT(B,C);M(I,J);"I"
440 NEXT J
445 A = A + 2
450 D = A + 1 - 1
450 GOSUB 600
465 PRINT
470 STOP
500 FOR K = 0 TO N
504 GOSUB 600
505 PRINT AT(B-1,B-1+3*N-(I-1)*3);"I"
510 NEXT K
515 RETURN
600 PRINT AT(C-2,D-(I-1)*3-1);
605 FOR L = 0 TO 3*N
610 PRINT " ";
615 NEXT L
620 RETURN
630 END

```

— restul liniilor se completează pornind cu numărul din pătratul de după mijlocul liniei și continuând cu primele;

— din șirul al doilea de asemenea se alege o combinație aleatoare;

— restul liniilor se completează din acest șir dar acum pornind din mijlocul liniei.

Adunând cele două pătrate se obține un pătrat magic :

2	1	4	3	5
3	5	2	1	4
1	4	3	5	2
5	2	1	4	3
4	3	5	2	1

Fig. 13.11. Primul pătrat.

5	0	20	10	15
20	10	15	5	0
15	5	0	20	10
0	20	10	15	5
10	15	5	0	20

Fig. 13.12. Al doilea pătrat.

7	1	24	13	20
23	15	17	6	4
16	9	3	25	12
5	22	11	19	8
14	18	10	2	21

Fig. 13.13. Pătratul rezultat.

```

10 PRINT "PATRAT MAGIC "
12 PRINT "-----"
20 DIM A(5),B(5),N(5),M(5),O(5),P(5)
30 DIM C(5,5),D(5,5),E(5,5)
40 MAT READ O
50 MAT READ P
60 FOR X = 1 TO 5
70 N(1) = INT( RND(X) * 5 ) + 1
80 FOR Z = 2 TO 5
90 N(Z) = INT( RND(X) * 5 ) + 1
100 U = 0
110 FOR J = 1 TO Z - 1
120 IF N(J) <> N(Z) THEN 140
130 U = J
140 NEXT J
150 IF U <> 0 THEN 90
160 NEXT Z
165 NEXT X
170 FOR X = 1 TO 5
180 N(1) = INT( RND(X) * 5 ) + 1
190 FOR Z = 2 TO 5
200 N(Z) = INT( RND(X) * 5 ) + 1
210 U = 0
220 FOR J = 1 TO Z - 1
230 IF N(J) <> N(Z) THEN 250
240 U = J
250 NEXT J
260 IF U <> 0 THEN 200
270 NEXT Z
280 FOR J = 1 TO 5
290 A(J) = O(N(J))
300 B(J) = P(N(J))
310 NEXT J
315 NEXT X
320 FOR I = 1 TO 5
330 FOR J = 1 TO 5
340 C(I,J) = A(J)
350 D(I,J) = B(J)
360 NEXT J
370 M(1) = A(1)
380 M(2) = A(5)
390 M(3) = A(1)
400 M(4) = A(2)
410 M(5) = A(3)
420 N(1) = B(3)
430 N(2) = B(4)
440 N(3) = B(5)
450 N(4) = B(1)
460 N(5) = B(2)
470 MAT A = N
480 MAT B = N
490 NEXT I
500 MAT E = C + D
510 INITP
520 PRINT AT(2,2): "PATRAT MAGIC "
525 PRINT AT(3,2): "-----"
530 X = 0
540 FOR I = 1 TO 5
550 Y = 0
560 K = X + 4
570 FOR J = 1 TO 5
580 Y = Y + 4
590 Q = Y
590 IF E(I,J) > 9 THEN 610
600 Q = Q + 1
610 PRINT AT(Q,M(I)): " "
620 NEXT J
630 NEXT I
640 GOTO 40
670 DATA 1,2,3,4,5

```

Testarea resurselor hardware și a interpretorului BASIC

Procedura de punere la punct a microcalculatorului urmează următoarea înlănțuire de faze distincte :

- pe placheta implantată cu componente electronice nealimentată se verifică — ohmetric — magistrala de adrese, date, comenzi; scurtcircuite între alimentări (respectiv masă), cum și restul pinilor de circuite;

- pe placheta alimentată, dar fără circuitele ISI se verifică prezența tensiunilor, nivelul lor, cum și modulele funcționale, în următoarea ordine: prezența și corectitudinea semnalelor delivrate de sincrogenerator, corectitudinea tastării RESET și INI;

- pe placheta alimentată cu circuitele LSI prezentate și cu un EPROM de test programat cu NOP (de la adresa 0) se verifică magistrala de adrese și date;

- se verifică corectitudinea semnalelor microprocesorului de comandă a memoriei, de selecție etc. în conformitate cu diagrama de timp, înlocuind EPROM-ul de test cu un altul care asigură un program de scriere/citire în memoria RAM și selecția circuitelor EPROM de pe plachetă;

- cu EPROM-ul monitor implantat se verifică corectitudinea tastării, se reglează interfața cu receptorul TV și casetofonul audio;

- se trec teste hardware, se verifică comenzile de monitor și setul de instrucțiuni BASIC.

14.1. Prezentare generală a setului de programe de test

Setul de programe realizat urmărește testarea logic-funcțională a modulelor harda microcalculatorului aMIC.

Programul TEST aMIC are următoarele module, corespunzătoare modulelor funcționale pe care le testează :

R-RAM	Testarea zonei de memorie RAM
E-EPROM	Testarea zonei de memorie EPROM
D-DISPLAY	Testarea afișării pe ecran
K-TASTATURA	Testarea preluării de caractere de la tastatură.

Aceste programe pot fi executate în regim automat, ciclind, fără intervenția operatorului, sau în regim manual în care operatorul poate selecta testul dorit.

În afară de aceste teste există proceduri de testare a interpretorului BASIC și a transferului de informații dinspre/spre casetofon.

Programul TEST AMIC este disponibil pe caseta magnetică, într-o variantă simplificată, sau este înscris pe EPROM. Completat cu câteva subrutine modificate din monitor, formează conținutul unei capsule 2716 de 2 Ko. Acest cip este amplasat în locul primului cip din EPROM-ul suport al programului BASIC. Programul utilizează generatorul de caractere, tabela de simboluri pentru tastatură și câteva subrutine din monitor.

Lansarea programului TEST AMIC se face prin intermediul monitorului, cu comanda

G0 800 (CR)

Această comandă cedază controlul monitorului de comenzi al programului de test. După ce se face inițializarea slivei și a variabilelor program (în zona de memorie RAM 5F00H 5FFFH, corespunzătoare ultimei linii de caractere) se afișează mesajul:

TEST AMIC
MOD DE LUCRU : A SAU M.

Apăsarea unei alte taste decât A sau M va produce menținerea mesajului de mai sus.

Următorul mesaj:

IN 2 CIP1 SAU 2 ?

Se cere numărul cipului de memorie EPROM plasat în poziția 2. Această mențiune este necesară pentru textul EPROM, deoarece cipul care conține TEST AMIC este plasat în poziția 1. În funcție de răspunsul la acest mesaj se va lua în considerare CHECKSUM-ul cipului 1 sau 2 la testarea poziției 2. din cele 8 EPROM-uri.

În funcție de modul de lucru, tratarea se ramifică în continuare.

REGIMUL AUTOMAT: Se cere:

ORDINEA TESTELOR

care vor fi executate în regimul AUTOMAT.

În acest regim pot fi executate testele RAM, EPROM și DISPLAY.

Ca răspuns la acest mesaj se introduce combinația dorită a testelor, permițându-se și repetarea unor teste cu condiția ca numărul total să nu depășească 10. Dacă succesiunea de caractere R, E și respectiv D se termină cu C, atunci programul va cicla pe succesiunea de teste anterior introdusă. Comanda S intercalată în succesiunea de comenzi va avea ca efect terminarea testelor și revenirea în monitorul microcalculatorului.

După introducerea celei de-a zecea comenzi, apare mesajul:

ZONE RAM TEST:

Acestea pot fi în funcție de varianta microcalculatorului, următoarele: 60-00; 60-C0; 60-80; 40-80, A0-00 (memoria RAM alocată DISPLAY-

ului poate fi în zona 40—60 sau 80—A0). În continuare vor fi executate testele înscrise în listă, în aceeași ordine. Dacă în lista testelor nu figurează comanda C atunci, după execuție, se trece controlul monitorului de bază.

Regimul manual :

Pe ecran apare mesajul :

SIMBOL. TEST ; așteptându-se una dintre comenzile R, E, D sau K, care să lanseze programul test corespunzător. Comanda S trece controlul monitorului de bază iar orice tastă apăsată reafixează mesajul de mai sus. De asemenea, după execuția unui test se revine cu același mesaj (la dispoziția operatorului).

14.2. Comanda E — testarea zonei de memorie EPROM

Testul se execută în mod identic în ambele regimuri de lucru. Acest test constă în calcularea sumei de control pentru fiecare cip de memorie EPROM-2716 și compararea ei cu suma corespunzătoare anterior calculată și înscrisă într-o listă de „semnături” în programul TEST aMIC.

Programul care urmează calculează sumele de control ale celor opt cipuri 2716 existente și afișează valorile respective. Acest program se utilizează la fiecare modificare a informației în memoria EPROM, valorile obținute pentru sumele de control trebuind să fie introduse în tabelul de „semnături” din EPROM-ul de test.

```

CSMART : MVI E,00          ; Se încarcă în E nr. cipului
ESI :    MOV B,E           ; de început
         CALL CSC          ; Calcul suma de control a cipului respectiv
         MOV D,A           ; Se încarcă în D suma de control
         CALL CRLF        ; Se afișează nr. cipului
         MOV A,B
         CALL BINASC
         CALL AF20H       ; Se afișează suma de control
         MOV A,D
         CALL BINASC
         INR E             ; Increment nr. cipului
         CALL KEYIN       ; Așteaptă comanda „continuă”
         CPI 43
         JZ CS1
         JMP MONIT.

```

Subrutina CSC calculează suma de control a cipului adresat

Intrare : B = nr. cipului adresat

Ieșire : A = suma de control

```

ESC :    PUSH B
         MOV A,B           ; calculul adresei de început
         RLC
         RLC
         RLC

```

```

MOV H, A           ; (H, L) conține adresa de început a cipului
MVI L, 00
ADI 08
MOV B, A           ; B conține adresa H de sfârșit a cipului
XRA A
CSC1: ADD M        ; calculul sumei de control
MOV C, A
INX H
MOV A, H
CMP B
MOV A, C
JNZ CSC1
POP B
RET

```

Deoarece cipul care conține programele de test reamplasează în poziția 1 (primul cip de BASIC), cipurile 1 și 2 se testează pe rând, amplasându-se în poziția 2. Eventualele erori care apar se afișează sub forma : număr cip, sumă de control din tabel (marilor), sumă de control calculată și o tratare a cauzei apariției erorii respective.

Pentru testarea logicii de citire a memoriei EPROM se face, citirea și calculul repetat al sumei de control pentru cipul care conține monitorul. O primă măsură care se poate lua în cazul apariției de erori este mișcarea cipurilor în soclu, după care se va executa din nou testul EPROM în regim manual.

14.3. Comanda K — testarea preluării de caractere de la tastatură

Deoarece în cadrul acestui test este necesară intervenția operatorului, testul se execută doar în regim manual. După afișarea mesajului de start test se așteaptă acționarea unei taste. Caracterul activat va fi pe o linie sau pe tot ecranul, funcție de comanda precedentă.

CTRL-L — rămâne memorată și este activată prin start test. Caracterul corespunzător tastei apăsată va fi afișat pe o linie, apăsarea tastei DEL are ca efect ștergerea ultimului rând, iar tasta BS repetă ultimul caracter tastat.

CTRL-C — rămâne memorată până la acționarea CTRL-L și implică afișarea caracterului selectat pe întregul ecran. Tasta DEL șterge ecranul, iar tastele BS și BLANK au același efect ca în regimul CTRL-L.

În ambele regimuri, acționarea tastei CR transferă controlul monitorului TEST aMIC. Testul se bazează pe observațiile operatorului. Se poate utiliza testul, prin menținerea pe ecran a caracterului H sau a caracterelor semigrafice pentru reglarea calității imaginii ecranului monitorului (dimensiunile rastrului, liniarizate, focalizare, afișare video-invers stabilă).

14.4. Comanda D — testarea afișării pe ecran

Acest test verifică funcționarea circuitelor de ceas și formatoare a semnalului video-complex, cit și a memoriei RAM din zona 4000-5FFFH rezervată afișării.

În primele 2 faze se afișează pe întregul ecran „table de șah” avînd dimensiunea pătratului elementar de una, respectiv patru, linii TV. Prima rețea se realizează prin înscrierea în RAM a octeților AAH și, respectiv, 55H iar a doua prin afișarea caracterelor semigrafice avînd codurile 68H și, respectiv, 6EH.

După durate egale de timp imaginile sînt negativate, ceea ce se realizează prin complementarea informației din RAM. Eventualele erori de înscriere în RAM la complementarea informației sînt sesizate ușor pe DISPLAY de către operator. O caracteristică a „tablei de șah” este egalitatea între nivelul de alb și de negru din imagine. Pentru testarea modulatorului s-au realizat succesiuni de imagini care conțin cantități diferite de alb și de negru.

Astfel, în a treia fază se afișează pătrățelele negre pe fond alb, utilizînd caracterele semigrafice (dimensiunea pătratului fiind de 8 linii TV), iar în a patra fază ecranul este șters (nivelul maxim de negru). Complementînd succesiv aceste imagini, nivelul de alb comută între 25% și 75% în primul caz și între 0% și 100% în al doilea caz. Pentru a obține stabilitatea imaginilor se fac reglaje asupra modulatorului, prin modificarea punctelor de funcționare a etajului modulator și modificarea raportului între semnalele de sincronizare și cel de imagine care concurează la realizarea semnalului video-complex (SVC).

Pentru a testa adresarea intercalată a memoriei RAM-DISPLAY se afișează benzi verticale de contrast, de grosime echivalentă cu 8 linii TV. Benzile se afișează de la stînga la dreapta și succesiv imaginea este complementată.

Pentru tastarea îndelungată a funcționării microcalculatorului, cu afișarea pe ecran, s-au realizat programe în limbaj mașină sau BASIC de trasare a unor figuri geometrice (pătrate și romburi înscrise, cercuri concentrice sau spirale). În continuare vor fi prezentate cîteva subprograme folosite pentru trasarea de figuri geometrice.

Considerîndu-se ecranul TV ca un pătrat cu latura de 256 linii TV și notînd axa orizontală cu Ox iar cea verticală cu Oy , subprogramele testează, setează sau șterg orice punct TV de coordonate $A(X, Y)$.

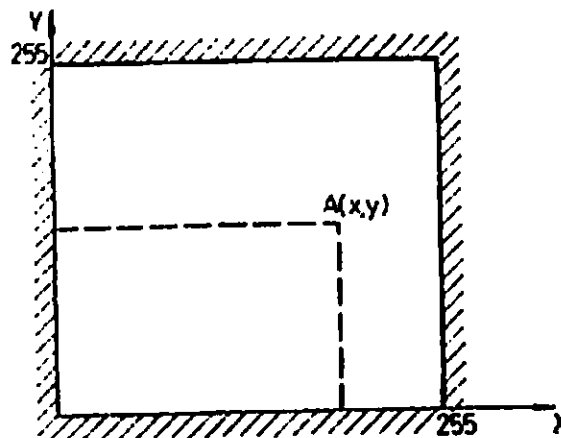


Fig. 14.1. Ecranul TV.

Subrutina ADR precizează poziția bitului în memoria RAM-DISPLAY, corespunzător punctului $A(X, Y)$.

Ca intrări se dau în registrele B și C respectiv coordonatele X și Y iar ca ieșiri se primește în registrul HL — adresa octetului și în acumulator poziția bitului căutat, codificată binar. Cunoșcând poziția bitului în memoria RAM, se pot opera asupra lui diferite acțiuni:

testare (TESTP), ștergere (RESETP) și punere pe „1” (SETP)

ADR :	PUSH DE	; Salvarea registrelor BC și DE
	PUSH BC	
	LD HL,3FE0H	; Adresa început zonă RAM-DISPLAY
	LD DE,0020H	minus 20H
	INC C	; DE= numărul de octeți corespunzător unei linii TV
A1 :	ADD HL, DE	
	DEC C	
	IP NZ, A1	; HL= adresa primului octet din rîndul TV căutat
	DEC HL	
A2 :	INC HL	
	LD A,B	
	SUB A,08H	
	LD B,A	
	JP NC A2	; HL conține adresa octetului căutat
	ADD A,08H ;	; A= conține nr. de ordine a bitului căutat în octet
	LD C,A	
	INC C	
	SCF	; CY= "1"
	XOR A	; A=00 ;
A3 :	RLA	
	DEC C	
	JR NZ,A3	; A= conține un singur "1" în poziția corespunzătoare
	POP BC	bitului căutat
	POP DE	
	RET	
SETP :	CALL ADR	; Punerea pe "1" a bitului adresat de registrele B și C.
	OR (HL)	
	LD (HL),A	
	RET	
RESETP :	CALL ADR	; Ștergerea bitului adresat de registrele B și C
	CPL	
	AND (HL)	
	LD (HL),A	
	RET	
TESP :	CALL ADR	; Testează bitul adresat
	AND (HL)	; Dacă este zero, flagul Z se poziționează pe 1.
	RET	

14.5. Comanda R — Testarea zonei de memorie RAM

14.5.1. Descrierea modurilor de lucru ale programului. După lansare calculatorul afișează mesajul :

TEST RAM

SWITCH : cerîndu-se 2 octeți pentru precizarea modului de lucru. În regim automat cei doi octeți care reprezintă SWITCH SOFT vor fi înscriși

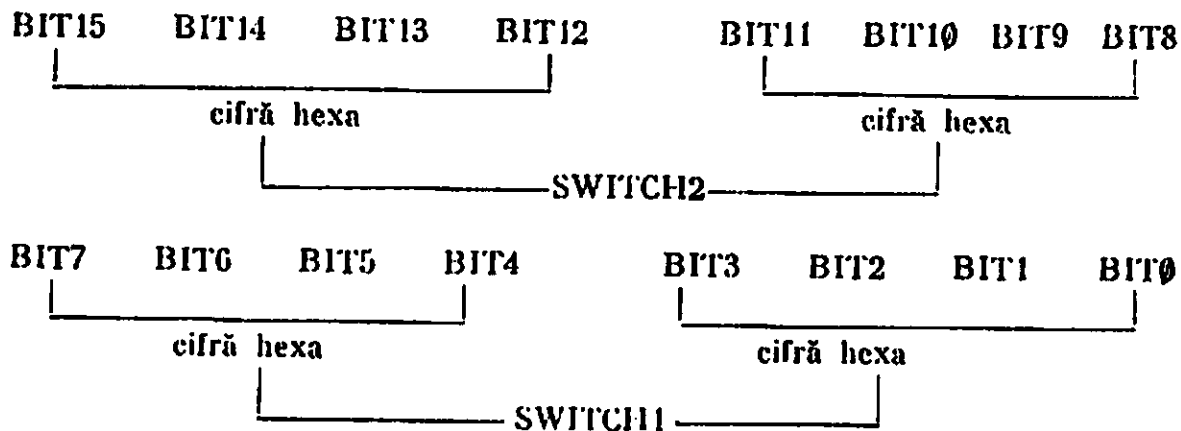
automat cu valorile : SWC1=01 ; SWC2=22 și vor fi tipăriți pe ecran. În regimul MANUAL vor fi preluați de la tastatură, biții lor având următoarea semnificație :

- BIT0 — 0 —
1 execută și testul complementar (subtestul adresare)
- BIT1 — 0 —
1 Oprire la eroare : continuă cu comanda Z.
- BIT2 — 0 —
1 Ciclează pe locația unde a apărut o eroare în textul SCCT
- BIT3 — 0 — (BIT 10 inefectiv)
1 — Inhibă tipărirea erorilor
- BIT4 — 0 —
1 — Ciclu pe subtestul definit de biții 6,7
- BIT5 — 0 —
1 — Inhibă tipărirea listei de erori după fiecare execuție a subtestului.
- BIT6,7 — Cod subtest
- BIT8 — 0 — Serie la începutul fiecărui subtest numărul corespunzător.
— 1 Oprire după fiecare subtest.
- BIT9 — 0
— 1 Inhibă execuția testului GALPAT
- BIT10 — 0
— 1 Tipărește numai prima eroare din modul când BIT3=1.
- BIT11 — 0
— 1 Execută subtestul PING PONG de 25 de ori
- BIT12 — 0
— 1 Inhibă tipărirea lui, END PASS **
- BIT13 — 0
— 1 Inhibă ciclul pe execuția testului RAM

Deci testul automat este caracterizat de următoarele :

- execuția nu se oprește la apariția unei erori ;
- execută subtestele în ordinea crescătoare a codului ;
- tipărește erorile și lista de erori ;
- inhibă erorile și lista de erori ;
- inhibă subtestul GALPAT ;
- execută subtestul PING PONG o singură dată ;
- inhibă ciclul pe execuția testului RAM

Cei 14 biți se introduc după mesajul 'SWITCH : ' (în regimul MANUAL) sub forma a 4 cifre hexa, astfel :



Ordinea de introducere este :

SWC2 (CR) SWC1 (CR).

Codificarea și ordinea de execuție în regim AUTOMAT sînt următoarele :

00— Adresare (ADR)
01— Scriere-citire (SCIT)
02— PING PONG
03— GALPAT

14.5.1. Descrierea modurilor de lucru ale programului roșu

Testul adresare : (ADR) constă în scrierea succesivă în pagini de memorie de câte 256 de octeți a unui contor cu valoarea 0-FFFF. La începutul fiecărei pagini contorul este dublu incremental astfel încît se poate identifica ușor fiecare pagină de memorie. Informația scrisă arată astfel :

C000	00	01	02	FF
C100	01	02	03	FE
C200	02	03	04	FD
:					

Testul se aplică pe zone compacte de memorie.

Facultativ se poate repeta testul, inversînd la scriere în memorie cei doi cuarței ai contorului.

Testul scriere-citire (SCIT) detectează biții de memorie blocați la zero/unu.

Se scrie succesiv în fiecare locație configurația inițială (de exemplu AA), apoi complementul ei și din nou configurația inițială, verificîndu-se de fiecare dată corectitudinea operației. Se aplică pe zone compacte de memorie.

Testele PING PONG și GALPAT operează la nivel de modul de 16 Ko (0, 4000H, 8000H, C000H). Aceste teste necesită o subrutină (SGP) care delimitează din zona compactă de memorie RAM limitele modulului de testat (ADINF respectiv ADSUP).

Testul GALPAT : detectează defecte de adresare, selecție multiplă, timp de acces. După ce se scrie în modul configurația inițială, se parcurge succesiv fiecare locație : se scrie în ea configurația complementară și se verifică toate locațiile modulului. Fiecare verificare este urmată de o verificare a locației curente. La sfîrșit se reface locația curentă. Se repetă testul pentru configurația complementară. Întrucît operarea cu fiecare locație implică verificarea întregului modul, testul durează de ordinul a citorva ore.

Testul PING-PONG : este asemănător cu testul GALPAT, dar optimizat ca durată. După ce modulul de memorie este înscris cu configurația inițială se parcurge succesiv fiecare locație înscrisu-se cu configurația complementară. Pentru fiecare locație curentă se verifică 14 locații, fiecare verificare fiind urmată de verificarea locației curente. Adresele acestor locații se obțin pornind de la adresa locației curente, prin complementarea câte unui bit de adresă.

În continuare se prezintă programul care execută acest test. Subrutina SGP apelată anterior a pregătit adresele limită (ADINF, ADSVP) pentru modulele testabile.

```

PING PONG:  CAL INIT           ; inițializare
P1:        CALL CYBR          ; copiază cuvânt inițial între (ADINF, ADSVP)
           LDA ADINF
           MOV D,A            ; (D,E) — adresa cuvântului test
           MVI E,00
P2:        LDAX D
           XRA B              ; Se verifică dacă în adresa cuvântului de test s-a
           JZ P21             înscris configurația inițială.
           PUSH B
           PUSH D
           CALL RWERR         ; subrutina de controlizare și tratare a erorilor
           POP D
           POP B
P21:       MOV A,C            ; Se înscris în locația test configurația complementară
           STAX D
           LDA ADINF
           CPI A0
           JZ P7              ; Se stabilește numărul de biți care vor fi complemen-
           MVI H,15H         tați din adresa cuvântului test (pentru un modul întreg
           JMP P8             se completează 14 biți)
P7:        MVI H,14
P8:        MOV A,H
           SUI 00H
           STA PNG
           MVI L,00           ; L — masca pentru complementarea unui bit din
           PUSH H             adresă
           MOV H,D
           MOV L,E           ; Adresa cuvântului test în stivă iar în H și L conținut
P9:        XT HL             și masca pentru complementare.
           DCRH
           JZ P6
           MOV A,L
           RLC                ; masca devine L=01H
           MOV L,A
           LDA PNG
           CMP H
           MOV A,L
           HTHLL              ; Masca și conținutul în stivă
           JNC P4             ; salt dacă PNG > conținutul de biți
           XRA L
           MOV L,A            ; complementarea bitului din L
           JMP P5
P4:        XRA H              ; complementarea bitului din H
           MOV H,A
P5:        LDA ADMAX         ; Adresa sfârșit RAM
           DCR A
           CMP H
           JC P9              ; salt dacă H > HMAX.
           MOV A,M
           XRA B              ; locația derivată se compară cu configurația inițială.
           JZ P9
           PUSH B
           PUSH D
           CALL RWERR.

```

```

POP D
POP B
P9: LDAX D      ; se compară cuvîntul test cu configurația complementată
XRA C
JZ P3      ; dacă nu este eroare salt la complementarea următorului
PUSH B     bit de adresă
PUSH D
MOV B,C
CALL RWERR
POP D
POP B
JMP P3
P6:  POP H      ; S-a terminat un ciclu de verificare
MOV A,B
STAX D     ; Rescriere cuvînt inițial la adresa
INX D     cuvîntului de test
LDA ADSUP;
CMP D
JNZ P2     ; Dacă adresa următoare ≠ ADSUP se închide bucla
MOV A,B    externă
MOV B,C    ; schimbă cuvîntul inițial cu complementarul său
MOV C,A
LDA VARPR1]
XRI 20     ; Se testează și modifică ITEST
STA VARPR1
ANI 20     ; Dacă ITEST=1 se execută testul și pentru configurația
JNZ P1     complementară
RET

```

14.5.2. Organizarea testului RAM. În figura 14.2 este reprezentată schematic organigrama testului RAM. După inițializări și mesaje de început test, se cere precizarea octeților SWC2, SWC1 în regimul manual, iar în regimul automat ei sînt afișați, fiind determinați interior.

În regimul automat zonele de test RAM au fost precizate la intrarea în programul TEST aMIC. În funcție de SWC4, se execută testele în ordinea crescătoare a codurilor sau se ciclază pe testul dat de SWC67.

Codul testului curent este memorat și, pe baza lui, se creează adresa de salt la subtest și codul eventualelor erori. Dacă SWC8=0 atunci la începerea unui subtest se afișează codul subtestului curent, informînd prin aceasta că testul anterior s-a terminat (în acest regim programul nu se oprește după subtest). După execuția subtestului, dacă SWC8=1 se dă mesaj: END** și se pune la dispoziția operatorului. Comanda CTRL-G va executa listarea erorilor și predă controlul monitorului TEST aMIC. Dacă SWC5=1 se imbibă listarea erorilor și continuă testul. În cazul în care subtestul terminat a fost GALPAT, se afișează sau nu END PASS**—funcție de SWC12. Afișarea este utilă în cazul în care se ciclază pe testul RAM. Bitul SWC11 decide dacă subtestul PING PONG se face o singură dată sau de 25 de ori. Se observă din organigramă complexitatea regimurilor de lucru a programului TEST RAM.

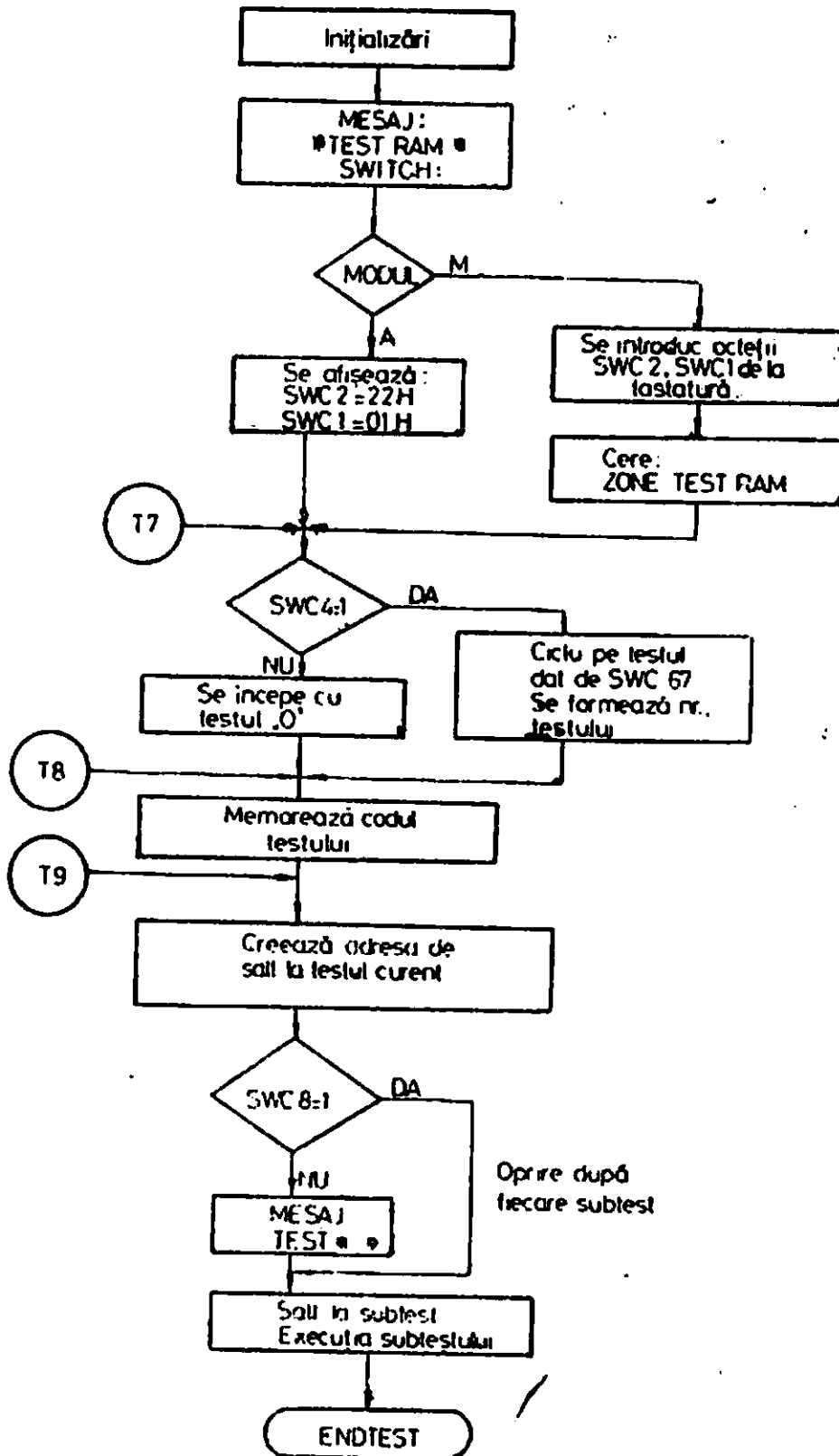


Fig. 14.2. Organigrama testului RAM.

Semnificația comenzilor pe parcursul testului este următoarea :

CTRL-Z — repornire după subtest sau după oprire la eroare ;

CTRL-G — tipărește lista de erori la sfârșitul subtestului curent în regim manual ;

CTRL-G — oprire TEST RAM după subtestul curent.

14.5.3. Modul de tratare a erorilor. Codul erorilor apărute în cursul testelor (ERR) indică în care subtest a apărut acea eroare :

01 — Eroare de R/W în testul de adresare (probabil pinii de adresă defecti)

02 — Eroare de R/W în testul SCRIE-CITEȘTE

03 — Eroare de R/W în testul PING-PONG

04 — Eroare de R/W în testul GALPAT

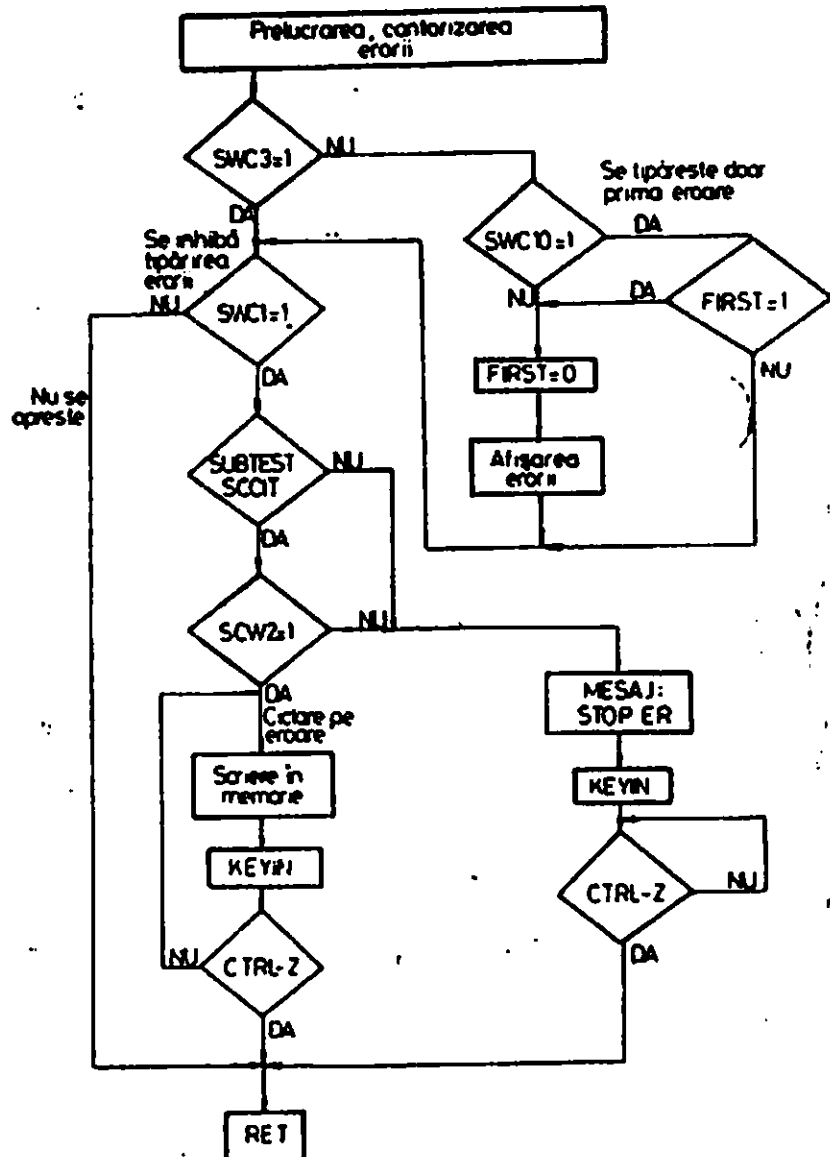
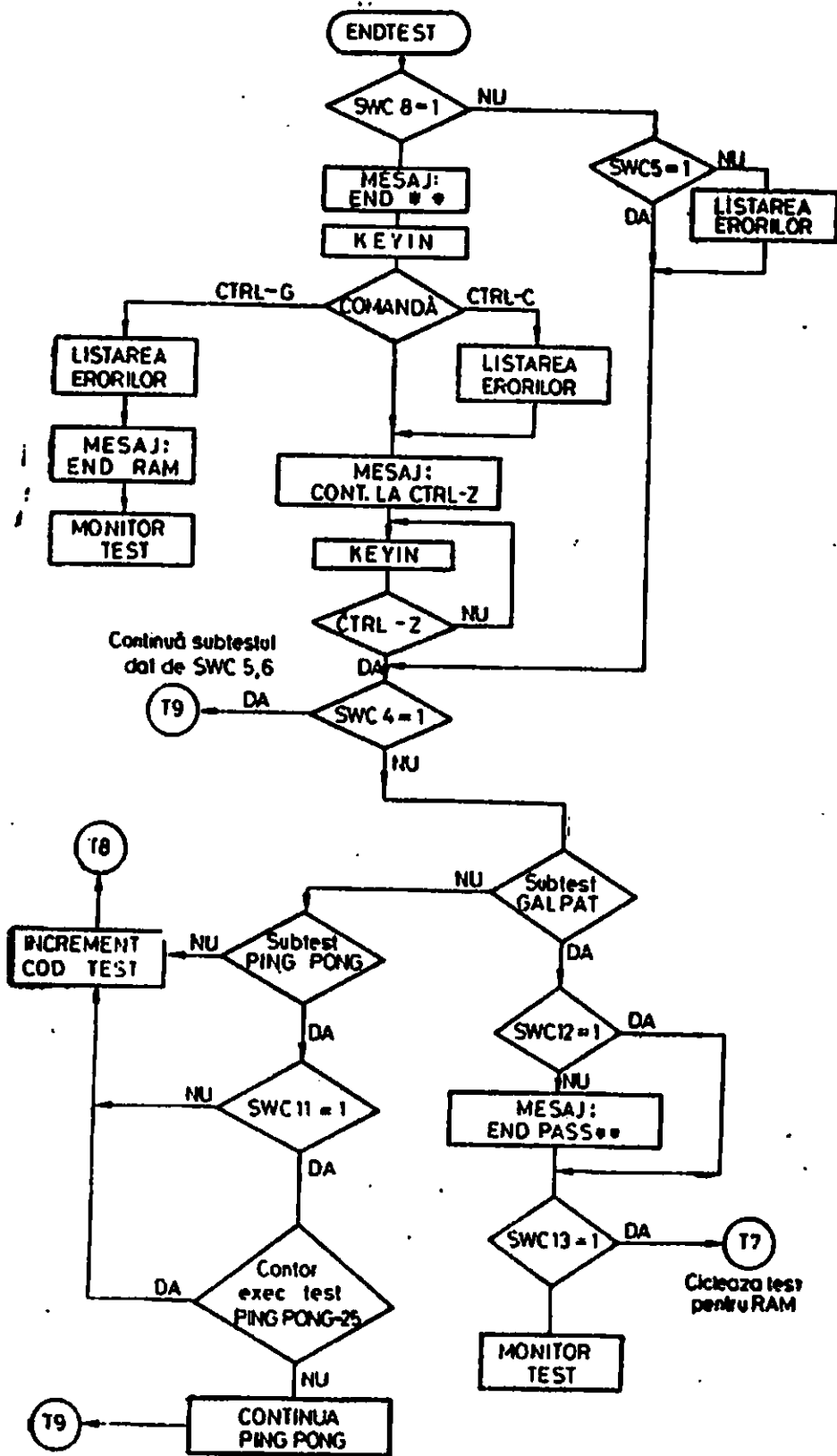


Fig. 14.3. Organigrama



testării erorilor < 2.

Mesajul tipărit la apariția unei erori are următoarea semnificație :

ERR : — Codul erorii apărute
 RDATA : — Data citită
 WDATA : — Data scrisă
 PC : — Valoarea PC la apelul rutinei de eroare
 ADDR : Adresa de memorie unde a apărut eroarea
 MODUL : Numărul modulului

Fiecare modul de memorie RAM (16K) are o zonă contoare distinctă, cuprinzând :

- un octet în care este înscris numărul modulului ;
- 8 contoare pentru erorile de R/W, câte unul pentru fiecare bit ;
- un octet depășire, indicând biți pentru care numărul de erori > 256.

Fiecare contor ocupă un octet, deci pot fi memorate distinct 256 erori. Pentru un număr mai mare de erori se afișează caracterul '*' în dreapta ecranului respectiv. Memoria RAM utilizată pentru DISPLAY nu este testată de către acest program, deoarece ea este testată în cadrul testului DISPLAY.

În figura 14.3 este prezentată organigrama de tratare a erorilor și de riprire la eroare.

După contorizarea erorii și completarea bufferului de afișare, este testat bitul SWC3 care dacă este poziționat pe „1” permite afișarea erorii. Dacă bitul SWC10=„1”, eroarea va fi afișată numai dacă este la prima apariție. Înainte de afișarea erorii, variabila FTRST este anulată.

În funcție de valoarea bitului SWC1, programul de testare va fi oprit sau nu. Subtestul SCRIERE-CITIRE permite buclarea pe locația eronată, dacă valoarea bitului SGW2=„1”. După retestarea locației, programul așteaptă comandă de la operator. Dacă aceasta este CTRL-Z, testarea locației eronate este reluată.

Pentru celelalte subteste și pentru subtestul SCCIT cu bitul SGW2=0, se va afișa mesajul

STOP_ER.

după care se pune la dispoziția operatorului. Se poate continua textul cu comanda CTRL-Z, încheind sub rutina de tratare a erorilor.

14.6. Testarea transferului de informații dinspre/spre casetofon

Pentru faza de punere la punct a interfeței cu casetofonul se comandă înregistrări cu informație constantă 00H, FFH și apoi AAH urmărindu-se cu osciloscopul forma și nivelul semnalului la intrarea în casetofon. La fel se urmărește și redarea semnalului la intrarea în interfața programabilă internă. După ce se fixează nivelul optim al volumului la înregistrare și redare, se fac înregistrări lungi (până la 44 Kocteti), cu informație constantă sau aleatoare, sau cu blocuri de date în care fiecare octet este identic cu octetul mai puțin semnificativ de adresă, corespunzător. Se execută de câteva ori încărcarea acestor

fișiere în memoria calculatorului și se verifică încărcarea atât prin suma de control a fișierului cit și prin programe de verificare a conținutului fișierului încărcat. Aceleași încercări se fac și sub controlul interpretorului BASIC, cu ajutorul instrucțiunilor LOAD și SAVE.

14.7. Procedura de test a interpretorului BASIC

Procedura de test a interpretorului BASIC constă în reluarea unui set de programe în care apar toate instrucțiunile specifice acestuia.

Aceste programe de test, scurte, trebuie alese astfel încât să solicite toate posibilitățile limbajului. Procedura de verificare a bunei funcționări a interpretorului BASIC 14 K, constă în rularea de programe de test scrise în limbajul BASIC. Aceste programe utilizează pe cât posibil câte o instrucțiune sau o funcție BASIC în așa fel încât se cunoaște aprioric rezultatul, sau domeniul de rezultate. După rulare se verifică prin comparare dacă s-a ajuns la rezultatul așteptat.

În continuare se prezintă câteva programe de acest tip (v. și cap. 9).

14.7.1. Verificarea funcțiilor standard

Funcțiile standard rezolvate în interpretorul BASIC sînt : sinus, cosinus, tangenta, arctangenta, logaritmul, funcția exponențială, radical, ridicare la putere.

Se știe că :

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

Se verifică dacă pentru un X dat, funcțiile BASIC satisfac această egalitate.

```
10 X=0
20 Y=TAN(X)
30 Z=SIN(X)/COS(X)
40 PRINT Y; " = "; Z
50 STOP
```

Se verifică egalitățile următoare :

$$\sin^2 x + \cos^2 x = 1$$

$$\operatorname{tg}(\operatorname{arc} \operatorname{tg}(x)) = x$$

$$e^{\ln x} = x$$

```
10 FOR X=0 TO 10
20 PRINT X, SIN(X)^2 + COS(X)^2
30 NEXT X
40 STOP
```

```
10 FOR X=-10 TO 10
20 PRINT X, TAN(ATN(X))
30 NEXT X
40 STOP
```

```

10 FOR X=1 TO 10
20 PRINT X, EXP (LOG (X))
30 NEXT X
40 STOP

```

Se calculează rădăcina pătrată dintr-un număr dat de utilizator prin formula interactivă :

$$y_{i+1} = \frac{1}{2} \left(y_i + \frac{A}{y_i} \right)$$

unde A este numărul introdus ;

y_i este o valoare arbitrară inițială

Iterația se oprește când $(y_{i+1} - y_i) = 0$. Se tipărește valoarea astfel obținută. Se tipărește apoi și valoarea obținută prin funcția BASIC de rădăcină pătrată (SQR) și se compară cele două rezultate astfel obținute.

```

10 REM "RADACINA PATRATA"
20 PRINT "DATI NUMARUL"
30 INPUT A
40 B=INT (RND (X)*A)+1
50 REM "B=Y ARBITRAR ALES"
55 REM "IN FUNCTIE DE A"
60 Y=0.5*(B+A/B)
70 IF B=Y THEN 100
80 B=Y
90 GOTO 60
100 PRINT "RADACINA PATRATA" ;
105 PRINT "PRIN ITERATIE:" ; Y
110 X=SQR (A)
120 PRINT "RADACINA PATRATA" ;
125 PRINT "PRIN FUNCTIE SQR:" ; X
130 Z=ABS (X-Y)
140 PRINT "DIFERENTA:" ; Z
150 STOP

```

Funcția internă de generare numere aleatoare (RANDOM), pune la dispoziția utilizatorului câte un număr aleator în domeniul (0, 1). Se generează tabele de câte 10 numere variabile în diferite domenii, înmulțind numărul aleator cu diferite constante și/sau luând valoarea întreagă cu funcția INTEGER.

```

10 REM "VERIFICARE RANDOM"
20 PRINT "VALORI REALE INTRE 0 SI 1"
30 FOR I=1 TO 10
40 PRINT RND (X),
50 NEXT I
60 PRINT "VALORI REALE INTRE 0, 10 SI 0, 100"
70 FOR I=1 TO 10
80 PRINT RND (X)*10 ; RND (X)*100
90 NEXT I
100 PRINT "VALORI INTREGI INTRE 0, 10 SI 0, 100"
110 FOR I=1 TO 10
120 PRINT INT (RND (X)*10) ; INT (RND (X)*100)
130 NEXT I

```

```

140 PRINT "VALORI INTREGI INTRE 0, 6 ŞI 10, 20"
150 FOR I=1 TO 10
  160 PRINT INT (RND (X)*6) : INT (RND (X) * 10+10)
170 NEXT I
180 STOP

```

Prin funcția PLOT se poate aprinde un punct pe ecranul TV, considerînd ecranul de 64*64 puncte. Se trasează curba sinus cu funcția PLOT și se șterge cu funcția UNPLOT.

```

10 INIT P
20 FOR N=1 TO 64
30 PLOT N, 22+20*SIN (N/32*PI)
40 NEXT N
50 FOR N=1 TO 64
60 UNPLOT N, 22+20*SIN (N/32*PI)
70 NEXT N
80 STOP

```

Funcția RANDOM se poate verifica și cu funcția PLOT. Se umple ecranul aleator prin funcțiile RANDOM și PLOT cu puncte aprinse. Dacă această variație aleatoare umple relativ uniform ecranul, funcția RANDOM este aleatoare.

```

10 INIT P
20 PLOT INT (RND (X)*64)+1, INT (RND (X)*64+1)
30 GOTO 20
40 END

```

Funcția PLOT aprinde un punct pe ecran și UNPLOT stinge punctul pe ecran. Se poate astfel aprinde și stinge același punct obținînd astfel o pulsație a unui punct.

```

10 INIT P
20 X=RND (X)
30 Y=RND (Y)
40 FOR I=1 TO 10
50 PLOT INT (X*64)+1, INT (Y*64)+1
60 UNPLOT INT (X*64)+1, INT (Y*64)+1
70 NEXT I
80 STOP

```

Funcția PRINT AT (X, Y) permite afișarea pe ecran, la coordonate date, a unei informații. Ecranul este văzut ca formînd 32 de linii a cîte 30 de caractere. În instrucțiune, X este linia și Y coloana de unde se va începe afișarea.

Se afișează pe ecran un tabel de formă cunoscută.

```

10 INIT P
20 PRINT AT (2, 2) : "TABEL CU NUMERE" :
25 PRINT "DE TELEFOANE"
30 PRINT AT (3, 2) : S$

```

```

40 PRINT AT (6, 5); "NUME 1 PREFIX 1";
45 PRINT "NR. TEL."
50 PRINT AT (7, 2); S$
60 FOR I=8 TO 16 STEP 2
70 READ N$(10)
80 READ N
100 PRINT AT (I, 2); N$; "1"; P; "1"; N
110 PRINT AT (I+1, 2); S$
120 NEXT I
130 STOP
140 DATA "NUME 1", 971, 12345
150 DATA "NUME 2", 00, 457321
160 DATA .....
.
.
.
200 DATA ...
210 S$="....."
220 END

```

Aprinderea și stingerea unei zone prin instrucția PRINT AT (X, Y)

```

10 INIT P...
20 X=INT(RND(X)*64)+1
30 Y=INT(RND(Y)*64)+1
40 FOR I=1 TO 20
50 PRINT AT (X, Y); " "
60 PRINT AT (X, Y); " "
70 NEXT I
80 STOP

```

Caracterul care aparține la o valoare A se obține prin CHR\$(A). Setul de caractere se afișează prin următorul program.

```

10 PRINT "SETUL DE CARACTERE"
20 FOR A=0 TO 255
30 PRINT A; CHR$(A)
40 NEXT A
50 STOP

```

Funcția INKEY\$ permite utilizatorului să dea unei variabile șir valoarea tastei apăsată. Următorul program permite utilizarea aMIC-ului ca mașină de scris. Apăsarea tastei RETURN poziționează pe linie nouă.

```

10 INITP
20 A$=INKEY$
30 B$=INKEY$
40 IF A$=B$ THEN
50 REM "S-A TASTAT UN CARACTER"
60 PRINT B$
70 GOTO 20
80 END

```

Valoarea variabilei șir se obține prin funcția VAL.

Dacă la linia 20 schimbăm expresia și la linia 30 valoarea lui X, putem efectua verificări pentru diferite expresii și valori ale lui X.

```

10 PRINT "VALOAREA EXPRESIEI"
20 F$ = "2X + 2 - X + 1";
30 FOR X=1 TO 5
40 T=VAL(F$)
50 PRINT "EXPRESIA :"; F$.
60 PRINT "PENTRU X="; X;
70 PRINT "ARE VALOAREA "; T
80 NEXT X
90 STOP

```

Se permit în Interpretorul BASIC trei bucle suprapuse. În următorul program există trei bucle K, J și I. Totodată se lucrează cu instrucțiuni matriciale.

```

10 PRINT "ORDONAREA UNUI SIR"
20 READ N
30 DIM A(N)
40 MAT READ A
50 FOR I=1 TO N-1
60 FOR J=I+1 TO N
70 IF A(I)<A(J) THEN 130
80 B=A(J)
90 FOR K=J TO I+1 STEP -1
100 A(K)=A(K-1)
110 NEXT K
120 A(I)=B
130 NEXT J
140 NEXT I
150 PRINT "ȘIRUL ORDONAT"
160 MAT PRINT A
170 STOP
180 DATA 5
190 DATA 10, 30, 5, 15, 5
200 END

```

În funcție de valoarea unui număr se pot executa diferite secvențe de program cu instrucțiunea ON. Pentru a verifica această instrucțiune se tastează un număr X de utilizator și dacă pe ecran apare „ați bătut X” atunci funcționează corect.

```

10 REM "VERIFICARE ON — GOTO"
20 PRINT "TASTATI UN NUMĂR"
30 INPUT M
40 ON M GOTO 80, 100, 120, 140, 160, 180, 200, 220, 240
50 PRINT "ATI TASTAT 0"
60 GOTO 20
80 PRINT "ATI TASTAT 1"
90 GOTO 20
100 PRINT "ATI TASTAT 2"
110 GOTO 20
120 PRINT "ATI TASTAT 3"
130 GOTO 20

```

```

140 PRINT "ATI TASTAT 4"
150 GOTO 20
160 PRINT "ATI TASTAT 5"
170 GOTO 20
180 PRINT "ATI TASTAT 6"
190 GOTO 20
200 PRINT "ATI TASTAT 7"
210 GOTO 20
220 PRINT "ATI TASTAT 8"
230 GOTO 20
240 PRINT "ATI TASTAT 9"
250 GOTO 20
260 END

```

Utilizarea instrucțiunilor READ, MATREAD, RESTORE se exemplifică în următorul program. A se observa că nu se ajunge niciodată la citirea valorilor 3 din blocul de date.

```

10 REM "DATA, READ, MAT, READ" [E]
20 READ A, B, C
30 PRINT A, B, C
40 READ A, B, C
50 PRINT A, B, C
60 RESTORE
70 MAT READ D (6)
80 MAT PRINT D
90 STOP
100 DATA 1, 1, 1, 2, 2, 2, 3, 3, 3
110 END

```

Următorul program utilizează instrucțiunile cu șiruri de caractere.

```

10 PRINT "SUPRAFATA"
15 PRINT "DREPTUNGHI, TRIUNGHI, CERC"
20 INPUT S$
30 IF S$ = "DREPTUNGHI" THEN 70
40 IF S$ = "TRIUNGHI" THEN 110
50 IF S$ = "CERC" THEN 160
60 GOTO 20
70 PRINT "DATI LUNGIMEA SI LATIMEA"
80 INPUT B, C
90 Z=B*C
100 GOTO 190
110 PRINT "DATI LUNGIMILE LATURILOR"
120 INPUT A, B, C
130 S=(A+B+C)/2
140 Z=SQR(S*(S-A)*(S-B)*(S-C))
150 GOTO 190
160 PRINT "DATI RAZA"
170 INPUT R
180 Z=PI*R^2
190 PRINT "SUPRAFATA"; S$; "="; Z
200 GOTO 10
210 END

```

Apelul de subprogram scris în limbaj BASIC se face prin GOSUB nr. Prima instrucțiune RETURN întâlnită provoacă revenirea la instrucțiunea

următoare instrucțiunii GOSUB executate ultima dată. În următorul program se verifică unele posibilități de salturi la subprograme în BASIC. Primul RETURN întâlnit la care nu există pereche GOSUB provoacă oprirea execuției în continuare a programului.

```

20 REM "UTILIZĂRI GOSUB ȘI RETURN"
25 REM "APEL SPR. LA PRIMA INTRARE"
30 GOSUB 200
35 PRINT "REVINE DIN SPR 200"
40 REM "APEL SPR. INTERIOR"
50 GOSUB 210
60 PRINT "REVINE DIN SPR 210"
55 REM "APEL SPR. 300 REVINE CU GOTO"
60 GOSUB 300
65 PRINT "REVINE AICI PRIN RETURN URMĂTOR"
70 PRINT "REVINE PRIN GOTO DIN SPR 300"
75 REM "URM. RETURN E PRIMUL ÎNTILNIT"
80 REM "DUPĂ GOSUB 300, REVINE DUPĂ EL"
85 RETURN
90 REM "AICI NU SE AJUNGE"
95 STOP
200 PRINT "INTRARE SPR. 200"
210 PRINT "CORPUL SPR. 200"
220 RETURN
300 PRINT "INTRARE SPR. 300"
310 GOTO 70
400 END

```

Pe ecran apare deci :

- Intrare spr. 200
- Corpul spr. 200
- Revine din spr. 200
- Corpul spr. 210
- Revine din spr. 210
- Intrare spr. 300
- Revine prin GOTO din spr. 300
- Revine aici prin RETURN următor.

Utilizarea subprogramelor scrise în limbaj mașină e posibilă prin instrucțiunea CALL. Următorul program în limbaj BASIC cheamă primul subprogram în cod mașină.

```

10 REM "CALL"
20 CALL (1)
30 STOP

```

Subprogramul scris în limbaj mașină se introduce în memoria calculatorului decodificat în hexazecimal, prin funcția S (substituție) din monitor.

Se dorește introducerea unui subprogram care să genereze în mijlocul ecranului caracterul A, diferit de caracterul A dat de generatorul de caractere al monitorului. Se cheamă subprogramul de inițializare ecran al monitorului și subprogramul de scriere caracter.

```

CALL INITV
LXI H, (ADR MIJLOC ECRAN TV)

```



```

                LXI D, (ADR GENERATOR A PROPRIU)
                MVI A, 6
                CALL WR26
                RET
CARGEN         DB 6, 9, 9, F, 9, 9,

```

Se va folosi următoarea secvență de substituire S :

```

6400          01           ; NR. SUBPROGRAM
              FF 66       ;ADR SPR 1
              FF         ; SFIRSIT TABEL SPR
60FF          CD A4 02    ; CALL INITV
              21 0A 50    ; LXI H, 500A
              11 OE 67    ; LXI D, CARGEN
              3E 06       ; MVI A, 6
              CD 05 02    ; CALL WR26
              C9         ; RET
670E          06 09 09    ; CARGEN PT. A
              0F 09 09

```

Pentru verificarea funcțiilor grafice MOVE, DRAW se dau în continuare câteva curbe specifice :

a) O serie de curbe sinusoidale

```

20 INIT P
30 X=0
40 Y=50
50 Z=X
60 MOVE X, Y
70 FOR I=0 TO 3*PI STEP PI/10
80 DRAW Z, SIN (I)*25+Y
90 Z=Z+2
100 NEXT I
110 X=X+2
120 Y=Y+2
130 IF X <=-26 THEN 50
140 STOP

```

b) O serie de cercuri care formează un con. A, B — coordonatele centrului primului cerc.

```

20 READ A, B
30 READ X, Y
40 INIT P
50 FOR T=1 TO 25
60 MOVE A+T, B
70 FOR I=1 TO 2*PI STEP PI/10
80 DRAW A+T*COS (I), B+T*SIN (I)
90 NEXT I
100 A=A+X
110 B=B+Y
120 NEXT T
130 STOP
140 DATA 10, 10, 5, 5
150 END

```

c) Linii de lungimi variabile din cele patru colțuri ale ecranului

```

10 INIT P
20 MOVE 0, 0
30 DRAW RND (X)*100, RND (X)*100
40 MOVE 100, 100
50 DRAW 100-RND (X)*100, 100-RND (X)*100
60 MOVE 0, 100
70 DRAW RND (X)*100, 100-RND (X)*100
80 MOVE 100, 0
90 DRAW 100-RND (X)*100, RND (X)*100
100 GOTO 20
110 END

```

Trasarea unei figuri cu instrucțiunile BASIC RMOVE și RDRAW și rotirea figurii cu ROTATE. Se trasează în triunghi și se rotește acest triunghi.

```

10 INIT P
20 WINDOW -50, 50, -10, 50
25 MOVE 0, 0
30 FOR U=0 TO 2*PI STEP PI/10
35 ROTATE U
40 RDRAW 10, 0
45 RDRAW 10, 10
50 RDRAW -20, -10
65 NEXT U
60 STOP

```



Verificarea funcției WINDOW și VIEWPORT. Se trasează un cerc de o mărime dată la diferite valori pentru WINDOW și VIEWPORT. Se observă că trebuie aleasă mărimea și zona ecranului în funcție de mărimea figurii

```

10 INIT P
20 REM "WINDOW SI VIEWPORT IMPLICIT"
25 REM "0, 100, 0, 100"
30 REM "APARE PE ECRAN 1/4 CERC"
40 GOSUB 200
50 INIT P
60 WINDOW -50, 50, -50, 50
70 REM "VIEWPORT IMPLICIT"
75 REM "0, 100, 0, 100"
80 REM "APARE TOT CERCUL IN MIJLOC ECRAN"
90 GOSUB 200
100 INIT P
110 WINDOW -50, 50, -50, 50
120 VIEWPORT 50, 100, 50, 100
130 REM "APARE CERCUL IN DREAPTA SUS"
140 GOSUB 200
150 STOP
200 MOVE 25, 0
210 FOR I=0 TO 2*PI STEP PI/10
220 DRAW 25*COS (I), 25*SIN (I)
230 NEXT I
240 FOR I=1 TO 50
250 REM "TEMPORIZARE"

```

```
260 NEXT I
270 RETURN
280 END
```

Trasări de pătrate care se măresc ; in video normal și video invers.

```
20 INIT P
30 PUT (34)=128
40 GOSUB 100
50 INIT P
60 PUT (34)=160
70 GOSUB 100
80 GOTO 20
100 MOVE 50, 50
110 FOR L=10 TO 100 STEP 10
120 MOVE 50-L/2, 50-L/2
130 RDRAW L, 0
140 RDRAW 0, L
150 RDRAW -L, 0
160 RDRAW 0, -L
170 NEXT L
180 RETURN
190 END
```

Bateria de teste, odată trecută, ne dă posibilitatea de a opera cu încredere cu exemplarul de AMIG.

COLECȚIE DE PROGRAME PENTRU REZOLVAREA UNOR PROBLEME DE MATEMATICĂ DIN MATERIILE CLASELOR A IX-A ȘI A X-A

Proiectarea și realizarea microcalculatoarelor individuale în țara noastră creează premisele necesare introducerii lor în învățământul liceal, în cadrul laboratoarelor de matematică, fizică, biologie, etc.

Trebuie amintit faptul că în acest an a intrat în fabricație de serie, la Întreprinderea de memorii electronice din Timișoara, microcalculatorul AMIQ.

Ministerul Educației și Învățământului a comandat circa o sută exemplare, care vor intra în acest an în dotarea unor instituții de învățământ superior. Se prevede ca aceluia de dotare să se extindă și la nivelul liceelor.

Între timp, specialiștii în tehnica de calcul din țara noastră au proiectat noi tipuri de microcalculatoare individuale, dintre care se amintesc: HQ-85, PRAE și DEGA-200.

Interesant este microsistemul HQ-85, care a fost proiectat la Catedra de calculatoare din Institutul Politehnic București și care va intra în curând în fabricație de serie la Întreprinderea de Calculatoare Electronice.

Arând în vedere performanțele acestor calculatoare legate, de implementarea unor limbaje evoluate (BASIC, PASCAL, FORTRAN, MICROPROLOG, FORTH, etc.), posibilitatea de folosire a unui ecran color pentru vizualizare, cât și faptul că ele vor putea intra în dotarea liceelor, în cele ce urmează se va prezenta o colecție de programe în BASIC, pentru asistarea studiului unor capitole de matematică din materia claselor a IX-a și a X-a.

1. PROGRAME PENTRU CALCULE CU POLINOAME

Se prezintă programe pentru :

- împărțirea unui polinom de un grad oarecare n cu polinoame de gradul întâi și doi, ireductibile și unitare în \mathbb{R} , adică de forma $x + a$ și $x^2 + px + q$.
- calculul valorii unui polinom,
- calculul rădăcinii reale a unui polinom într-un interval dat, știind că la capetele intervalului $[a, b]$, polinomul la valori de semne contrare și că în acest interval funcția polinomială este strict monotonă.

1.1. PROGRAM PENTRU ÎMPĂRȚIREA UNUI POLINOM CU UN BINOM

Fie polinomul :

$$\sum_{i=0}^n a_i x^{n-i} \text{ și binomul : } x + a.$$

Considerând citul de forma : $\sum_{i=1}^n b_i x^{n-i}$ și restul de forma : b_{n+1} rezultă următoarele :

$$\sum_{i=0}^n a_i x^{n-i} = \left(\sum_{i=0}^n b_i x^{n-i} \right) \cdot (x + a) + b_{n+1} \quad \text{sau}$$

$$\sum_{i=0}^n a_i x^{n-i} = b_0 x^{n+1} + \sum_{i=0}^n (b_{i+1} + a b_i) \cdot x^{n-i}.$$

Prin identificare se obțin următoarele relații :

$$b_0 = 0$$

$$a_i = b_{i+1} + a \cdot b_i, \text{ pentru } i=0, 1, 2, \dots, n$$

sau :

$$b_{i+1} = a_i - a \cdot b_i,$$

și restul :

$$b_{n+1} = a_n - a \cdot b_n,$$

```

20 PRINT "introduceți gradul n al polinomului"
30 INPUT n
40 PRINT "n="; n
50 PRINT "introduceți coeficienții polinomului SUMA"
(a(i)*x^(n-1))
60 DIM a (2*n)
65 DIM b (2*n)
70 FOR i=0 to n
80 PRINT "a (" ; i ; ")="
85 INPUT a (i+1)
90 PRINT a (i+1)
100 NEXT i
110 PRINT "introduceți termenul liber a al divizorului"
120 INPUT a
130 PRINT "a="; a
200 REM calculul cîtului
210 b (1)=0
220 FOR i=1 TO n+1
230 LET b (i+1)=a (i)-a*b (i)
240 NEXT i
300 REM afisare rezultate
310 FOR i=1 TO n
320 PRINT "b (" ; i ; ")="; b (i+1)
330 NEXT i
340 PRINT "r="; b (n+2)
350 END

```

1.2. PROGRAM PENTRU ÎMPĂRȚIREA UNUI POLINOM DE GRADUL n CU TRINOMUL $x^2 + px + q$.

Se consideră :

– polinomul de forma : $\sum_{i=0}^n a_i x^{n-i}$

– cîtul de forma : $\sum_{i=0}^n b_i x^{n-i}$

– restul de forma $rx + s$,

prin identificare, din relația :

$$\sum_{i=0}^n a_i x^{n-i} = \left(\sum_{i=0}^n b_i x^{n-i} \right) \cdot (x^2 + px + q) + rx + s$$

se obțin următoarele rezultate :

$$\begin{aligned} b_0 &= 0 & r &= b_{n+1} \\ b_1 &= 1 & s &= b_{n+2} + pb_{n+1} \\ b_{i+2} &= a_i - pb_{i+1} - qb_i \end{aligned}$$

```

10 REM împărțirea unui polinom de gradul n cu un trinom
20 PRINT "introduceți gradul n al polinomului"
30 INPUT n
40 PRINT "n="; n
50 PRINT "introduceți coeficienții polinomului SUMA (a (i) * x ^ (n-1))"
60 DIMA (2*n)
65 DIMB (2*n)
70 FOR i=0 TO n
80 INPUT a (i+1)
90 PRINT "a (" ; i ; ")="; a (i+1)
100 NEXT i
200 PRINT "introduceți coeficienții divizorului x^2 + p.x + q"
210 PRINT "introduceți p"
220 INPUT P
225 PRINT "p="; P
230 PRINT "introduceți q"
240 INPUT q
250 PRINT "q="; q
300 REM calculul coeficienților cîtului

```

```

310 LET b(1)=0
320 LET b(2)=0
330 FOR i=1 TO n+1
340 LET b(i+2)=a(i)-p*b(i+1)-q*b(i)
350 NEXT i
400 REM afisare rezultate
410 FOR i=2 TO n
420 PRINT "b (" ; i ; ")=" ; b(i+1)
430 NEXT i
440 PRINT "r=" ; b(n+2)
450 PRINT "s=" ; b(n+3)+p*b(n+2)
500 END
    
```

1.3. PROGRAM PENTRU CALCULUL VALORII UNUI POLINOM DE GRADUL n

Programul folosește algoritmul lui Horner pentru calculul valorii polinomului:

$$P(x) = \sum_{i=0}^n a_i x^{n-i}$$

Polinomul $P(x)$ poate fi scris și sub următoarea formă:

$$P(x) = a_n + x(\dots + x(a_2 + x(a_1 + x(a_0)))) \dots$$

Calculul începe cu produsul cu cel mai mare grad de înbricare $x \cdot a$.

```

10 REM calculul valorii unui polinom de gradul n
20 PRINT "introduceți gradul n al polinomului"
30 INIT n
40 PRINT "n=" ; n
50 PRINT "introduceți coeficienții polinomului SUMA (a(i)*x^(n-i))"
60 DIM a(2*n)
70 FOR i=0 TO n
80 INPUT a(i+1)
90 PRINT "a (" ; i ; ")=" ; a(i+1)
100 NEXT i
110 PRINT "introduceți valoarea lui x"
120 INPUT x
130 PRINT "x=" ; x
200 REM calculul valorii polinomului prin metoda Horner
210 LET p=a(1)
220 FOR i=1 TO n
230 LET p=p*x+a(i+1)
240 NEXT i
300 REM afișarea rezultatului
310 PRINT "p (" ; x ; ")=" ; p
320 END
    
```

1.4. STABILIREA SOLUȚIEI UNEI ECUAȚII PRIN METODA DIHOTOMIEI

Se știe că dacă funcția f este continuă și strict monotonă în intervalul $[a, b]$ și dacă $f(a) \cdot f(b) < 0$, atunci ecuația: $f(x) = 0$ are o rădăcină unică în $[a, b]$.

Dacă c este un număr pozitiv și dacă $x_0 \in [a, b]$, iar $f(x_0 - c) \cdot f(x_0 + c) < 0$, atunci x_0 este o rădăcină a ecuației date cu o precizie c .

Calculul numeric al produsului $f(x_0 - c) \cdot f(x_0 + c)$ este în general aproximativ. Se impune ca precizia să fie suficientă pentru ca

$$f(x_0 - c) \cdot f(x_0 + c) < 0.$$

Se consideră jumătatea intervalului $[a, b]$: $c = (a + b) / 2$ și se testează semnul lui $f(a) \cdot f(c)$. Dacă $f(a) \cdot f(c) < 0$, rădăcina aparține lui $[a, c]$, în caz contrar aparține lui $[c, b]$.

Se construiesc următoarele intervale:

$$[a_0, b_0] = [a, b]$$

$$[a_1, b_1] = \begin{cases} [a, c], & \text{dacă } f(a) \cdot f(c) < 0 \\ [c, b], & \text{dacă } f(a) \cdot f(c) > 0 \end{cases}$$

$$[a_n, b_n] = \dots$$

Lungimea intervalului $[a_n, b_n]$ este $(b-a)/2^n$. Dacă se dorește o precizie ϵ , atunci este suficient să se construiască $[a_n, b_n]$ pentru un n astfel încât $(b-a)/2^n < \epsilon$.

```

10 REM stabilirea soluției unei ecuații folosind metoda dihotomiei
20 PRINT "introduceți datele: a=lim. stînga; b=lim. dreapta; e=precizia"
30 INPUT a
35 PRINT "a="; a
40 INPUT b
45 PRINT "b="; b
40 INPUT e
60 PRINT "e="; e
51 PRINT "introduceți gradul n al polinomului"
52 INPUT n
58 PRINT "n="; n
68 PRINT "introduceți coeficienții"
60 DIM a(2*n)
62 FOR i=0 TO n
64 INPUT a(i+1)
66 PRINT "a("; i; ")="; a(i+1)
68 NEXT i
80 REM calculul lui f(a) și f(b)
90 LET c=a
92 GO SUB 1210
94 LET x=z
96 LET c=b
98 GO SUB 1210
99 LET y=z
100 LET c=(a+b)/2
110 GO SUB 1210
120 LET d=x*z
130 IF d=0 THEN 200
140 IF d < 0 THEN 160
150 LET a=c
153 LET x=z
155 GO TO 170
160 LET b=c
165 LET y=z
170 IF b-a > e THEN 100
200 PRINT "c="; c
210 STOP
1210 LET p=a(1)
1220 FOR i=1 TO n
1230 LET p=p*c+a(i+1)
1240 NEXT i
1250 LET z=p
1260 RETURN
    
```

2. CALCULUL INTEGRALEI DEFINITE PRIN METODA SIMPSON

Pentru calculul integralei definite se folosește formula:

$$B = \int_{x_0}^{x_p} f(x) \cdot dx = (h/3) \cdot [(y_0 + y_{2p}) + 4(y_1 + y_3 + \dots + y_{2p-1}) + 2(y_2 + y_4 + \dots + y_{2p-2})]$$

unde:

$h = (x_p - x_0)/2 \cdot p$ este pasul,

$p =$ numărul de diviziuni.

$y = f(x)$ se dă la linia 500 din program; în cazul de față

$y = ((x-2) \cdot x - 1) \cdot x + 2$.

```

10 REM CALCULUL INTEGRALEI DEFINITE
20 PRINT "X0=";
30 INPUT D
40 PRINT "X2P="
50 INPUT E
60 PRINT "P=";
70 INPUT P
80 B=(E-D)/2/F
90 A=0
100 X=D
110 GOSUB 500
120 A=Y+A
130 X=X+B
140 GOSUB 500
150 X=4*Y+A
160 X=X+B
170 GOSUB 500
180 A=Y+A
190 F=F-1
200 IF F=0 THEN 220
210 GO TO 40
220 C=A*B/3
230 PRINT "REZ="; C
240 STOP
500 Y=((X-2)*X-1)*X+2
510 RETURN
    
```



(continuare de la vol. 1)

- Un număr la fel de important de programe, din toate domeniile de aplicație, rezolvă probleme simple și complexe de matematică-fizică din industrie și învățământ (aici este inclus și un îndrumar pentru predarea matematicii asistate de calculatorul personal); alte programe rezolvă jocuri, trasează grafice și figuri etc.
- Specialiștilor le atragem atenția asupra existenței comenzilor pentru toate tipurile de monitoare aMIC și – în acest context – asupra expunerii complete a microprocesorului Z80, cum și a uneia succinte a microprocesoarelor 8080.
- Sunt incluse programe de testare și indicații de depanare, ca și referiri concrete la perifericele utilizabile cu aMIC – display-uri (televizoare SPORT, de exemplu), casete ș.a.
- Editura Tehnică se mândrește cu activitatea sa de pionierat în sprijinirea introducerii intensive a calculatoarelor personale românești și recomandă cu căldură consultarea acestei cărți din noul ciclu „Totul despre...”, ca și a altor cărți de informatică, inclusiv a volumelor din seria continuă AMC (Automatică-Management-Calculatoare).

Vol. I și II Lei 42

ADDRESS	DATA	EXTER
0000		
0001		
0002		
0003		
0004		
0005		
0006		
0007		
0008		
0009		
000A		
000B		
000C		
000D		
000E		
000F		
0010		
0011		
0012		
0013		
0014		
0015		
0016		
0017		
0018		
0019		
001A		
001B		
001C		
001D		
001E		
001F		
0020		
0021		
0022		
0023		
0024		
0025		
0026		
0027		
0028		
0029		
002A		
002B		
002C		
002D		
002E		
002F		
0030		
0031		
0032		
0033		
0034		
0035		
0036		
0037		
0038		
0039		
003A		
003B		
003C		
003D		
003E		
003F		
0040		
0041		
0042		
0043		
0044		
0045		
0046		
0047		
0048		
0049		
004A		
004B		
004C		
004D		
004E		
004F		
0050		
0051		
0052		
0053		
0054		
0055		
0056		
0057		
0058		
0059		
005A		
005B		
005C		
005D		
005E		
005F		
0060		
0061		
0062		
0063		
0064		
0065		
0066		
0067		
0068		
0069		
006A		
006B		
006C		
006D		
006E		
006F		
0070		
0071		
0072		
0073		
0074		
0075		
0076		
0077		
0078		
0079		
007A		
007B		
007C		
007D		
007E		
007F		
0080		
0081		
0082		
0083		
0084		
0085		
0086		
0087		
0088		
0089		
008A		
008B		
008C		
008D		
008E		
008F		
0090		
0091		
0092		
0093		
0094		
0095		
0096		
0097		
0098		
0099		
009A		
009B		
009C		
009D		
009E		
009F		
00A0		
00A1		
00A2		
00A3		
00A4		
00A5		
00A6		
00A7		
00A8		
00A9		
00AA		
00AB		
00AC		
00AD		
00AE		
00AF		
00B0		
00B1		
00B2		
00B3		
00B4		
00B5		
00B6		
00B7		
00B8		
00B9		
00BA		
00BB		
00BC		
00BD		
00BE		
00BF		
00C0		
00C1		
00C2		
00C3		
00C4		
00C5		
00C6		
00C7		
00C8		
00C9		
00CA		
00CB		
00CC		
00CD		
00CE		
00CF		
00D0		
00D1		
00D2		
00D3		
00D4		
00D5		
00D6		
00D7		
00D8		
00D9		
00DA		
00DB		
00DC		
00DD		
00DE		
00DF		
00E0		
00E1		
00E2		
00E3		
00E4		
00E5		
00E6		
00E7		
00E8		
00E9		
00EA		
00EB		
00EC		
00ED		
00EE		
00EF		
00F0		
00F1		
00F2		
00F3		
00F4		
00F5		
00F6		
00F7		
00F8		
00F9		
00FA		
00FB		
00FC		
00FD		
00FE		
00FF		

