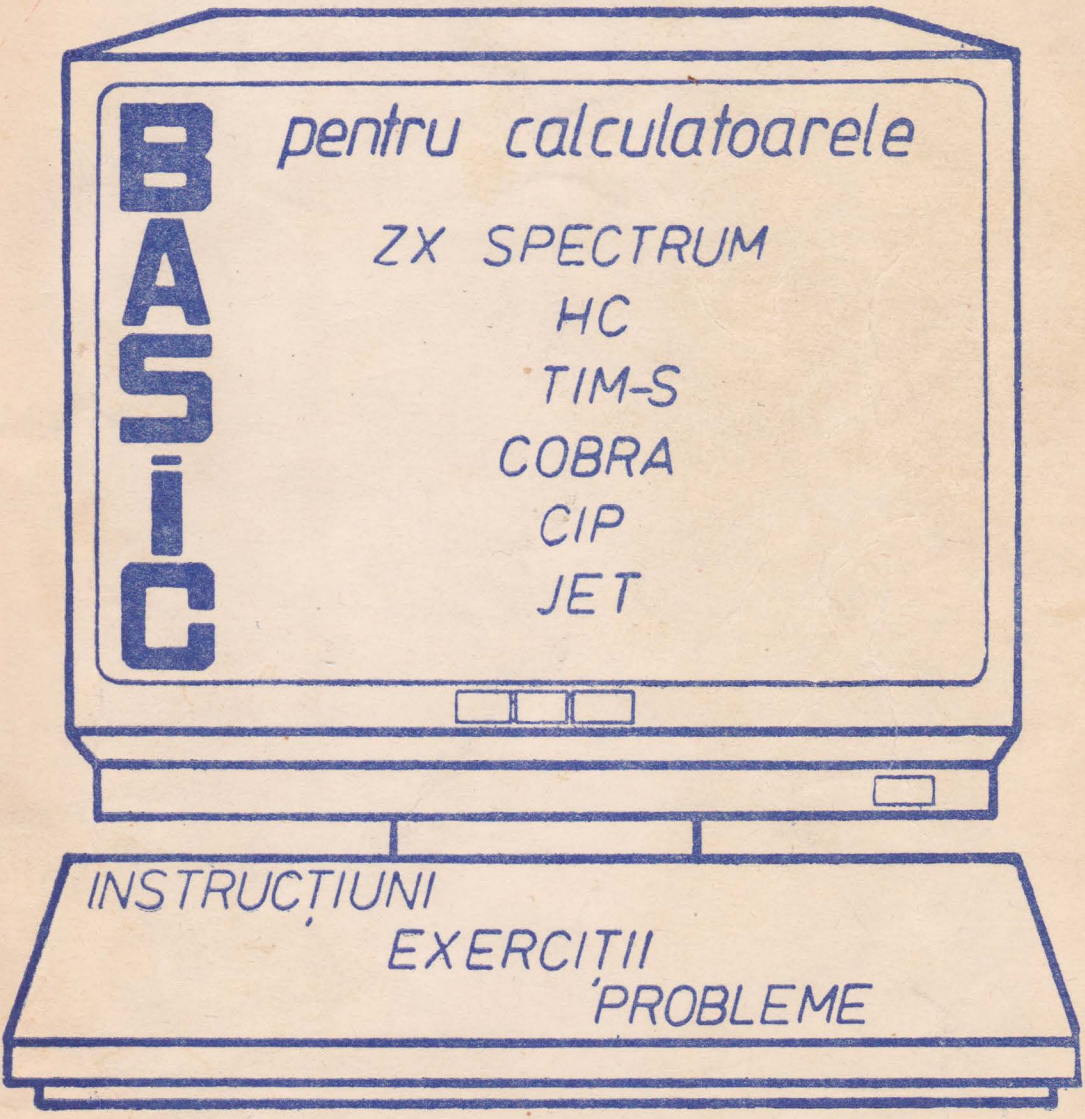
SERIA

INFORMATICA

MIRCEA – MIHAIL POPOVICI



EDITURA APH BUCUREȘTI 1992

Lei 396

În curs de apariţie de acelaşi autor :

● **BASIC pentru calculatoarele ZX SPECTRUM, HC, TIM-S, COBRA, CIP, JET...**

(*COLECŢIE DE PROGRAME*)

Lucrarea conţine programe tehnico-ştiinţifice, de matematică, de interes general, programe de divertisment (jocuri), precum şi prezentarea programelor *BETA BASIC şi HISOFT BASIC* însoţite de aplicaţii.

● **LIMBAJUL MAŞINĂ al calculatoarelor ZX SPECTRUM, HC, TIM-S, COBRA, CIP, JET...**

Este prima lucrare care tratează în mod unitar folosirea limbajului de asamblare Z80 şi este ilustrată cu peste 150 rutine care realizează spectaculoase efecte vizuale sonore, de scriere, de animaţie, etc.

Au apărut :

R. M. Hristev — **Introducere în PROLOG**

Limbajul inteligenţei artificiale pentru calculatoarele compatibile cu ZX Spectrum.

GHIDUL utilizatorului SPECTRUM

Scheme hard, harta memoriei ROM, modurile de utilizare ale tuturor compilatoarelor disponibile : BASIC, BETA-BASIC, FIFTH, Asamblor, Dezasamblor, PASCAL, C, FORTH, PROLOG.

ISBN 973 — 95175 — 4 — 4

SERIA

INFORMATICA

MIRCEA–MIHAIL POPOVICI

BASIC

pentru calculatoarele

ZX SPECTRUM

HC

TIM-S

COBRA

CIP

JET...

●

INSTRUCȚIUNI

EXERCIȚII

PROBLEME

EDITURA APH BUCUREȘTI 1992

Lucrarea *inițiază* cititorul în cunoașterea și folosirea limbajului **BASIC** utilizat la calculatoarele personale compatibile cu **ZX SPECTRUM** și anume **HC, TIM-S, COBRA, CIP, JET...**

Prin *explicații, exerciții* progresive ca dificultate și apoi prin *probleme*, cititorul este condus treptat spre *tehnici de programare avansate*, capabile să realizeze produse informatice complexe, din orice domeniu de preocupare, utilizînd efecte vizuale și sonore atractive.

Capitolul *1* are ca principal obiect de studiu însușirea modului de lucru cu tastatura calculatorului.

Capitolul *2* face o analiză a limbajului *BASIC* și tratează organizarea memoriei și a ecranului, codurile caracterelor și mesajele de eroare.

În **capitolele** *3—10* sînt tratate pe larg instrucțiunile limbajului; în acest scop sînt prezentate circa 200 programe prin care se efectuează calcule, se prezintă diverse modalități de scriere alături de efecte audio-vizuale, precum și tehnici de desenare sau de animație.

Capitolul *11* este consacrat unor artificii pentru perfecționarea și protejarea programelor.

Capitolul *12* tratează alcătuirea și copierea programelor complexe, prezentîndu-se modele de programe *"loader"*.

Prin întreaga ei structură, lucrarea se adresează posesorilor de calculatoare personale *ZX SPECTRUM, HC, TIM-S, COBRA, CIP, JET,...*, fiind un **ghid complet** de inițiere și conducere în programarea calculatoarelor.

M. M. POPOVICI

Copyright © 1992 EDITURA APH – SRL

str. Cap. Preda nr. 12 sect. 5, 76437 București 69
tel. 80.20.30 80.93.97, 80.74.77

Redactor, tehnoredactor: **A. Hristev**

Bun de tipar: 9. VI. 1992. Apărut 1992
Format 70×100/16. Coli de tipar: 10,5

Tipografia S.C. „Universul S.A.", Cd. 434/1992

CAPITOLUL 1

NOȚIUNI INTRODUCTIVE

1.1. PRELIMINARII

Realizarea importantă a industriei de tehnică de calcul din ultimii ani, calculatorul personal are utilizatori din cele mai diverse categorii sociale: ingineri, economiști, tehnologi, proiectanți, matematicieni, fizicieni, chimiști, medici, meteorologi, metrologi, profesori, funcționari, artiști, muncitori, agricultori, sportivi, elevi ș.a. Explicația constă în virtuțile acestui echipament dintre care se detașează: viteză mare de calcul, accelerarea procesului decizional folosind un mare volum de informații stocat și prelucrat corespunzător, creșterea calității cercetării științifice și proiectării tehnologice, folosirea cu rezultate superioare a resurselor materiale și umane, realizarea de economii de materiale, materii prime și combustibili, perfecționarea serviciilor și, nu în ultimul rînd, paleta largă de divertisment oferită de „jocurile pe calculator" cu o grafică și ilustrare muzicală extrem de atractive.

Se alătură acestor calități și faptul că un calculator personal are un gabarit mic și o greutate redusă, este interactiv și ușor de deplasat și montat unde se dorește și se caracterizează printr-un preț accesibil.

În structura lor se disting:

— unul sau mai multe microprocesoare;

— o memorie internă cu o capacitate de stocare suficientă pentru nevoile curente (64 Ko);

— o tastatură alfanumerică prin intermediul căreia operatorul comunică cu calculatorul;

— un sistem de afișare pe ecran TV sau pe ecran plat;

— o memorie externă formată din casete magnetice și discuri flexibile sau rigide.

Prin convenție sînt considerate calculatoare personale cele care folosesc un microprocesor de 8 biți, o memorie internă de 64 Ko, un echipament de vizualizare tip TV alb-negru sau color și o memorie externă ce are ca suport caseta magnetică. Din această categorie fac parte calculatoarele *ZX SPECTRUM* (cu variantele 48 Ko și „+2" realizate în Anglia și cu o mare răspîndire în Europa) precum și cele compatibile cu acesta executate în țara noastră: *HC-85*, *Tim S*, *Cobra*, *Cip*, *JET*.

Calculatoarele personale-profesionale sînt considerate acele echipamente bazate pe microprocesoare de 8/16 biți, memorii interne de minimum 64 Ko, dispozitiv de vizualizare de tip monitor (monocrom sau policrom), unitate de discuri flexibile/rigide, imprimantă, etc. Din această categorie

fac parte *FELIX M 118*, *ZX SPECTRUM +3*, *HC-88*, *TPD Junior*, *FELIX PC*, *XT-Junior* ş.a.

Principalul limbaj folosit de calculatoarele personale este limbajul BASIC.

1.2. CONFIGURAŢIA HARD A CALCULATOARELOR PERSONALE

Componentele fizice ale calculatorului denumite *hard* (prescurtare de la cuvîntul în limba engleză *hardware* — parte tare) sînt: *calculatorul* propriu zis (care conţine şi *tastatura*), *sursa de alimentare*, *televizorul* (TV) şi *casetofonul* (fig. 1.1). Aceste elemente sînt conectate între ele prin cabluri de legătură prevăzute cu conectări ce se potrivesc cu mufele din calculator, TV şi casetofon.

În fig. 1.2 este prezentată schema-bloc a unui calculator personal.

Cea mai importantă componentă a sa este **microprocesorul** sau **Unitatea centrală** *(UC)*. Calculatoarele *ZX SPECTRUM* şi *HC-85* au de exemplu microprocesorul *Z80A*. Întrucît Unitatea centrală este inutilă fără memorie, calculatorul dispune de două tipuri de memorie internă:

ROM *(Read Only Memory)*

RAM *(Random Access Memory)*

Memoria **ROM** conţine date şi programe nevolatile, care permit calculatorului să înceapă lucrul imediat ce este pornit. În **RAM** se realizează programul conceput de utilizator, dar conţinutul **RAM**-ului se pierde atunci cînd calculatorul este oprit; din acest motiv datele şi programele din **RAM** trebuie transferate (salvate) pe caseta magnetică.

Comunicarea calculatorului cu exteriorul se face prin intermediul *dispozitivelor periferice* (claviatura, TV, casetofonul şi difuzorul), controlate de *Blocul logic de control* **BLC**. *Difuzorul* şi *TV* se folosesc pentru comunicarea cu operatorul, iar *casetofonul* este un dispozitiv de intrare/ieşire pentru memorarea de programe şi date. *Claviatura* serveşte pentru a se introduce instrucţiuni, comenzi sau date.

Utilizînd *conectorul interfaţă* se pot conecta alte periferice în plus faţă de cele indicate de schema-bloc (de ex. imprimanta).

1.3. PUNEREA IN FUNCŢIUNE A CALCULATORULUI

Pentru a se evita eventualele defecţiuni, trebuiesc respectate următoarele etape de montare:

1) *Se montează cablul de legătură dintre calculator şi TV.*

2) *Se conectează televizorul la reţeaua de 220 V/50 Hz şi se acordează pe canalul specificat în cartea calculatorului.*

3) *Se montează cablul de legătură dintre calculator şi casetofon.*

4) *Se verifică funcţionarea casetofonului, prin apăsarea tastei PLAY.*

5) *Se introduce sursa de alimentare a calculatorului în priza de 220 V şi se conectează firul ei de legătură la calculator* (sursa va furniza o tensiune continuă de 9 V).

Se menţionează că sistemul nu dispune de un întrerupător PORNIT/OPRIT şi deci calculatorul intră imediat în funcţiune după ce a fost

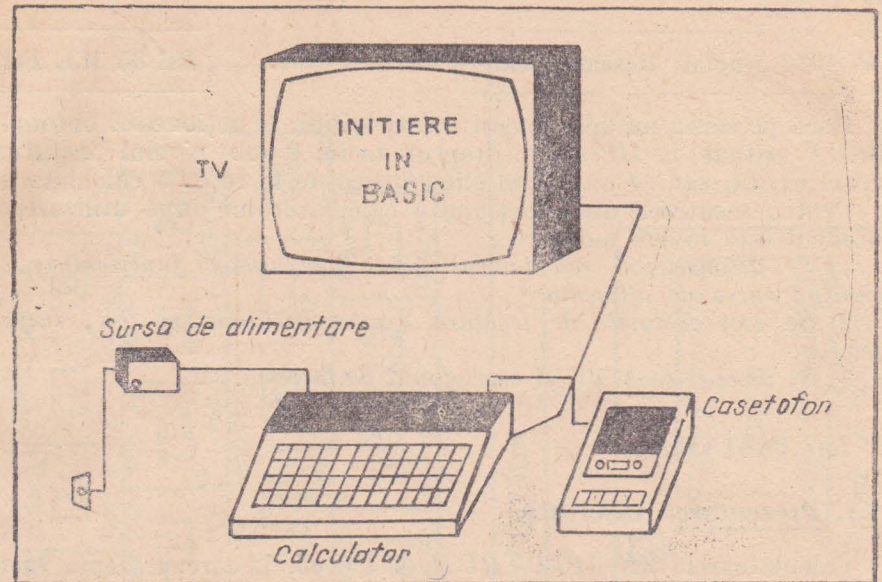


Fig. 1.1 Configuratia hard a calculatoarelor personale

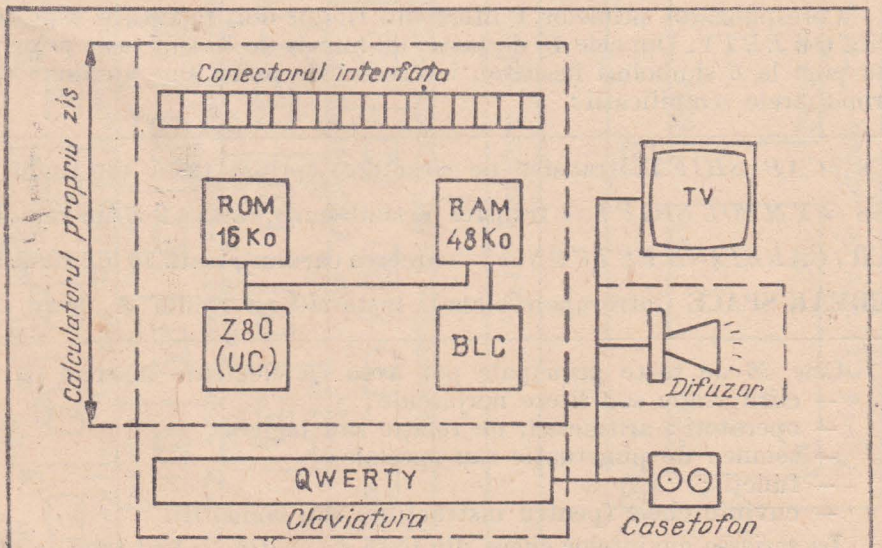


Fig. 1.2 Schema - bloc a unui calculator personal

racordat la rețea. Pe ecranul *TV* apare un mesaj specific tipului de calculator; astfel la calculatoarele *ZX-SPECTRUM* și *HC 85* aceste mesaje sînt:

© 1982 Sinclair Research Ltd	HC 85 ICE Felix

Dacă pe ecran nu apare acest mesaj inițial, se acționează butonul de *RESET* (situat la *HC 85* în dreptul tastei **0** sub nivelul tastaturii), dacă el există, sau se întrerupe alimentarea de la rețea a calculatorului.

Pentru scoaterea din funcțiune a calculatorului după utilizarea sa, procedeul este invers montării:

1) *Se deconectează calculatorul prin întreruperea funcționării sale, decuplînd sursa de alimentare.*

2) *Se scot cablurile de legătură dintre calculator și TV, respectiv casetofon.*

3) *Se deconectează TV și casetofonul de la rețea.*

1.4. TASTATURA

1.4.1. Prezentarea tastaturii

Calculatoarele *ZX-SPECTRUM* și *HC 85*, la care prezenta lucrare se referă cu precădere motiv pentru care se va folosi termenul generic de *calculator*, au o tastatură (claviatură) similară unei mașini de scris, deoarece literele și cifrele sînt plasate în aceeași poziție (cu excepția literelor *Q*, *Z*, *M*). Această claviatură conține *40* taste plasate pe 4 rînduri cîte *10* pe o linie (fig. 1.3)

Corespunzător primelor *6* litere din rîndul doi, tastatura este denumită *QWERTY*. Din cele *40* de taste, un număr de *36* sînt **taste principale** (au pînă la *6* simboluri înscrise), iar restul de patru sînt **auxiliare** și au următoarele semnificații:

CS *(CAP SHIFT*=transfer pe caractere), prima tastă din rîndul *4*;

SS *(SYMBOL SHIFT* = transfer pe simboluri), tasta a *9*-a din rîndul *4*;

CR *(CARRIAGE RETURN*=înapoierea carului), tasta *10* din rîndul *3*;

BREAK SPACE (întrerupere/blanc), tasta *10* pe rîndul *4*.

Cele *36* de taste principale pot avea ca elemente înscrise pe ele:
- cifre arabe sau litere majuscule;
- operatori: aritmetici, de relație sau logici;
- semne: de punctuație sau speciale;
- funcții;
- cuvinte cheie (pentru instrucțiuni sau comenzi).

La scrierea cuvintelor cheie, din lipsă de spațiu au fost folosite următoarele prescurtări:

TR VIDEO	de la TRUE VIDEO
INV VIDEO	de la INVERSE VIDEO
RAND	de la RANDOMIZE

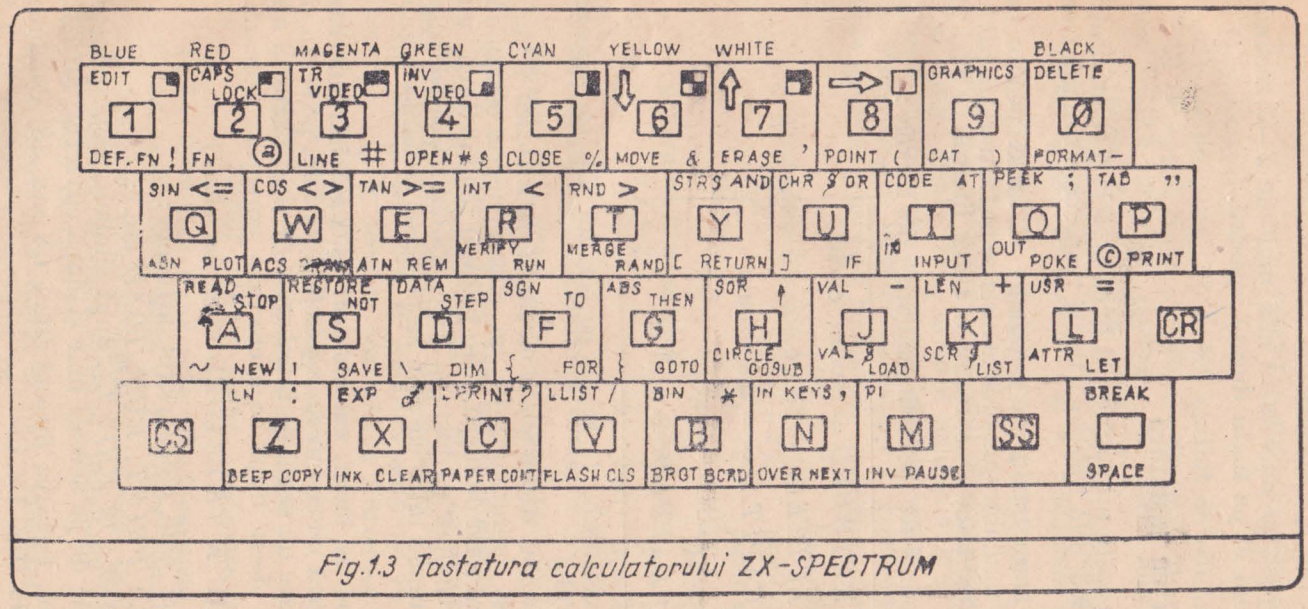


Fig.1.3 Tastatura calculatorului ZX-SPECTRUM

Fig. 1.4 Exemple de simboluri înscrise pe taste literale și numerice

BRGT	de la BRIGHT
INV	de la INVERSE
SCR$	de la SCREEN$
CONT	de la CONTINUE

Se menționează că apăsarea unei taste mai mult de *2—3* secunde determină repetarea acțiunii ei pe toată durata apăsării.

1.4.2. Modurile de lucru cu tastatura

Fiecare din cele *36* de taste principale pot avea pînă la *6* semnificații diferite care se selecționează prin apăsarea tastei respective simultan cu tastele *CS* și/sau *SS*.

Se numește **mod de lucru** modul de acționare al tastei pentru ca în memorie să se introducă unul din simbolurile înscrise pe tasta respectivă. Pentru exemplificare în fig. 1.4a, b se redau o tastă literală *(TL)* și o tastă numerică *(TN)*.

Calculatorul are 5 moduri de lucru desemnate prin literele:

K	de la *KEYWORD* = cuvînt cheie
L	de la *LETTERS* = litere
C	de la *CAPITALS* = majuscule
E	de la *EXTEND* = extins
G	de la *GRAPHICS* = grafic.

După conectarea calculatorului la rețea, pe ecranul *TV* se afișează mesajul specific calculatorului.

Apăsînd *CR*, în partea din stînga jos a ecranului se afișează *K* sub forma unui **cursor clipitor**, reprezentat de un pătrat negru conținînd litera *K*. Acesta este primul mod de lucru, care poate fi schimbat cu un alt mod prin respectarea regulilor de tastare. Este evident că și celelalte moduri de lucru se pot schimba între ele prin respectarea acelorași reguli de tastare.

Odată instituit un mod de lucru, calculatorul așteaptă introducerea unui anumit tip de date. Dacă nu se introduc datele așteptate, pe ecran și în locul unde s-a greșit apare un **semn de întrebare** *(?)* și utilizatorul nu poate trece mai departe decît după ce a efectuat corecția neceară.

A) **MODUL DE LUCRU K** *(KYEBORD = cuvînt cheie)*

Modul *K* apare atunci cînd calculatorul așteaptă:

— o comandă (de ex.: *PRINT, RUN, LOAD* etc.);

— o linie program (de ex: *10 PRINT)*, după simbolul două puncte *(:)* ce separă instrucțiunile de pe aceeași linie și după cuvîntul *THEN*.

În acest mod de lucru, tastarea directă (fără *CS* sau *SS*) conduce la apariția simbolurilor din (fig. 1.4, c):

— poziția 3 pentru tasta laterală *(TL)*

— poziția *1* pentru tasta numerică *(TN)*.

Rezumînd, în modul *K* tastele literale sînt interpretate drept **cuvînt-cheie** conform indicațiilor din poziția 3, iar tastele numerice sînt interpretate ca **numere**, potrivit notațiilor din poziția *1*.

Exemplul 1 : Introducerea comenzii

PRINT (adică „tipărește")

se face astfel :

1) La începutul liniei din spațiul de editare este afișat cursorul *K*.

2) Se apasă tasta *P* ceea ce determină apariția pe ecran a cuvîntului-cheie **PRINT** aflat pe poziția *3* a tastei literale *P*.

Exemplul 2 : Scrierea cifrei *10* :

1) La începutul liniei este afișat cursorul *K*.

2) Se apasă pe tastele *1* și *0*, rezultînd numărul *10*.

B) MODUL DE LUCRU L *(LETTERS = litere)*

Acest mod alternează cu modul *K* și apare **automat** după introducerea unui cuvînt cheie. Prin urmare modurile *K* și *L* sînt cele mai frecvent folosite moduri de lucru ale calculatorului.

Semnificația tastelor în modul *L* este indicată în cele ce urmează unde semnul plus (+) semnifică *„acționare simultană"* :

— pentru tasta literală *(TL)*

TL	poziția *1* (literă mică)
CS+TL	poziția *1* (literă mare)
SS+TL	poziția *2*.

— pentru tasta numerică *(TN)*

TN	poziția *1*
CS+TN	poziția *4*
SS+TN	poziția *3*.

Exemplul 1 : Introducerea comenzii

PRINT 10 (adică *„tipărește numărul 10"*) :

1) La începutul liniei este afișat cursorul *K*.

2) Se acționează tasta *P*, ceea ce semnifică afișarea cuvîntului cheie **PRINT** (din poziția a *3*-a a tastei *P*).

3) Modul de lucru se schimbă automat în *L*; deci se apasă tastele numerice *1* și *0*, ceea ce conduce la introducerea simbolurilor aflate pe poziția *1* a acestor taste numerice.

4) Se apasă pe *CR* și pe ecran apare cifra *10*.

Exemplul 2 : Introducerea comenzii

PRINT "10%" (adică *„tipărește textul 10%"*) :

1) La începutul liniei este afișat cursorul *K*.

2) Se acționează tasta *P* și apare comanda **PRINT**.

3) Modul de lucru se schimbă automat în *L*; deci se apasă simultan *SS* și *P* ceea ce produce scrierea simbolului ghilimele (") de pe poziția a *2*-a a tastei literale *P*.

4) În continuare modul de lucru rămînînd *L* se acționează tastele *1* și *0* pt afișarea numărului *10*.

5) Fiind în modul *L*, se apasă simultan tastele *SS* și *5*, rezultînd afișarea simbolului %.

6) Rămînînd în modul *L*, se apasă simultan tastele *SS* și *P* pentru închiderea ghilimelelor.

7) Se apasă *CR* și pe ecran apare mesajul *10%*.

● Acționînd simultan, în modul *L*, *CS* și o tastă numerică se obține simbolul specificat în poziția a *4*-a (**EDIT CAPS, LOCK DELETE**)

Exemplu : Introducerea comenzii

DELETE (adică *„șterge un număr de caractere"*)

se face acționînd simultan tastele *CS* și *0* după cum urmează :

— o singură dată pentru ștergerea unui caracter ;

— de atîtea ori cîte caractere trebuiesc șterse.

● Acționînd simultan în modul *L* tastele *CS* și o tastă literală se produce transformarea literei mici în literă majusculă.

Exemplu : Introducerea comenzii

PRINT A*B (adică *„tipărește produsul numerelor A* și *B"*) se face astfel :

1) Calculatorul fiind în modul *K* se va tasta *P* și pe ecran apare cuvîntul cheie **PRINT**.

2) Modul de lucru schimbîndu-se automat în *L*, se acționează simultan tastele *CS* și *A*, rezultînd litera majusculă *A*.

3) Se tastează simultan *SS* și *B* rezultînd simbolul asterisc (*).

4) Se apasă simultan *CS* și *B* rezultînd litera majusculă *B*.

C) **MODUL DE LUCRU C** *(CAPITAL = litere majuscule)*

Reprezintă o variantă a modului de lucru *L*, în care scrierea se face cu litere majuscule. Trecerea în modul *C* se face acționînd simultan tastele *CS* și *2* (adică se realizează **CAPS LOCK**). Pentru modul *C* semnificațiile tastelor este prezentată în cele ce urmează :

— pentru tasta literală *(TL)* $\left[\begin{array}{ll} TL & \text{poziția } 1 \text{ literă majusculă} \\ SS+TL & \text{poziția } 2 \end{array}\right.$

— pentru tasta numerică *(TN)* $\left[\begin{array}{ll} TN & \text{poziția } 1 \\ SS+TN & \text{poziția } 3. \end{array}\right.$

Ieșirea din modul *C* pentru a se reveni în modul *L* se face acționînd din nou *CS* și *2*.

Exemplu : Introducerea comenzii

PRINT A * B

se derulează astfel :

1) Fiind în modul *K* se va tasta *P* și pe ecran apare **PRINT**.

2) Modul de lucru devine automat *L* ; deci se apasă simultan *CS* și *2* pentru a se obține modul *C*, după care se apasă direct tasta *A*, rezultînd litera majusculă *A*.

3) Se apasă simultan *SS* și *B* rezultînd semnul asterisc (*).

4) Păstrîndu-se modul *C*, se apasă direct tasta *B* obținîndu-se litera majusculă *B*.

Din cele prezentate se deduce că:

— pentru a scrie numai cu litere majuscule este necesară folosirea modului *C*;

— pentru a scrie întîmplător o literă majusculă este suficientă acționarea simultană a tastei *CS* și a tastei literale respective.

D) MODUL DE LUCRU E *(EXTENDED = extins)*

Este folosit pentru a se obține simboluri noi sau instrucțiuni și comenzi mai puțin uzitate. Intrarea în modul *E* se face acționînd simultan tastele *CS* și *SS*, iar ieșirea din acest mod se face automat după prima tastare.

În modul *E* semnificația tastelor este următoarea:

— pentru tasta literală *(TL)*
- *TL* — poziția *4*
- *SS*+*TL* — poziția *5*

— pentru tasta numerică *(TN)*
- *TN* — (*1* la *7* și *0*, controlul culorii)
- *SS*+*TN* — poziția *5*.

Corespondența dintre culori și tasta numerică este indicată mai jos:

1 — albastru *(BLUE)*
2 — roșu *(RED)*
3 — purpuriu *(MAGENTA)*
4 — verde *(GREEN)*
5 — bleu *(CYAN)*
6 — galben *(YELOOW)*
7 — alb *(WHITE)*
0 — negru *(BLACK)*.

Exemplul 1 : Introducerea comenzii

READ p (adică *„citește valoarea variabilei p"*)

se face în succesiunea:

1) Modul *K* se schimbă în modul *E* apăsînd simultan tastele *CS* și *SS*.

2) Se apasă tasta *A* rezultînd apariția cuvîntului cheie **READ**.

3) Modul de lucru devenind automat *L*, se apasă tasta *P* rezultînd litera mică *p*.

Exemplul 2 : Introducerea comenzii

BEEP 2,2 [adică *„emite timp de 2 secunde* (prima cifră) *nota re"* (cifra a 2-a)]

se face astfel:

1) Se trece de la modul *K* la modul *E* apăsînd simultan *CS* și *SS*.

2) Se apasă simultan *SS* și *Z*, rezultînd simbolul din poziția *5*, respectiv cuvîntul cheie **BEEP**.

3) Modul de lucru devenind automat *L*, se apasă tasta *2* și se obține afișarea cifrei *2*.

4) Menținîndu-se modul *L*, se apasă simultan *SS* și *N* rezultînd simbolul virgulă (,).

5) Se apasă tasta *2*, afișîndu-se cifra *2*.

6) Se apasă *CR* și se aude sunetul.

E) MODUL DE LUCRU G *(GRAPHICS = grafic)*

Acest mod se obține apăsînd simultan tastele *CS* și *9*; el permite folosirea caracterelor grafice de care dispune calculatorul sau cele ce sînt definite de utilizator. Astfel, dacă se tastează:

— o tastă numerică *(TN)*, exceptînd *0* și *9* care nu au poziția *2*, se obține mozaicul grafic desenat cu negru pe tasta respectivă în poziția *2*

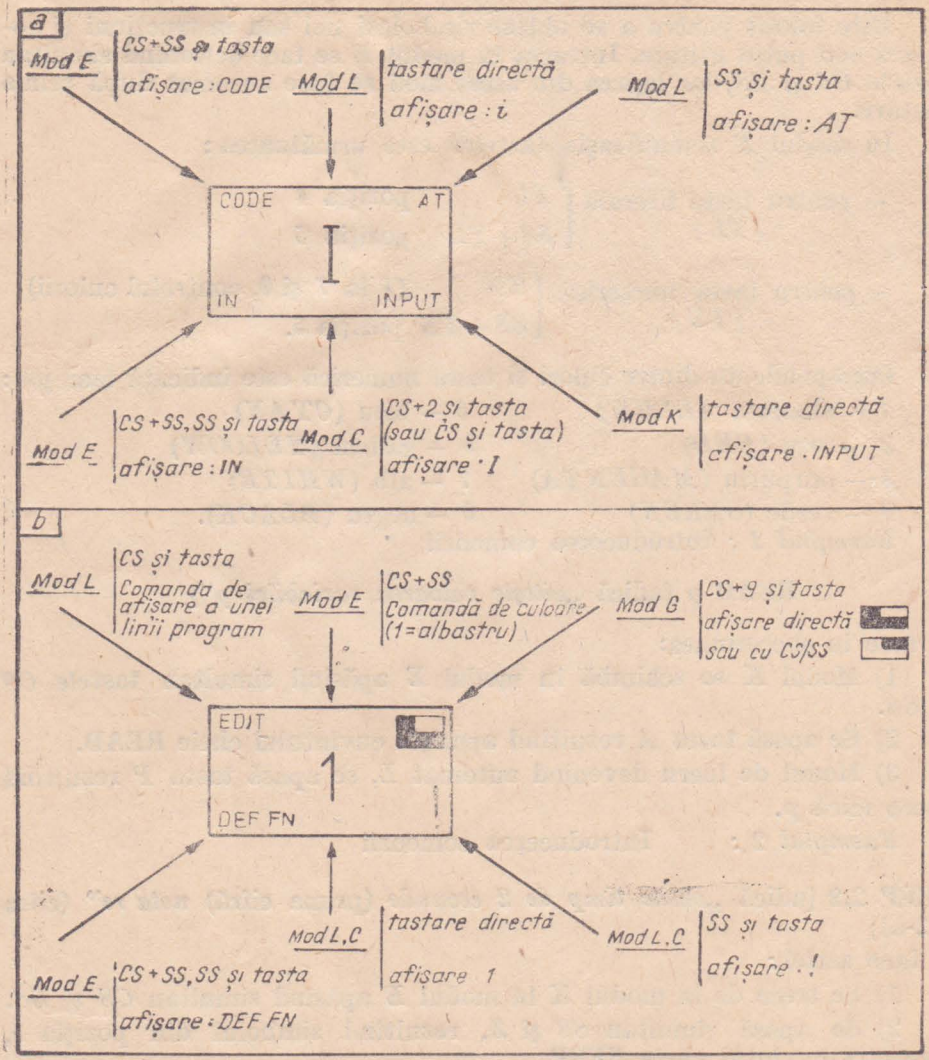


Fig.1.5 Modalitățile de obținere a simbolurilor înscrise pe taste

sau cel desenat cu culoarea albă (complementul lui) acționînd *CS/* sau *SS* și tasta numerică;

— o tastă literală între literele alfabetului *A* și *U* (21 litere), exceptînd *V*, *W*, *X*, *Y*, *Z*, rezultînd un caracter grafic nou definit de utilizator (v. Cap. 9).

Exemplu : **Introducerea comenzii**
DRAW 10, 100, PI (adică *„trasează un semicerc avînd centrul de coordonate 10, 100 și lungimea arcului* **PI** *radiani"*), se face astfel :

1) Fiind în modul *K*, se apasă tasta *W* și rezultă cuvîntul cheie **DRAW**.

2) Întrucît modul de lucru devine *L*, se acționează tastele *1* și *0*, rezultînd numărul *10*.

3) Menținîndu-se modul *L*, se apasă simultan *SS* și *N* pentru a rezulta simbolul virgulă (,).

4) Modul de lucru rămînînd *L*, se apasă pe tastele *1* și *0* (de două ori) pentru a se afișa cifra *100*.

5) Se apasă simultan *SS* și *N* rezultînd simbolul virgulă.

6) Se trece în modul *G* apăsînd simultan *CS* și *9* după care se apasă tasta *X* rezultînd simbolul *PI*.

7) Se apasă *CR* și pe ecran se desenează un semicerc.

În fig. 1.5 sînt indicate modalitățile de obținere a simbolurilor înscrise pe tastele literale și numerice.

1.5. PROBLEME

Pl.1 Apăsați butonul RESET pentru a face să apară mesajul specific calculatorului și apoi :

— tastați pe rînd toate tastele numerice ;
— tastați toate tastele literale (minuscule și majuscule).

Pl.2 Afișați toate simbolurile situate în poziția *2* pe tastele literale.

Pl.3 Afișați toate cuvintele cheie.

Pl.4 Treceți în modul *E* și obțineți toate comenzile înscrise pe tastele literale în pozițiile *4* și *5*.

Pl.5 Treceți în modul *G* și afișați toate simbolurile grafice ale calculatorului.

Capitolul 2

LIMBAJUL BASIC

2.1. NOȚIUNI INTRODUCTIVE

Orice program elaborat pentru un calculator reprezintă forma *codificată a unui algoritm*; el este exprimat într-un anumit *limbaj de programare* ce asigură dialogul operator—calculator. Calculatorul execută **instrucțiuni/comenzi** exprimate intern sub forma unei succesiuni de cifre binare (*0* și *1*). Programul care utilizează o asemenea notație este un program în **limbaj cod-mașină** propriu circuitelor electronice ale calculatorului. Programarea în limbaj cod-mașină este însă laborioasă și incomodă. Din acest motiv s-a căutat ca acest inconvenient să fie eliminat prin exprimarea instrucțiunilor/comenzilor sub *formă simbolică*, rezultînd:

— **limbaje de nivel redus**, cum ar fi *limbajul de asamblare* specific unui anumit tip de calculator,

— **limbaje de nivel înalt**, independente de calculator, cum sînt limbajele *BASIC*, *FORTRAN*, *COBOL*, ș.a.

● *LIMBAJELE DE ASAMBLARE* au:

— ca **avantaje**: programe eficiente din punctul de vedere al timpului de execuție și al dimensiunii memoriei ocupate, alături de un acces direct la toate operațiile executabile de către partea electronică a calculatorului;

— ca **dezavantaje**: dependența de microprocesor (deoarece fiecare microprocesor are propriul său limbaj de asamblare), dimensiuni mari ale programelor, lipsa modalității de structurare complexă a datelor.

● *LIMBAJELE DE NIVEL ÎNALT* prezintă:

— ca **avantaje**: independență de calculator, ușurință în scrierea-depanarea-modificarea programelor, alături de posibilitatea utilizării bibliotecilor de programe.

— ca **dezavantaje**: timp de execuție și memorie ocupată mai mari și imposibilitatea utilizării facilităților microprocesorului.

● *PROGRAMELE* care sînt scrise în limbaje simbolice se numesc **programe-sursă**. Pentru a fi executate de calculator este necesară *traducerea* (translatarea) programelor sursă în codurile numerice ale limbajului mașină. Această traducere este realizată de un *program de translație* care prelucrează textul programului sursă și produce un alt text numit **program-obiect**.

Din categoria programelor de translație fac parte :

— **asamblorul**, pentru programe-sursă scrise în limbaj de asamblare ;

— **interpretorul** și **compilatorul**, pentru programele de nivel înalt. Traducerea prin :

— *compilare* are avantajul obținerii unui program-obiect cu timp scurt de execuție (deci eficient) și dezavantajul că operația de compilare nu este interactivă deoarece se compilează tot programul ;

— *interpretor* prezintă avantajul că este *interactivă* (programul este testat linie cu linie) și dezavantajul că este lentă.

Este util de precizat că din acest punct de vedere, orice calculator este folosit în două etape

— *etapa de compilare/interpretare*, care constă în executarea programului *COMPILATOR/INTERPRETOR* cu scopul traducerii programului-sursă în program-obiect ;

— *etapa de execuție a programului obiect*, prin care se asigură **prelucrarea datelor și obținerea rezultatelor.**

2.2. CARACTERIZAREA LIMBAJULUI BASIC

Acest limbaj a fost creat în *1964* de către profesorii *JOHN KEMENY* și *THOMAS KURTZ* de la Dartmonth College SUA, iar numele său provine de la inițialele definiției din limba engleză „*Beginner's All-purpose Symbolic Instruction Code*", adică un cod de instrucțiuni simbolice utilizabil de orice începător. El este un limbaj **conversațional, universal** și **algoritmic** folosit inițial pentru rezolvarea problemelor științifice-tehnice, de proiectare sau în învățământ. Datorită simplității sale, cunoaște o mare răspîndire fiind implementat în majoritatea sistemelor de calcul.

Principalele calități ale limbajului *BASIC* sînt :

— **universalitatea** (rezolvă orice problemă științifică, tehnică, economică, de învățământ și din alte domenii, inclusiv grafică și emitere de sunete) ;

— **portabilitate** (implementat pe toate calculatoarele personale, folosind o structură de bază a instrucțiunilor și comenzilor pentru toate variantele de *BASIC)* ;

— **interactivitate** (dialog operator-calculator, ultimul semnalînd instrucțiunile incorecte) ;

— **simplitate** (un număr de *95* instrucțiuni din care *68* sînt curente programele fiind ușor de depanat).

Ca dezavantaje se menționează : timp mai mare de execuție (deoarece este un limbaj interpretat) și nepretare la tehnicile programării structurate.

2.3. STRUCTURA LIMBAJULUI BASIC

Ca orice limbaj de programare, limbajul *BASIC* **posedă alfabet, vocabular, gramatică** și **semantică.**

2.3.1. Alfabetul

Acesta este format din :

— zece cifre : *0, 1, 2, 3, 4, 5, 6, 7, 8, 9* (cifra zero se scrie în programe ca **zero tăiat** pentru a nu se confunda cu litera o) ;

— două zeci și șase de litere (minuscule și majuscule ale alfabetului englez : *a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q ,r, s, t, u, v, w, x, y, z, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z* ;

— trei zeci și patru de caractere speciale

:	!	©	&	#	%	'	(	)	-	< =	< >	> =	<	>	"	↑	+	—	*	=	~	\	/	{	}	[	]	?	/	·	£	;	©

2.3.2. Vocabularul

Reprezintă totalitatea cuvintelor acceptate de limbaj și este constituit din :

a) *etichete* (număr de maximum *4* cifre întregi cuprins între *1* și *9999*, care precede orice linie de program) ;

b) *texte* (alcătuite din șiruri de caractere încadrate între ghilimele ; exemplu : "*Basic*") ;

c) *constante*, ce pot fi :

— *numerice* [format *întreg* (ex : *127*), format *real* (ex : *3.14* ; *.5*) unde virgula se reprezintă prin punct ; format *exponențial* [ex : *5e7* ; *876E0*) unde *e* sau *E* semnifică „*10 la puterea*"] ; se menționează că se pot face operații cu numere cuprinse între $4 . 10^{-59}$ și 10^{38}, dar se afișează maximum *8* cifre semnificative ;

— *șir de caractere* (scris între ghilimele ; exemple : „*BXm9*" și „*0+?z*").

a) *variabile* (simboluri algebrice reprezentînd nume) :

— *simple*, format dintr-o *literă* urmată sau nu de unul sau mai multe litere și/sau cifre (exemple : *SIGMA, sigma, MC, mc, BOx7*) ;

— *indexate* (tablouri cu *n* dimensiuni), formate *dintr-o literă* urmată între paranteze de unul sau mai mulți indici care pot fi : constante, variabile sau expresii aritmetice exemple : *A(3), B(X), C(2, w*2)* ;

— *șir*, al căror nume se formează din orice literă urmată de semnul *$* (exemple : *A$, x$*).

SE PRECIZEAZĂ CĂ ESTE PERMISĂ FOLOSIREA ACELUIAȘI NUME PENTRU O VARIABILĂ NUMERICĂ, O VARIABILĂ SIMPLĂ, O VARIABILĂ ȘIR DE CARACTERE SAU O VARIABILĂ INDEXATĂ

[exemple : *B, B$, B(2, 1), B$(3, 2)*]

e) *Operatori :*

— *aritmetici :* adunare +, scădere —, înmulțire *, împărțire /, ridicare la putere ↑ ;

— *relaționali :* egal =, mai mic <, mai mare >, mai mare sau egal > =, mai mic sau egal < =, diferit de < > ;

— *logici :* *NOT* (nu), *AND* (și), *OR* (sau) ;

— *de concatenare* (adunare) : +

EVALUAREA UNEI EXPRESII ARITMETICE SE FACE RESPECTÎND URMĂTOARELE PRIORITĂȚI:

— ridicare la putere;
— înmulțire și împărțire;
— adunare și scădere.

Exemple:

1) $7 . 10,2\left(\frac{23}{x} + B\right) + 0,5$ se scrie $7 * 10 .2 * (23/x + B) + .5$

2) $\frac{(CY + W)}{Z} \leqslant D^2$ se scrie $(C * Y + W)/Z <= D \uparrow 2$

3) $Z5 > 100$ și $B = 50$ sau $V \geqslant 3$ se scrie $(Z * 5 > 100)$ AND $B = 50$ OR $V >= 3$

4) Cuvîntul PRONOSPORT poate fi concatenat astfel „PRONO" + „SPORT"

2.3.3. Gramatica

Reprezintă totalitatea regulilor sintactice (de îmbinare a vocabularul lui) care trebuiesc respectate în scrierea instrucțiunilor unui program, Aceste reguli vor fi prezentate la studiul instrucțiunilor.

2.3.4. Semantica

Se referă la sensurile (semnificațiile) instrucțiunilor; ele vor fi prezentate la studiul instrucțiunilor.

2.4. FORMA GENERALĂ A UNUI PROGRAM BASIC

Pentru exemplificare se prezintă un program ce calculează suprafața cercului:

```
1Ø REM
2Ø INPUT "Introdu raza [m]", r
3Ø LET s=PI*r*r
4Ø PRINT AT 1Ø, 1; "Suprafața cercului este␣:"; AT 12, 1; "s="; s; "␣metri patrati" [1]
```

Analizînd acest program se disting:

— *etichetele* (numerele de linie: *1Ø, 2Ø, 3Ø, 4Ø*; se recomandă ca programele să se scrie cu numere de linie avînd rata *1Ø*, pentru a fi posibilă înserarea unei linii intermediare între două linii deja scrise:

— *cuvintele cheie* specifice limbajului *BASIC*: **REM, INPUT, LET, PRINT**

— *variabilele simple*: r; s.

[1] Prin semnul ␣ se înțelege un spațiu liber numit blanc (se obține apasînd tasta SPACE.

Se poate formula concluzia că forma unei instrucțiuni simple este:

nr. linie (eticheta)	cuvînt cheie *BASIC*	corp instrucțiune

(exemplu: **100 READ a**).

Instrucțiunile *multiple* se despart prin caracterul două puncte *(:)*; (exemplu: **110 BORDER 7 : PAPER 7 : INK 0 : CLS**); numărul maxim de instrucțiuni ce pot fi scrise într-o instrucțiune multiplă este de *127*.

Cuvintele cheie *BASIC* sînt următoarele (în ordine alfabetică):

ABS, ACS, AND, ASN, AT, ATN, ATTR, BEEP, BIN, BORDER, BRIGHT, CAT, CHR$, CIRCLE, CLEAR, CLOSE#, CLS, CODE, CONTINUE, COPY, COS, DATA, DEF FN, DIM, DRAW, ERASE, EXP, FLASH, FN, FOR, FORMAT, GOSUB, GOTO, IF, IN, INK, INKEY$, INPUT, INT, INVERSE, LEN, LET, LINE, LIST, LLIST, LN, LOAD, LPRINT, MERGE, MOVE, NEW, NEXT, NOT, OPEN#, OR, OUT, OVER, PAPER, PAUSE, PEEK, PLOT, POINT, POKE, PRINT, RANDOMIZE, READ, REM, RESTORE, RETURN, RND, RUN, SAVE, SCREEN$, SGN, SIN, SQR, STOP, STR$, TAB, TAN, THEN, TO, USR, VAL, VAL$, VERIFY.

2.5. ORGANIZAREA MEMORIEI

Memoria are roul de a păstra date și programe și este gestionată de unitatea centrală *UC* (sau cu acronimul din limba engleză *CPU* = Central Processing Unit), reprezentată la calculatoarele din familia *ZX—SPECTRUM* prin microprocesorul *Z80A*.

Pentru înțelegerea adreselor zonelor de memorie este necesar să se specifice că unitatea de măsură a cantității de informație este *BIT*-ul (de la *BInary digiT*). Pentru a codifica **un caracter** oarecare se folosesc *8 biți*, ceea ce a determinat elaborarea unui multiplu al bitului numit *octet*, *byte* sau *cuvînt de memorie* (multiplul de *4 biți* poartă denumirea de *nibble* sau *cuartet*).

Se definește *kilooctetul* ca fiind

$$\mathbf{1\ ko = 1024\ octeți = 2^{10}\ octeți.}$$

Memoria calculatoarelor din familia *ZX-SPECTRUM* are o capacitate de *65536* cuvinte (octeți), sau *64* Ko, deoarece

$$65536/1024 = 64 \text{ Ko} \quad (\text{unde } 65536 = 1024 \times 64 = 2^{10} \times 2^{6} = 2^{16})$$

În acest fel se poate afla adresa oricărui cuvînt (octet) de memorie întrucît adresa este *„locul de ordine minus 1"* (numerotarea octeților se face de la *0* la *65536*). Astfel, al *10000*-lea octet are adresa *9999*.

În fig. 2.1. este prezentată schema memoriei calculatorului *ZX SPECTRUM*.

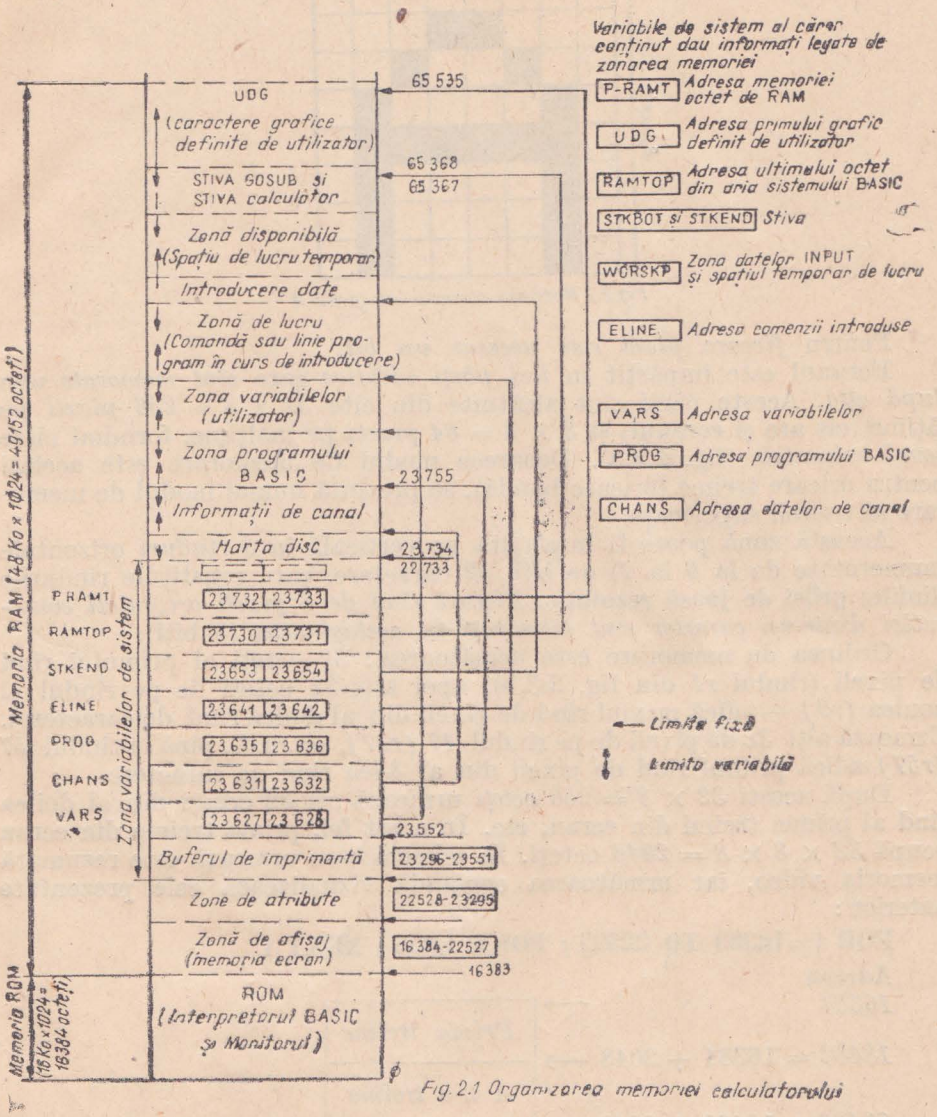


Fig. 2.1 Organizarea memoriei calculatorului

● Între **adresele 0 și 16383** se află memoria *ROM* nevolatilă unde se găsesc *monitorul* și *interpretorul BASIC*. (memoria *ROM* are *16* Ko).

● De la **16384 la 22527** se află *memoria video* (memoria ecran) sau bufferul de afișare, unde sînt reținute informațiile privitoare la fiecare ***pixel*** din ecran. Se precizează că fiecare caracter se realizează pe o matrice de

8 × 8 pixeli

(de la *PICture CELL* = celulă picturală). Pentru exemplificare, în fig. 2.2. se prezintă matricea caracterului grafic *A*.

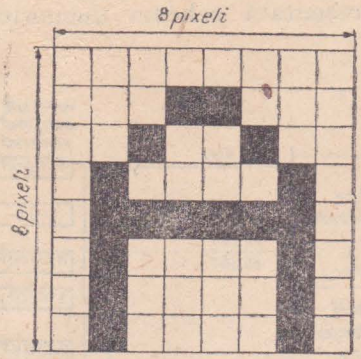


Fig.2.2 Matricea caracterului grafic A

Pentru *fiecare pixel este necesar un bit.*

Ecranul este împărțit în *trei părți identice* care *sînt memorate una după alta.* Aceste părți sînt alcătuite din cîte $32 \times 8 = 256$ *pixeli* pe lățime (cît are și ecranul) și $8 \times 8 = 64$ *pixeli* pe înălțime, formînd niște *benzi orizontale* (fig. 2.3,a). Deoarece modul de memorare este același pentru oricare treime (oricare bandă), se prezintă numai modul de memorare al treimii superioare.

Această zonă poate fi împărțită pe verticală în *8 rînduri* orizontale (numerotate de la *0* la *7*) de cîte *32* caractere, care constituie rîndurile (liniile) **grilei de joasă rezoluție**. *Fiecare rînd de 8 pixeli orizontali consecutivi dintr-un caracter sînt memorați în același octet* (8 biți).

Ordinea de memorare este următoarea: *32* octeți ai primului rînd de pixeli (rîndul *r1* din fig. 2.3,b), apoi alți *32* octeți de pe rîndul al noulea *(r9)* — adică primul rînd de pixeli din al doilea rînd de caractere). Urmează alți *32* de pixeli de pe rîndul *17 (r17)*, ș.a.m.d., pînă la rîndul *57 (r57)* adică primul rînd de pixeli din al *8*-lea rînd de caractere.

După acești $32 \times 8 = 256$ *octeți* urmează cei *32* octeți din al doilea rînd al primei treimi din ecran, etc. În acest fel, prima treime din ecran ocupă $32 \times 8 \times 8 = 2048$ octeți. În schema care urmează este rezumată memoria video, iar următoarea comandă vizualizează cele prezentate anterior:

FOR i=16383 TO 22527 : POKE i,255 : NEXT i

Adresa

16384 → *Prima treime*

18432 = 16384 + 2048 → *A 2-a treime*

20480 = 18432 + 2048 → *A 3-a treime*

22528 = 20480 + 2048 →

● De la **22528 la 23295** se află *768* de octeți care memorează fiecare *atributele unui caracter* de 8×8 *pixeli* pe ecran. Pentru a memora imaginea de pe ecran sînt necesari

3 treimi × *2048* octeți = *6912* octeți.

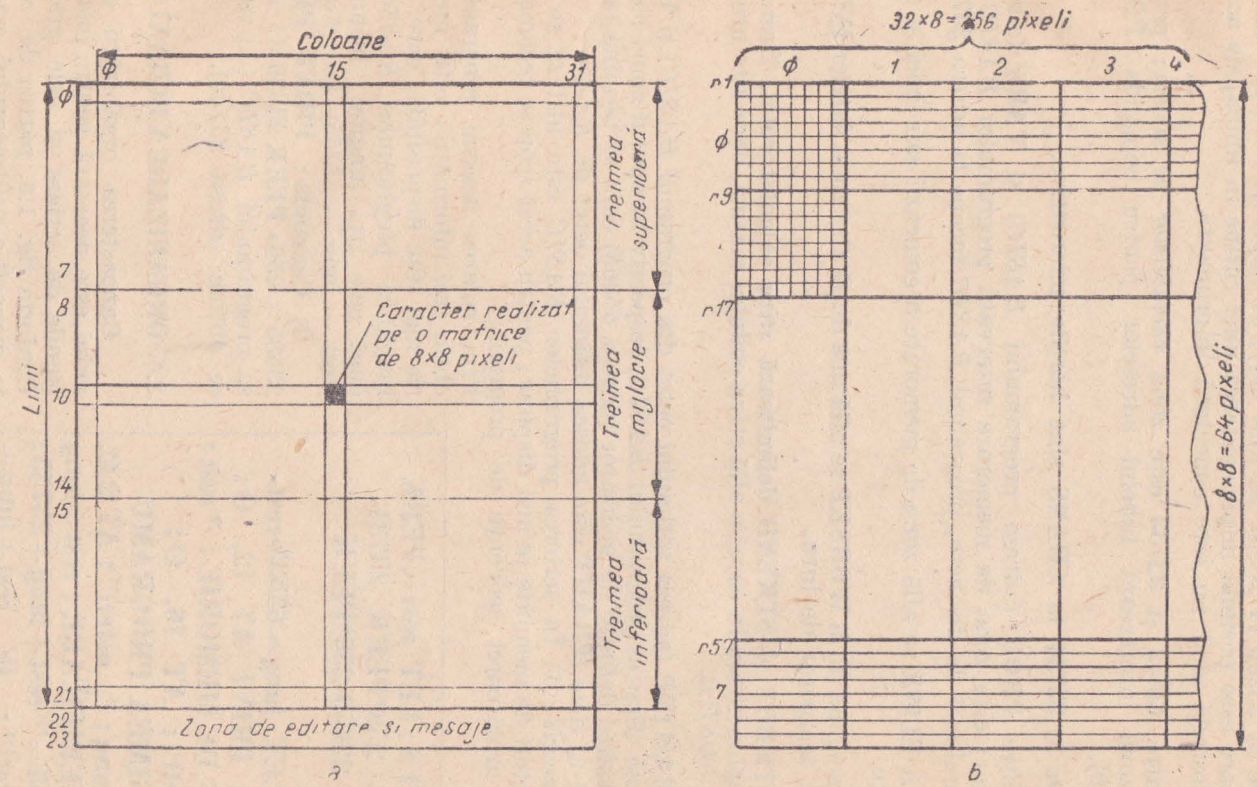


Fig 2 3 Ordinea de memorare a memoriei ecran (memoria video sau zona de afișare)

Atributele definesc modul în care va fi afișat fiecare caracter, adică felul culorii hîrtiei (**PAPER**), al culorii cernelei (**INK**), contrastul (**BRIGHT**) și pulsarea sau clipirea (**FLASH**).

● De la 23296 la 23551 se află *bufferul imprimantei (256 octeți)*, zonă în care este posibilă implementarea unei rutine în limbaj de asamblare, folosibilă cînd nu este conectată imprimanta.

● Între **23552** și **23733** este zona *variabilelor de sistem*; pentru această zonă interesează tabelul adreselor pentru comanda **POKE** (v. cap. 9).

● De la **23734** la **CHANS** sînt hărțile microdrive.

● Între **PROG** (adresa programului *BASIC*) și **VARS** (adresa variabilelor) este zona de memorare rezervată programului *BASIC* (în absența unității de dischete, programul *BASIC* începe la adresa **23755**).

● La **ELINE** se află zona de memorare a comenzii sau liniei în curs de editare.

● Începînd de la **WORSKP** se află una după alta zona datelor **INPUT** și spațiul temporar de luru.

● **STKBOT** și **STKEND** delimitează *stiva* calculatorului. Urmează o zonă suplimentară în care se află stiva codului mașină, care este urmată de stiva *GOSUB*.

● **RAMTOP** (adresa ultimului octet din programul *BASIC*) și **UDG** delimitează sfîrșitul programului *BASIC*, respectiv începutul zonei caracterelor grafice definite de utilizator (adresa *65368*) care se termină odată cu *RAM*-ul fizic (**PRAMT** este adesea ultimului octet de *RAM*).

Observații : 1) În scrierea programelor *BASIC* este util să se știe de cîți octeți de memorie se mai dispune; pentru acest scop se recomandă scrierea următoarei secvențe de program :

```
9998 CLS : LET rest=PEEK
2373 Ø+256*PEEK 23731-
PEEK 23653-256*PEEK
23654
9999 LET lung=65535-rest-
23755: PRINT AT 12, 'Ø;
"REST DE MEMORIE : "'rest;
" octeti"; AT 16, Ø;
"LUNGIMEA PROGRAMU
LUI"'lung; " octeti"; AT 2 Ø,
Ø; "VERIFICARE : rest+lung
+23755="' rest+lung+23755;
" octeti": IF rest+lung+
+23755=65535 THEN PRINT
AT 21, 14; FLASH 1; "O.K."
```

Prin rularea acestui program se obțin trei informații : cîtă memorie mai este disponibilă, care este lungimea programului *BASIC* și dacă cele trei lungimi însumate (rest + lung + *23755*) fac *65535*.

2) Comanda : **PRINT PEEK** 23635 + 256* **PEEK** 23636 dă adresa programului *BASIC* (de regulă se obține adresa *23755*).

2.6. ORGANIZAREA ECRANULUI

Cunoașterea organizării ecranului este necesară pentru instrucțiunile de scriere și cele grafice. Conform fig. 1.6, partea de ecran la dispoziția utilizatorului este un dreptunghi ce dispune de :

— *22 linii* (numerotate de la Ø la *21*);

— *32 coloane* (numerotate de la ∅ la *31*).

Liniile cu numerele *22* și *23* sînt folosite de calculator pentru scrierea de mesaje și editare. În fig. 2.4 este indicată și organizarea ecranului în pixeli.

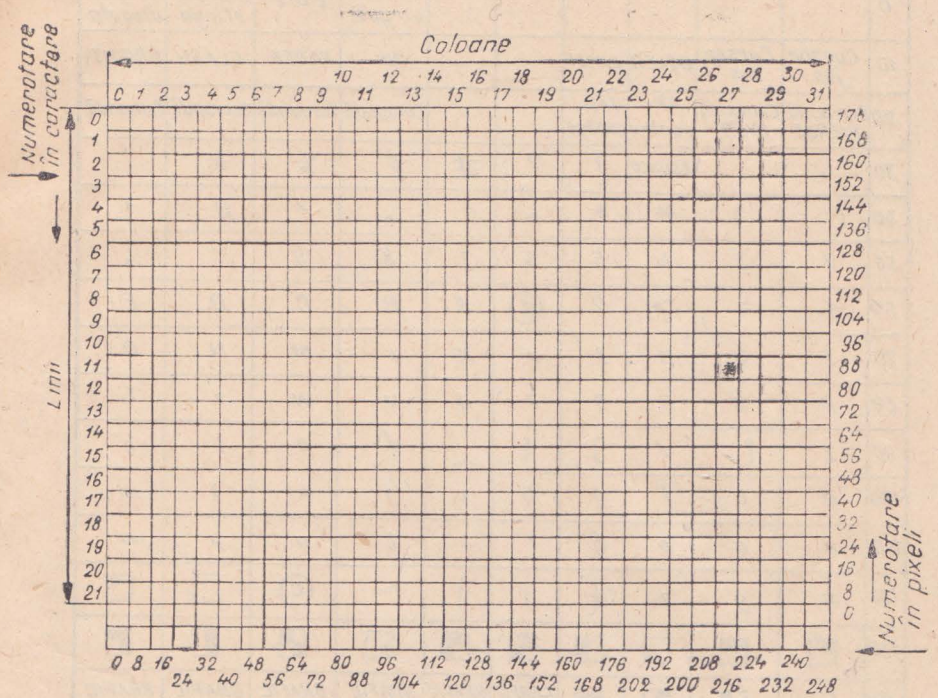


Fig. 2.4 Organizarea ecranului (în caracteri și pixeli)

2.7. CODURILE CARACTERELOR

Calculatoarele din familia *ZX SPECTRUM* folosesc codul *ASCII (American Standard Code Information Interchange)* pentru litere și cifre, conform tabelului 2.1.

Examinînd acest tabel se constată că literele și cifrele au codurile cuprinse între *48—57* (cifrele), *65—90*(majuscule), *92—122* (minuscule), *33—47* și *58—64*, *91—96* și *123—127* diverse semne.

Pentru a afla codul unui caracter trebuie tastată comanda

PRINT CODE "caracterul"

De *exemplu:* **PRINT CODE "A"** determină calculatorul să afișeze *65*.

Setul de caractere este afișat de calculator folosind comanda

```
FOR I = 32 TO 255 : PRINT CHR$ I; : NEXT I
```

Comanda **PRINT CHR$** număr afișează caracterul corespunzător (de ex: **PRINT CHR$** 65 afișează *A*).

Tabelul 2.1 Codurile caracterelor

	0	1	2	3	4	5	6	7	8	9
0							Print virgulă	EDIT	Prompter stinga	Prompter dreapta
10	Cursor jos	Cursor sus	DELETE	ENTER			INK	PAPER	FLASH	BRIGHT
20	Comandă INVERSE	Comandă OVER	AT-control	TAB control			Comandă	Comandă	Comandă	Comandă
30			Espace	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	Ø	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	ⓐ	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	/	]	↑	_	£	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	ⓒ	□	▗
130	▘	▀	▖	▌	▞	▛	▗	▚	▌	▛
140	▄	▙	▟	■	GRAFIC A	GRAFIC B	GRAFIC C	GRAFIC D	GRAFIC E	GRAFIC F
150	GRAFIC G	GRAFIC H	GRAFIC I	GRAFIC J	GRAFIC K	GRAFIC L	GRAFIC M	GRAFIC N	GRAFIC O	GRAFIC P
160	GRAFIC Q	GRAFIC R	GRAFIC S	GRAFIC T	GRAFIC U	RND	INKEY$	PI	FN	POINT
170	SCREEN$	ATTR	AT	TAB	VAL$	CODE	VAL	LEN	SIN	COS
180	TAN	ASN	ACS	ATN	LN	EXP	INT	SQR	SGN	ABS
190	PEEK	IN	USR	STR$	CHR$	NOT	BIN	OR	AND	<=
200	>=	<>	LINE	THEN	TO	STEP	DEF FN	CAT	FORMAT	MOVE
210	ERASE	OPEN	CLOSE	MERGE	VERIFY	BEEP	CIRCLE	INK	PAPER	FLASH
220	BRIGHT	INVERSE	OVER	OUT	LPRINT	LLIST	STOP	READ	DATA	RESTORE
230	NEW	BORDER	CONTINUE	DIM	REM	FOR	GO TO	GO SUB	INPUT	LOAD
240	LIST	LET	PAUSE	NEXT	POKE	PRINT	PLOT	RUN	SAVE	RANDOMIZE
250	IF	CLS	DRAW	CLEAR	RETURN	COPY				

2.8. MESAJELE DE EROARE

Sînt transmise de calculator în zona de editare (ultimile două linii ale ecranului, adică liniile *22* și *23*) și conțin informații despre cauzele întreruperii programului. Aceste mesaje au structura:

1	2,	3 : 4

unde: *1* reprezintă un *număr* sau o *literă* ce semnifică *codul erorii*;
2 descrie sumar eroarea în limba engleză;
3 indică numărul de linie din program unde s-a depistat eroarea;
4 conține numărul de ordine al instrucțiunii ce conține eroarea.
Despărțirea dintre cîmpurile *3* și *4* se face prin două puncte *(:)*
Exemple:

2 Variabile not found, 30 :1

7 RETURN without GOSUB, 35 :1

Lista mesajelor de eroare este indicată în tabelul 2.2.

Tabelul 2.2. **LISTA MESAJELOR DE EROARE**

Mesajul	Semnificația
1 NEXT with out FOR	Instrucțiunea *NEXT* nu are *FOR*
2 Variable not found	Programul are variabile nedefinite anterior cu *LET, READ, INPUT, FOR-NEXT* sau *DIM*
3 Subscrit wrong	La variabilele tablou indicii depășesc dimensiunile declarate cu *DIM*, sau la variabilele șir se depășește lungimea șirului de caractere
4 Out of memory	La evaluarea expresiilor sau la instrucțiuni *FOR-NEXT, DIM, LOAD, MERGE*, cînd nu mai este loc în memorie
5 Out of screen	Instrucțiunea *PRINT AT* încearcă scrierea în liniile *22* sau *23*, sau instrucțiunea *INPUT* încearcă introducerea a mai mult de *23* de linii
6 Number too big	La evaluarea expresiilor cînd rezultă un număr mai mare ca 10^{38}
7 Return with out GOSUB	S-a depistat o instrucțiune *RETURN* fără să fi fost apelată o subrutină prin *GOSUB*
8 End of file	Sfîrșit de fișier la lucrul cu fișiere pe casetă
9 Stop statement	După o comandă/instrucțiune *STOP*,
A Invalid argument	Parametrii incorecți la funcțiile *SQR, LN, ASN, ACS, USR*

B Integer out of range	Parametrii în afara domeniului în instrucțiunile *RANDOMIZE*, *POKE*, *PEEK*, *PLOT*, *DRAW*, *CIRCLE*, *USR*
C Nonsens in BASIC	Cînd instrucțiunea nu este validă
D BREAK-CONT repeats	Se întrerupe programul cu *BREAK* sau după apariția mesajului „*scroll*" se tastează *SPACE* sau *STOP*
E Out of DATA	Sînt mai puține date în instrucțiunea *DATA*
F Invalid file name	Instrucțiunea *SAVE* are un nume mai mare ca *10* caractere.
G No room for line	Memorie plină nefiind posibilă introducerea unei noi linii.
H STOP in INPUT	În datele de intrare cu *INPUT* se află un *STOP* sau s-a tastat *STOP*.
I FOR with out NEXT	Un ciclu (buclă) începe cu *FOR* dar nu se termină cu *NEXT*.
J Invalid I/O device	Operații greșite de cuplare a perifericelor la calculator.
K Invalid colour	Parametrii greșiți în instrucțiunile *INVERSE* și *OVER*
L BREAK into program	Apare o întrerupere *BREAK* între două linii de program.
M RAMTOP no good	Parametrul lui *CLEAR* este prea mare sau prea mic.
N Statement lost	Se încearcă un salt prin *RETURN*, *NEXT*, *CONTINUE* la o instrucțiune inexistentă.
O Invalid stream	Operații eronate cu perifericele.
P FN whith out DEF	Se încearcă utilizarea unei funcții nedefinite anterior în program.
Q Parameter error	Parametrii de apelare a unei funcții sînt diferiți ca tip de cei fixați la definirea funcției.
R Tape leading error	Încărcarea unei înregistrări eronate

Atunci cînd nu există nici o eroare sau se încearcă un salt la o linie cu număr mai mare decît numărul maxim existent în program, mesajul este

Ø OK

2.9. PROBLEME

P2.1. Să se indice etichetele și textele incorect scrise:

a) *100*; b) *0100*; c) *1020.5*; d) *"Conținut"*; e) *"Mircea"*.

Răspuns: *c)*

P2.2. Care dintre constantele următoare sînt identice și care sînt incorect scrise: a) *1* ∗ *e3*; b) *5000*; c) *5e3*; d) *.5E4*.

Răspuns: incorect scrisă este constanta *a)*; identice sînt *b), c), d).*

P2.3. Care dintre variabilele următoare sînt simple, indexate sau şir: a) *suma*; b) *zI*; c) *V(3)*; d) *X(5, m * n)*; e) *C$*.

Răspuns: Variabilele simple *a), b)*; variabilele indexate *c), d)*; variabilă şir *e).*

P2.4. Să se scrie următoarele expresii: a) $-cos^4\alpha\ sin^3\alpha/(p+1)$;

b) $h = 2L^2\sigma/3Ef$; c) $w = x^2 - 2x\cos\alpha + 1$; d) $v = \sqrt[4]{\dfrac{5760\ F_{max}aL}{\pi^2G\theta}}$;

e) $z = \sqrt[3]{a^2 + b^3\sin^3 x}$; f) $u = \dfrac{a^{0,7} + 2\sqrt{3b + 2a^{0,7}\cos C + 1}}{a^{0,7}(b + a^{0,7}\cos C) + 2}$.

Răspuns: a) — COS(ALFA) ↑ 4 * SIN(ALFA) ↑ 3/(p+1);
b) h=(2*L*L*sigma)/3*E*f; c) w=x*x−2*x*COS(ALFA)+1;
d) v=((5760*FMAX*a*L)/PI*PI*G*TETA) ↑ (1/4);
e) z=((a ↑ 3+ b↑ 3*SIN*x ↑ 3) ↑ (1/3)); f) u=(a ↑ .7+2*((3*b+
+2*a ↑ .7*COSC+1) ↑ (1/2)))/(a ↑ .7*(b+a ↑ .7*COSC+2)).

Se vor reţine următoarele reguli: 1) $\sqrt[n]{a}$ se scrie a↑(1/n) [v. d), e), f)]; 2) un numitor format dintr-o însumare algebrică se scrie **între paranteze** [v. a) şi f)]; 3) o funcţie trigonometrică se poate scrie sau nu cu argumentul între paranteze [v. a); c); e); f)]; 4) se preferă scrierea x^2 sub forma $x * x$ [v. c)]: 5) *puterea* funcţiei trigonometrice se scrie *după argumentul ei* (v. a), e); 6) *puterea argumentului* funcţiei trigonometrice se scrie în *interiorul perechii de paranteze* care conţin argumentul [ex: **sin(A + 2)²** se scrie **SIN((A + 2) ↑ 2)**].

P2.5. Care este: a) comanda ce trebuie transmisă calculatorului pentru a afişa codul semnului mai mic (<); b) codul instrucţiunii *TAB*; c) comanda pentru a afişa litera *K*.

Răspuns: *a)* **PRINT CODE "<"**; *b)* **173** (v. tabelul 3.1); *c)* **PRINT CHR$** 75. (v. tabelul *3.1*).

Capitolul 3

INSTRUCTIUNI/COMENZI PENTRU COMENTARII, EDITAREA, STERGEREA, LANSAREA, OPRIREA, CONTINUAREA, ÎNCARCAREA, SALVAREA PROGRAMELOR SI STERGEREA ECRANULUI TV

3.1. NOȚIUNILE „INSTRUCȚIUNE" ȘI „COMANDA"

Instrucțiunea și comanda sînt definite printr-un **cuvînt cheie** provenit din limba engleză și înscris întreg sau prescurtat pe tastele calculatorului (exemple : **PRINT, LET, RUN, RAND, INV,** ș.a.). Rezultă că instrucțiunea și comanda pot avea *aceeași denumire*, dar ele se deosebesc :

— principial după modul de memorare- execuție și păstrare în memorie ;

— formal după numerotare (eticheta este absentă la comenzi). Astfel :

a) **Instrucțiunea :**

— are sintaxa

nr. linie	cuvînt cheie	corp instrucțiune

— se memorează imediat ca linie-program după scriere și acționarea tastei CR ;

— se execută de cîte ori se dorește acționînd tasta **RUN** ;

— se păstrează în memorie pînă la ștergerea ei sau anularea instrucțiunii sau a programului (tastînd **NEW**).

Exemplu: **10 PRINT "Basic"**

b) **Comanda :**

— are sintaxa

cuvînt cheie	corp comandă

— se scrie și se execută o singură dată prin apăsarea tastei CR ;

— nu se păstrează în memorie, ceea ce înseamnă că execuția ei nu poate fi reluată decît printr-o nouă scriere a comenzii.

Exemplu: **PRINT "Basic"**

Observații: 1) Corpul instrucțiunii/comenzii poate fi alcătuit din constante, expresii, condiții, mesaje, șiruri de caractere, funcții sau combinații ale acestora, care sînt separate prin caracterele virgulă, punct și virgulă sau apostrof (al căror rol va fi precizat la studiul instrucțiunilor).

2) Apăsarea tastei *CR* (acronimul de la *Carriage Return* = retur de car) este **obligatorie** la sfîrșitul oricărei comenzi și la terminarea fiecărei linii-program. În ultimul caz, această tastare determină deplasarea liniei-program din spațiul de editare (liniile *22* și *23* ale ecranului) — situat în partea inferioară a ecranului — către partea superioară a acestuia, unde se afișează toate liniile program introduse în memoria calculatorului.

3) Calculatorul poate afișa pe ecran maximum *22* linii conținînd cel mult tot atîtea linii-program, cu condiția ca o linie-program să nu aibă mai mult de 32 **caractere**. Dacă programul este mai lung de *22* linii, pe ecran se vor afișa cele 22 de linii ale primei pagini, după care calculatorul va emite mesajul

scroll?

care semnifică continuarea programului pe pagina următoare. În acest scop se acționează tasta *CR*.

3.2. INSTRUCȚIUNEA REM PENTRU COMENTARII

Comentariile sînt introduse prin instrucțiunea **REM** și se referă fie la întregul program (de exemplu: titlul programului), fie la un anumit grup de linii-program (de exemplu: se indică ce face acest grup de linii), cu scopul înțelegerii, utilizării și depanării mai ușoare a programului.

Sintaxa: **nr. linie REM comentariu**

Exemplu: **10 REM Verificarea compatibilității problemei.**

Efect: introduce comentarii într-un program care sînt **ignorate** de calculator.

Observații: 1) La o instrucțiune multiplă trebuie să fie ultima instrucțiune. Exemplu: **10 LET** *a* = 2 : **LET b** = *3* : **REM inițializarea variabilelor**

2) Calculatorul ignoră complet tot ceea ce urmează după **REM**

Se recomandă folosirea instrucțiunii **REM** în orice parte a programului unde amplasarea ei este necesară, dar se va reține că ocupă memorie și consumă timp.

3.3. COMENZI PENTRU EDITAREA ȘI ȘTERGEREA UNEI LINII DE PROGRAM

O linie-program se scrie în **spațiul de editare** și după apăsarea tastei *CR* ea se deplasează în partea superioară a ecranului unde se afișează toate liniile programului în **ordinea crescătoare a numerelor de linie (etichetelor)**, deoarece **execuția programului se face în această ordine.**

Este însă posibil ca în cursul redactării unui program să apară necesitatea *ştergerii* unor caractere, cuvinte cheie sau linii, precum şi *introducerea* altor instrucţiuni. În aceste cazuri se folosesc comenzile **DELETE** şi **EDIT**.

3.3.1. Cursorul de program (prompterul)

În afară de cursorul clipitor care indică poziţia curentă de scriere şi modul de lucru *(K, L, C, E* sau *G)*, calculatorul mai foloseşte un *cursor de program* şi anume *prompterul* >, care indică în stînga părţii de sus a ecranului ultima linie de program introdusă (linia curentă).

Se pot aduce modificări :

— *în linia curentă*, care poate fi copiată în spaţiul de editare prin comanda **EDIT** (se apasă *CS* şi *1*) ;

— *la o altă linie*-program, cînd este necesară deplasarea prompterului > în dreptul acestei linii, acţionînd simultan tasta *CS* şi una din tastele cu cifre avînd săgeata corespunzătoare : în sus (↑) pe tasta *7* sau în jos (↓) pe tasta *6*.

O linie-program se şterge în întregime prin **tastarea numărului liniei** şi apoi *CR* (de exemplu : se doreşte ştergerea liniei *1000* din program ; pentru aceasta se tastează *1000* şi apoi *CR*).

3.3.2. Comanda DELETE

Această comandă serveşte pentru ştergerea unor caractere sau cuvinte cheie şi se obţine tastînd simultan *CS* şi *0*. Ştergerea are loc **poziţie cu poziţie de dreapta spre stînga** *pentru caracterele aflate în stînga prompterului*. De exemplu s-a realizat programul :

10 PRINT "Initiere in Basic"

20 PRINT "pe ZX SPECTRUM"

şi se doreşte ca linia *10* să aibă forma

10 PRINT "Basic".

În acest scop :

1) Se tastează *CS* şi *7* pentru ca prompterul > să fie adus pe linia de modificat (linia *10*).

2) Se tastează *CS* şi *1* (**EDIT**) pentru a se copia linia *10* în spaţiul de editare.

3) Se tastează *CS* şi *8* pentru a se deplasa cursorul clipitor *L* spre dreapta pînă la ultima literă a cuvîntului *„în"*.

4) Se apasă *CS* şi *0* (**DELETE**) şi se şterg, de la stînga spre dreapta, cuvintele *"Iniţiere în"*.

5) Se apasă *CR* şi linia modificată va apare în partea superioară a ecranului.

3.3.3. Comanda EDIT

Se utilizează pentru modificarea unei linii-program fără a se rescrie acționînd simultan tastele *CS* și *1*. Modul de lucru este următorul:

1) Se selectează linia-program de modificat tastînd *CS* și una din tastele *6* (↓) sau *7* (↑).

2) Se apasă *CS* și *1* și în spațiul de editare va apare copia liniei program selectate.

3) Se efectuează modificările necesare după cum urmează:

— pentru ștergerea unui caracter se deplasează cursorul clipitor acționînd *CS* și una din tastele *5* (←) sau *8* (→), pînă în dreptul caracterului de modificat și apoi se tastează *CS* și *0* (**DELETE**) pentru ștergerea acestui caracter;

— în locurile șterse se introduc, prin tastarea corespunzătoare, corecturile necesare;

— după epuizarea corecturilor se tastează *CR* pentru ca linia program modificată să-și reia locul în partea de sus a ecranului.

Observații: 1) Dacă programul este lung, poziționarea prompterului se face cu comanda

LIST număr linie și CR

iar aducerea liniei în spațiul de editare prin apăsarea tastelor *CS* și *1*.

2) O linie-program se poate modifica și prin *rescrierea ei cu același număr de linie.*

3.4. INSTRUCȚIUNEA/COMANDA NEW PENTRU ȘTERGEREA PROGRAMELOR

Ștergerea din memorie a unui program se face cu instrucțiunea sau comanda **NEW**.

Sintaxa: [**nr. linie**] **NEW** unde nr. linie este opțional

Exemple: **9000 NEW**

2000 IF INKEY$ = "." THEN NEW
NEW

Efect) determină ștergerea din memorie a programului curent pentru a permite introducerea unui nou program (ștergerea memoriei se face pînă la *RAMTOP*).

Observații) 1) Comanda **NEW** trebuie urmată de apăsarea tastei *CR* (**ENTER**).

2) Ștergerea din memorie a programului mai poate fi realizată apăsînd butonul de *RESET* sau întrerupînd alimentarea de la rețea a calculatorului.

3) Comanda **NEW** realizează în același timp ștergerea ecranului și repunerea culorilor în stare normală (**BORDER 7 : PAPER 7 : INK 0**).

3.5. INSTRUCȚIUNEA/COMANDA RUN PENTRU LANSAREA PROGRAMULUI

Declanșarea executării unui program se face cu instrucțiunea/comanda **RUN**.

Sintaxa: [nr. linie] **RUN** [nr. linie] unde nr. linie este opțional

Exemple: **1000 RUN 1**

RUN

Efect: lansează în execuție programul aflat în memoria calculatorului; cînd este indicat numărul de linie din dreapta instrucțiunii **RUN** execuția începe cu linia specificată, iar dacă numărul de linie lipsește execuția începe cu prima linie a programului.

Observații: 1) Dacă numărul de linie indicat după **RUN** nu este un număr întreg, el este rotunjit la valoarea întreagă cea mai apropiată.

De exemplu, dacă la programul:

```
10 PRINT "Turnul EIFEL"
20 PRINT "se află"
30 PRINT "la PARIS"
```

se dă comanda **RUN 20.1** pînă la **RUN 20.4** se vor executa liniile *20* și *30*, iar dacă se dă comanda **RUN 20.5** se va executa numai linia *30*. Desigur, exemplul are valoare didactică, deoarece comenzile **RUN** număr linie se dau cu numele de linie corect.

2) Dacă **RUN** este comandă, după apăsarea acestei taste se apasă și *CR*; în acest fel se declanșează executarea programului fie că are sau nu o linie program ce-l conține pe **RUN**.

3) **RUN** șterge ecranul și toate variabilele din memorie, efectuează **RESTORE**, șterge memoria pentru **GOSUB**, inițializează cursorul pentru **PRINT** la linia *0* și coloana *0* și originea grafică la coordonatele *0*, *0*.

3.6. INSTRUCȚIUNILE STOP, BREAK ȘI PAUSE PENTRU OPRIREA PROGRAMULUI

Execuția unui program poate fi întreruptă:
— *temporar* prin instrucțiunile/comenzile **STOP**, **BREAK** și **PAUSE**;
— *definitiv* cu instrucțiunea **STOP**.

3.6.1. Instrucțiunea STOP

Sintaxa: [nr. linie] **STOP** unde nr. linie este opțional.

Exemple: 9999 **STOP**

5000 IF A = 0 THEN STOP

Efect: La întîlnirea instrucțiunii **STOP** calculatorul suspendă execuția programului și afișează în spațiul de editare mesajul **STOP statement n : m**
unde *n* — nr. de linie și *m* — nr. instrucțiunii din linie care conține cuvîntul cheie **STOP**.

Observați : 1) Pot exista mai multe instrucțiuni **STOP** într-un program.

2) Execuția programului poate fi reluată cu una din comenzile :

CONTINUE sau **GO TO** *nr. linie* sau **RUN** *nr. linie*

(ultima șterge din memorie variabilele programului).

3.6.2. Comanda BREAK

Sintaxa : Se apasă simultan tastele *CS* și *SPACE*

Efect : determină oprirea execuției programului și apariția în spațiul de editare a mesajului :

L BREAK into program, n : m

(unde *n* — nr. linie și *m* — nr. instrucțiune din linia n)

3.6.3. Instrucțiunea PAUSE

Sintaxa : [**nr. linie**] **PAUSE p** unde nr. linie este opțional, iar $p \in [0; 65535]$; pentru $p = 0$ programul este întrerupt definitiv, iar pentru $p = 65535$ timp de *22* minute.

Exemple : **1000 PAUSE** 4e4

2000 IF INKEY$ = "." THEN PAUSE 200

Efect : realizează o pauză în execuția programului a cărei durată depinde de numărul *p*, ecranul rămînînd neschimbat. (durata pauzei *p/50* secunde)

Observații : 1) **PAUSE 50** determină o pauză de circa o secundă.

2) Pauza obținută poate fi scurtată apăsînd orice tastă, cu excepția tastelor *CS*, *SS* și *SPACE*.

3) Instrucțiunea **PAUSE** poate fi utilizată și pentru animarea unei imagini, reproducerea pauzelor muzicale, ș.a.

3.7. COMANDA CONTINUE

Se folosește pentru reluarea execuției unui program întrerupt prin **STOP** sau **BREAK**.

Sintaxa : **CONTINUE**

Efect : relansează programul în execuție întrerupt cu **STOP** sau **BREAK**, de la linia la care s-a produs întreruperea.

Observație : Instrucțiunea/comanda **STOP** împreună cu comanda **CONTINUE** reprezintă o modalitate de lucru pentru punerea la punct a programelor complexe sau de dimensiuni mari, prin rularea eșalonată a liniilor sau a unui grup de linii.

3.8. INSTRUCȚIUNILE/COMENZILE LOAD ȘI MERGE PENTRU ÎNCĂRCAREA PROGRAMELOR

Calculatorul poate fi cuplat cu un casetofon cu scopul de a utiliza banda magnetică a casetei drept suport pentru stocarea informațiilor. Un program existent pe o asemenea bandă poate fi introdus în memoria calculatorului utilizînd instrucțiunile/comenzile **LOAD** și **MERGE**.

3.8.1. Instrucțiunea/Comanda LOAD

Sintaxa: [**nr. linie**] **LOAD** "[**nume program**]", unde *nume* — șir de maximum *10* caractere, care poate fi opțional, ca și numărul de linie.

Exemple: **LOAD "MECANISME"**

LOAD " "

9000 LOAD "HARDWARE"

Efect: șterge vechiul program din memorie, caută pe banda magnetică înregistrarea cu numele precizat — afișînd numele tuturor programelor întîlnite — și cînd a găsit programul cerut afișează mesajul

Program : numele programului

(de exemplu : **Program : MECANISME**).
Apoi încarcă programul în memoria calculatorului și dacă programul nu se autolansează afișează în spațiul de editare mesajul

0 O.K.

Observați : 1) Forma **LOAD " "** *încarcă primul program întîlnit pe bandă magnetică.*

2) Cu comanda arbitrară **LOAD "p"** se determină succesiunea programelor înregistrate pe bandă (**DIRECTORY**), presupunînd că nu există nici un program cu numele "*p*" pe bandă.

3) Forma **LOAD "nume" CODE** încarcă un program scris în limbaj de asamblare, la adresa unde acesta a fost salvat. Dacă forma este **LOAD "nume" CODE adresă** (exemplu : **LOAD "udg" CODE 60000**), atunci se încarcă rutina în limbaj de asamblare la adresa indicată în comanda **LOAD**, adresă care poate fi *diferită de cea de salvare.* Atunci cînd forma este **LOAD "nume" CODE adresă, nr octeți** (de exemplu : **LOAD "udg" CODE** 65368, 168), calculatorul *testează și lungimea programului în limbaj de asamblare.*

4) Forma **LOAD "nume" SCREEN$** (echivalentă cu **LOAD "nume" CODE** 16384, 6912) determină încărcarea în memorie și afișarea pe ecran a unei *mire* (desen, text sau combinație desen-text).

5) Forma **LOAD "nume1" DATA nume2( )** determină încărcarea unui *șir de caractere*, unde *nume1* este numele sub care șirul se găsește

înregistrat pe banda magnetică, iar *nume2* este noul nume sub care se va încărca șirul în memorie. De exemplu : **LOAD** "**text**" **DATA** var( ) determină calculatorul să caute pe banda magnetică șirul cu numele „*text*" și cînd îl găsește verifică dacă nu mai există în memorie un șir cu numele *var* (dacă îl găsește îl anulează) și încarcă șirul „*text*" sub numele de „*var*".

În legătură cu folosirea instrucțiunii/comenzii **LOAD** trebuie precizată organizarea programului pe banda magnetică. Astfel, programul are o primă porțiune de scurtă durată numită **header**, care are întotdeauna o lungime de *17 octeți*; ea conține informații despre program privind: *titlul, tipul programului* (în *BASIC* sau în limbaj de asâmblare), *lungimea acestuia în octeți și adresa de înregistrare*. Cea de a doua porțiune este *programul propriu zis*.

Cele două părți (*header*-ul și *programul*) au înainte de începutul zonei de date o porțiune unde *sunetul în timpul încărcării este uniform*, ceea ce indică începerea unui bloc de program. În timpul citirii și înregistrării lor **BORDER**-ul (marginea) ecranului este în *dungi roșu-albastru* deschis (read-cyan), iar în restul programului dungile sînt *albastru-galbene* (blue-yellow). Aceste precizări sînt valabile numai pentru programele care folosesc rutina pentru casetofon rezidentă în **ROM**.

Se menționează că sînt programe la care **BORDER**-ul în timpul încărcării programului poate fi *static* (unele jocuri) sau să fie blocuri *fără boader*, după cum pot exista blocuri la care lățimea dungilor pe **BORDER** să fie modificate.

3.8.2. Comanda MERGE

Este o comandă folosită numai pentru programele scrise în BASIC cu scopul fuzionării unui program de pe banda magnetică cu altul ce se află în memoria calculatorului (sau ambele pe bandă)

Sintaxa: **MERGE** "[nume]" unde *nume* este opțional

Exemple: **MERGE** "pene"

MERGE" "

Efect: încarcă în memoria calculatorului o înregistrare de pe banda magnetică *fără a șterge vechiul program stocat în memorie*. Comanda **MERGE** anulează din programul aflat în memorie, înainte de începerea transferului, doar acele variabile și numere de linii existente în programul ce urmează a fi încărcat.

Observații: 1) Comanda **MERGE**" " determină încărcarea primului program *BASIC* întîlnit pe casetă.

2) Un program salvat pe banda magnetică cu auto lansare, nu pornește automat în execuție la încărcarea cu comanda **MERGE**.

3.9. INSTRUCȚIUNILE /COMENZILE SAVE ȘI VERIFY PENTRU SALVAREA PROGRAMULUI ȘI VERIFICAREA ÎNREGISTRĂRII SALE

3.9.1. Instrucțiunea/Comanda SAVE

Sintaxa: [nr. linie] **SAVE** "**nume**" [**LINE** x], unde *nr. linie* și *LINE x* sînt opționale (*nume* — numele programului format din maximum *10* caractere, iar *x* — număr de linie)

Exemple: 9999 **SAVE** "**ANGRENAJE**" **LINE** 1

SAVE "**mira**" **SCREEN$**

SAVE "**SCREEN**" **CODE 50000, 6912**

SAVE "**matrice**" **DATA matrice ()**

Efect) salvează programul din memoria *RAM* pe banda magnetică cu numele precizat. După darea comenzii calculatorul afișează în spațiul de editare mesajul

Start tape then press any key

În acest moment se apasă tasta *START* și tasta de înregistrare ale casetofonului (simultan) și se apasă orice tastă a calculatorului. La terminarea unei înregistrări corecte se afișează mesajul *O.K.*, iar dacă înregistrarea nu s-a efectuat se afișează în spațiul de editare

R Tape loading error

În acest din urmă caz se reia înregistrarea.

Observații) 1) Forma **SAVE** "**nume**" salvează pe bandă programul *BASIC* din memoria *RAM* și toate variabilele sale cu valorile curente. Dacă aceste variabile se inițializează prin program, atunci se recomandă folosirea instrucțiunii **CLEAR** înainte de salvarea programului. De exemplu :

9999 CLEAR : SAVE "**nume**"

2) Forma **SAVE** "**nume**" **LINE** x, salvează programul *BASIC* ca în cazul anterior care va *porni automat de la linia cu eticheta x.* (ex : **SAVE** "**MP**" **LINE 10**).

3) Forma **SAVE** "**nume**" **SCREEN$** salvează pe bandă conținutul memoriei ecran inclusiv atributele; ea este echivalentă cu comanda **SAVE** "**nume**" **CODE 16384, 6912.**

4) Forma **SAVE** "**nume1**" **DATA nume2 ( )** salvează conținutul șir de caractere unde *nume1* este noul nume sub care va fi înregistrat șirul iar *nume2* este numele șirului din memoria ce va fi salvat. De exemplu :

SAVE "**MIRA**" **DATA v( )**

înregistrează șirul *v* pe banda magnetică sub numele "*MIRA*".

5) Forma **SAVE "nume" CODE adresa, nr. octeți** salvează o rutină în limbaj de asamblare la adresa indicată, avînd o lungime egală cu numărul de octeți precizat (exemplu: **SAVE "voce" CODE 60000, 801**)

3.9.2. Instrucțiunea/Comanda VERIFY

Sintaxa: **[nr. linie] VERIFY "[nume]"**, unde *nr. linie* și *nume* sînt opționale.

Exemple: 9999 **VERIFY "Lagare"**

VERIFY" "

VERIFY" "CODE

VERIFY "efecte" CODE 50000, 1950

VERIFY "MATRICE" DATA v( )

Efect: verifică înregistrarea salvată pe banda magnetică cu instrucțiunea/comanda **SAVE**. În acest scop se procedează astfel:

— se derulează înapoi banda magnetică;

— se dă comanda **VERIFY** corespunzătoare tipului de program salvat, urmată de apăsarea tastei *CR*.

Dacă înregistrarea este corect efectuată calculatorul va afișa mesajul

Program: numele programului

0 O.K.

În caz contrar mesajul afișat va fi

R Tape loading error

și se procedează la o nouă salvare a programului.

La lansarea comenzii **VERIFY** (ca și la **LOAD** sau **MERGE**), calculatorul așteaptă apariția unui header pe bandă; cînd a găsit acest header citește informațiile pe care le conține și afișează:

— pentru programele *BASIC:* **Program: titlu program** (exemplu: *Program: vectori);*

— pentru programele în limbaj de asamblare: **Bytes: titlu program** (exemplu: *Bytes: udg1);* în mod analog pentru screen-uri;

— pentru o matrice numerică: **Number array: nume program** (exemplu: *Number array: mat);*

— pentru o matrice de caractere: **Character array: nume program.** (exemplu: *Character array: cod).*

Observații: 1) **Forma VERIFY" " [CODE]** verifică primul program de pe banda magnetică (exemple: **VERIFY" "** sau **VERIFY" " CODE).**

2) Forma **VERIFY" "SCREEN$** nu este acceptată de calculator.

3.10. INSTRUCȚIUNEA/COMANDA CLS PENTRU ȘTERGEREA ECRANULUI

Sintaxa: [nr. linie] **CLS**, unde *nr. linie* este opțional.

Exemple: **100 CLS**
2000 INPUT a : CLS
CLS

Efect: determină ștergerea ecranului, programul și variabilele sale rămînînd neschimbate.

Observații) 1) Ecranul capătă culoarea hîrtiei (**PAPER**-ului) avută în momentul întîlnirii instrucțiunii/comenzii **CLS**.

2) Ștergerea ecranului se poate realiza și cu ajutorul instrucțiunilor comenzilor **RUN** și **CLEAR** (care au însă alte roluri).

3) După execuția **CLS** este posibilă folosirea în continuare a instrucțiunilor programului sau a comenzilor **RUN**, **EDIT**, **LIST**, etc.

3.11. PROBLEME

P3.1. Tastați programul

10 PRINT "Numele meu este"

20 PRINT "Alexandru Ionescu"

30 STOP

și corectați linia *10* astfel încît să devină

10 PRINT "Mă numesc"

P3.2. Ștergeți din memoria calculatorului programul de la problema **P3.1**, acționînd fie butonul de **RESET**, fie tastînd **NEW**.

P3.3. Retastați programul **P3.1** în care înlocuiți linia *30* prin **30 GO TO 10** și introduceți linia **25 POKE 23692, 255**
după care încercați oprirea rulării cu **BREAK**. și continuarea cu **CONTINUE**. Repetați oprirea și continuarea rulării programului.

P3.4. Înlocuiți în programul anterior linia *30* alternativ prin liniile

30 PAUSE 0

sau

30 PAUSE 100

urmată de linia

40 CLS

și observați efectele.

P3.5. Salvați pe casetă programul de la problema **P3.4**, verificați corectitudinea înregistrării și apoi înregistrați în memoria calculatorului programul salvat, tastînd **RUN** și **CR**.

P3.6. Scrieți și salvați programul

10 "PRINT Numele" ;

20 PRINT "Alexandru Ioneseu"

Apoi retastați calculatorul și scrieți liniile :

30 PRINT "Profesia : inginer"

40 STOP

În continuare derulați banda magnetică la începutul programului înregistrat și dați comanda **MERGE" "**. După afișarea mesajului *0 O.K., 0: 1*, dați comanda **LIST** și observați ce s-a întîmplat. Dați apoi comanda **RUN** urmată de *CR*.

Capitolul 4

INSTRUCȚIUNI PENTRU INTRODUCEREA DATELOR ȘI AFIȘAREA REZULTATELOR

4.1. INSTRUCȚIUNI PENTRU INTRODUCEREA DATELOR

Introducerea datelor necesare programului se poate face în două moduri :

— în momentul *scrierii programului* folosind instrucțiunile **READ, DATA** și **RESTORE**;

— în momentul *execuției programului* cu instrucțiunea **INPUT.**

4.1.1. Instrucțiunile READ ȘI DATA

Aceste instrucțiuni se folosesc împreună (**READ** — a citi ; **DATA** — date).

a) INSTRUCȚIUNEA READ

Sintaxa: nr. linie **READ** nume1, nume2, . . ., unde *nume 1, nume2 . . .*, sînt numele variabilelor (SIMPLE SAU INDEXATE de tip numeric sau șir) de inițializat, separate prin virgulă ; valorile variabilelor sînt luate dintr-o instrucțiune **DATA** (sau mai multe).

Exemple:
```
10 READ A, B
20 READ T(1), T(2), T(3)
30 READ A10, A20
40 READ A$
```

Efect: determină transferul valorilor din instrucțiunea **DATA** în lista de variabile a instrucțiunii **READ.**

b) INSTRUCȚIUNEA DATA

Sintaxa: nr. linie **DATA** lista de valori (cu constante numerice sau alfanumerice).

Exemple: **10 DATA 10, 20, 30**

100 DATA "A", "B", "C"

3000 DATA "intrare", "ieșire"

Efect: introduce valori pentru variabilele programului; toate valorile introduse cu **DATA** formează o linie unică de date ce *se extrag în ordinea scrierii lor* pe măsură ce se execută instrucțiunile **READ**.

Observații: 1) Instrucțiunile **READ** și **DATA** se pot scrie *oriunde în program.*

2) În cazul cînd toate instrucțiunile **DATA** au fost epuizate fără să se poată atribui valori tuturor variabilelor din lista instrucțiunii **READ**, se afișează mesajul de eroare

E Out of DATA

3) Valorile utilizate pentru transfer trebuie să corespundă tipului variabilelor receptoare (adică valori numerice pentru variabile simple, valori literale pentru variabile alfanumerice).

Exercițiul 4.1. Să se atribuie variabilelor *X, Y, Z* valorile *4*; *2,75* și *— 3*. Progrramul este următorul:

10 READ X, Y, Z

20 DATA 4, 2.75, —3

Corectitudinea programului se poate verifica dînd comanda **RUN** și apoi comanda **PRINT X'Y'Z.**

Exercițiul 4.2. Se cere programul prin care variabilei *a$* i se atribuie șirul *„4 ianuarie 1930".*

10 READ a$

20 DATA "4 ianuarie 1930"

Cu comanda **PRINT** aS se verifică, după rularea programului prin apăsarea tastei **RUN**, corectitudinea acestuia.

Exercițiul 4.3. Următorul program:

10 READ N$, T$, A

20 DATA "MIRCEA POPOVICI"

30 DATA "NR. TELEFON", 219749

determină ca variabilele să capete valorile indicate și anume: *N$= „MIRCEA POPOVICI", T$=„NR. TELEFON"* și *A = 219749.*

4.1.2. Instrucțiunea RESTORE

Instrucțiunea/comandă **RESTORE** permite efectuarea recitirii datelor pentru a se evita rescrierea datelor identice pentru mai multe variabile.

Acest fapt este necesar atunci cînd programul folosește repetat aceleași date.

Sintaxa: **[nr. linie] RESTORE [n]**, unde *nr. linie* și *n* sînt opționali (*n* — număr linie unde se găsește o instrucțiune DATA)

Exemple: **100 RESTORE 200**

50 PRINT A : RESTORE

Efect: determină ca instrucțiunea **READ** următoare să citească datele din instrucțiunea **DATA** cu numărul de linie *n*.

Dacă numărul de linie *n* lipsește, se consideră implicit că are numărul de linie pe care se află prima instrucțiune **DATA**. (analog dacă la numărul de linie indicat nu se află instrucțiunea **DATA**).

Exemplul 4.4. Următorul program atribuie variabilelor *D* și *X*, *E* și *Y*, respectiv *F* și *Z*, valorile *1*, *2*, respectiv *3:*

```
10 READ D, E, F
20 RESTORE
30 READ X, Y, Z
40 DATA 1, 2, 3
```

Exemplul 4.5. Variabilele *a$*, *b* și *c$* capătă succesiv valorile "bec de␣", *40*, "␣*wați*", respectiv "*bec de*␣", *60*, "␣*wați*", folosind programul:

```
10 DATA "bec de␣"
20 DATA 40, "␣wati"
30 DATA 60, "␣wati"
40 READ a$, b, c$
50 RESTORE 10
60 READ a$
70 RESTORE 30
80 READ b, c$
```

Verificarea se obține introducînd liniile

```
45 PRINT a$, b, c$
90 PRINT a$, b, c$
```

4.1.3. Instrucțiunea INPUT

Această instrucțiune/comandă permite *introducerea datelor în timpul rulării programului* (numere, șiruri de caractere).

Sintaxa: [nr. linie] INPUT ["comentariu"] n1, n2, ... unde *nr. linie* și *"comentariu"* sînt opționale, iar *n1, n2, ...* sînt numele variabilelor de inițializat.

Exemple: **50 INPUT A**

80 INPUT B, D, K

90 INPUT T1, W, T2

100 INPUT a$ — afișează ghilimele între care se introduce șirul

200 INPUT "Introdu lungimea l[cm]", l

300 INPUT LINE a$ — nu afișează ghilimele între care să se introducă șirul

Efect: determină calculatorul să întrebe utilizatorul prin cursorul cliptor *L* ce valoare trebuie transferată variabilei din lista instrucțiunii **INPUT**.

Observații: 1) Trebuie să existe corespondență între tipul variabilei (simple sau indexate, numerice sau tip șir de caractere) și data introdusă.

2) Atribuirea valorii unei variabile se face în momentul tastării *CR*, după ce valoarea a fost introdusă de la tastatură; dacă sînt mai multe variabile de introdus, se va tasta *CR* după fiecare valoare introdusă în parte.

3) Elementele *n1, n2, ...*, ale instrucțiunii pot fi separate prin:

— *virgulă*, cînd capul de scriere este poziționat la *începutul* și apoi a *mijlocul liniei* (poziția *0*, respectiv poziția *16* — fig. 4.1));

— *punct și virgulă*, datele fiind introduse imediat *alăturat* după scrierea fiecărei date:

— *apostrof*, cînd poziționarea capului de scriere este *dedesubtul* fiecărei date.

Exemple: **10 INPUT "lungimea l="; L**

20 INPUT "Raza", r

30 INPUT "Introdu diametrul barei" 'd

4) Mai pot fi folosite formele:

INPUT # cale — a se vedea *capitolul 10* și instrucțiunea **PRINT**

INPUT AT linie, coloană
INPUT TAB coloană — vezi instrucțiunea **PRINT**

INPUT culori — vezi *capitolul 7.*

4.2. INSTRUCȚIUNEA PRINT PENTRU AFIȘAREA REZULTATELOR

Sintaxa: **[nr. linie] PRINT lista** (vidă, constante, variabile, expresii, mesaje), unde *nr. linie* este opțional.

Exemple:

```
2 PRINT
3 PRINT 22
5 PRINT (4 * X↑2+3 * X)/2
10 PRINT 2 * X, (3 * Y+4)/Z, X13, Z
20 PRINT a, b, c, d
30 PRINT a; b; c; d
40 PRINT a'b'c'd
50 PRINT "suprafața cercului s="; s; "_m.."
60 PRINT AT 11, 16; "*"
70 PRINT AT 11, 5; "LECTIA DE BASIC"
80 PRINT TAB 1; "MOM"; TAB 6; "MOM", TAB
   11; "MOM"
90 PRINT AT 6, 10; "CUCU"; AT 7, 8; "BAU"
```

Efect: determină transferul din memorie pe ecran a valorilor desemnate prin variabile sau constante.

● Forma

nr. linie PRINT

determină *avansarea în jos* a unei linii a ecranului și amplasarea începutului afișării la linia următoare. Scrierea succesivă a unor instrucțiuni de acest fel este echivalentă cu avansarea cu atîtea linii cîte instrucțiuni **PRINT** sînt scrise. De exemplu :

```
10 PRINT : PRINT : PRINT
```

determină *avansarea cu trei linii în josul ecranului.*

● Forma

nr. linie PRINT valoare numerică

permite afișarea numărului respectiv în format întreg, real sau exponențial (cînd sînt mai mici de 10^{-5} sau mai mari de 10^{8}), cu maximum *8* cifre semnificative. De exemplu

```
10 PRINT. 000005, 3000000000
```

determină afișarea

5E — 6 3E + 9.

● Forma

nr. linie PRINT expresie

determină calcularea și afișarea valorii expresiei fără a o stoca în memorie (v. liniile *5* și *10* de la exemple).

● Forma

nr. linie PRINT lista (constante, variabile, expresii)

permite afișarea valorilor :

— *două cîte două pe o linie*, prima începînd cu coloana *0* și cealaltă cu coloana *16*, dacă elementele listei sînt separate prin *virgulă* (v. liniile *10* și *20*);

— *în succesiune* pe aceeași linie începînd cu coloana *0* fără blancuri între ele, dacă elementele liniei sînt separate între ele prin *punct și virgulă;* atunci cînd valorile nu încap pe o linie se trece automat la linia următoare (v. linia *30*)

— *una sub alta* pe linii consecutive, atunci cînd elementele liniei sînt separate prin *apostrof* (v. linia *40*).

Exemplul 4.6: **10 INPUT a, b, c, d**

20 PRINT a, b, c, d

Tastînd **RUN** se afișează cursorul clipitor *L* și dacă se tastează succesiv *100* (urmat de *CR*), apoi *200* (urmat de *CR*) ,în continuare *300* (urmat de *CR*) și în fine *400* (urmat de *CR*), pe ecran va apare această suită de cifre ca în fig. 4.2.

Exemplul 4.7: **10 INPUT a, b, c, d**

20 PRINT a; b; c; d

Se apasă tasta **RUN** și apoi se introduc succesiv valorile *1* (urmate de *CR*), apoi *2* (urmat de *CR*), în continuare *3* (urmat de *CR*) și în fine *4* (urmat de *CR*), pe ecran se afișează aceste numere ca în fig. 4.3.

● Forma

nr. linie PRINT "mesaj"

determină afișarea mesajului care a fost scris între ghilimele.

Exemplul 4.8: **10 PRINT "LECTIA DE";**

20 PRINT "␣BASIC"

va afișa pe ecran

LECȚIA DE BASIC

● Forma

nr. linie PRINT AT linie, coloana

determină afișarea la linia și coloana dorită (conform împărțirii ecranului în *22* linii și *32* coloane, numerotate astfel : liniile de la *0* la *21* și coloanele de la *0* la *31* — vezi fig. 2.4).

Exemplul 4.9: **10 PRINT AT 11, 16; "#"**

determină afișarea caracterului # în mijlocul ecranului (linia *11*, ocloana *16*).

Exemplul 4.10: **10 PRINT AT 11,5; "LECTIA DE BASIC"**

conduce la afișarea textului dintre ghilimele începînd din linia *11*, coloana *5*.

● Forma

nr. linie PRINT TAB n	(*n* — nr. coloanei)

deplasează capul de scriere în coloana specificată (**TAB** tipărește spații pînă la coloana *n*).

Exemplul 4.11: **10 PRINT "ROMANIA"; TAB 10; "ROMANIA"; TAB 19; "ROMANIA"**

determină afișarea următoare (fig. 4.4) :

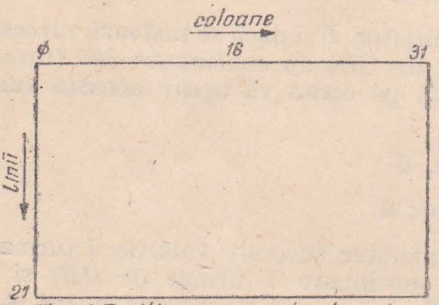


Fig. 4.1 Poziționarea capului de scriere la instrucțiunea de forma INPUT a1, a2

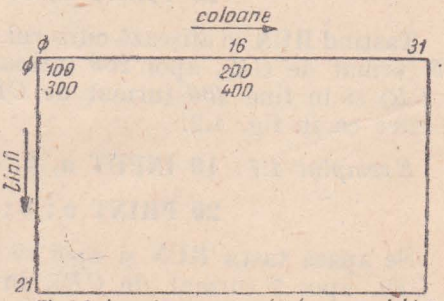


Fig. 4.2 Imaginea ecranului la exemplul 4.8

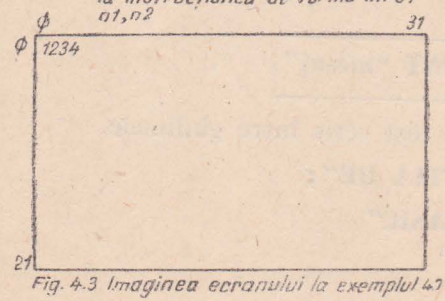


Fig. 4.3 Imaginea ecranului la exemplul 4.7

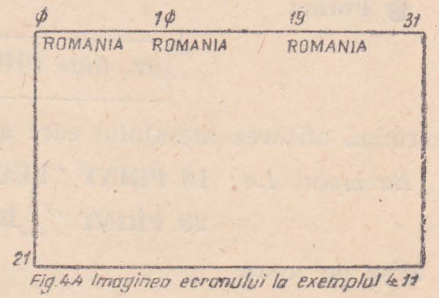


Fig. 4.4 Imaginea ecranului la exemplul 4.11

● Forma

nr. linie PRINT AT linie, coloana; "mesaj"; AT linie, coloana, variabila

permite scrierea de mesaje sau de rezultate pe linii succesive și în poziția dorită.

Exemplul 4.12:

```
9 REM Introducerea șirurilor de caractere
10 PRINT "Cum va numiți?"
20 INPUT n$
30 PRINT "Numele dvs. este "; n$
```

Tastînd **RUN** se afișează „*Cum vă numiți?*"; tastînd de exemplu cuvîntul *MIRCEA*, calculatorul va afișa

Numele dvs. este MIRCEA

Exemplul 4.13:

```
9 REM Transformarea gradelor Fahrenheit în grade
  Celsius, pe baza relației c=(f−32)5/9
10 PRINT "Grade F", "Grade C" : PRINT
20 INPUT "Introdu gradele F", f
30 PRINT f, (f−32)*5/9
```

Pentru rularea programului se apasă **RUN** și pe ecran apare scris

Grade F *Grade C*

Dacă se tastează apoi *100*, ecranul se va completa cu rezultatele calculului și va arăta în felul următor:

Grade F	*Grade C*
100	*37.777778*

- Forma

PRINT nume variabilă

determină afișarea valorii calculate a variabilei cu numele dorit, efectuată în timpul rulării programului.

- Forma

PRINT expresie numerică

determină calcularea și afișarea valorii expresiei respective, ca un calculator de buzunar. De exemplu **PRINT (16.379+115.895)/2.005** va afișa *65.97207*.

Observații:

1) Se pot afișa mesaje și în spațiul de editare (liniile *22* și *23*) folosind forma:

nr. linie PRINT #0; AT linie, coloana; "mesaj" : PAUSE 0

Exemplul 4.14: **10 PRINT AT 21, 0; "Ultima linie (linia 21)"**

20 PRINT # 0; AT 0, 0; "Linia 22
Linia 23" : PAUSE 0

2) Forma **PRINT TAB** consideră coloanele *„modulo 32"*, respectiv dacă numărul coloanei este mai mare decît *31*, se scade succesiv *32* pentru a se obține un număr cuprins între *0* și *31*. De exemplu:

PRINT TAB 32 este echivalent cu **PRINT TAB 0**

PRINT TAB 42 este echivalent cu **PRINT TAB 10**

PRINT TAB 75 este echivalent cu **PRINT TAB 11**, etc.

3) Tipărind cu **PRINT AT** pe o poziție deja scrisă, ultima tipărire o anulează pe precedenta.

4.3. PROBLEME

P4.1. Să se scrie un program care folosește instrucțiunile **READ** și **DATA** pentru a atribui valorile $a = 10$, $b = 5$, $c = -3$, $d = 8$, $e = 0$.

Rezolvare: **10 DATA 10, 5, −3, 8, 0**

20 READ a, b, c, d, e

P4.2. Se cere programul prin care se poate atribui variabilei șir *I$* valoarea *„Programare în BASIC"*.

Rezolvare: **10 READ I$**

20 DATA "Programare in BASIC"

P4.3. Idem pentru patru variabile șir care să conțină fiecare numele echipelor de fotbal: *Steaua*, *Dinamo*, *Universitatea Craiova*, *Politehnica Timișoara*.

Rezolvare: **10 READ a$, b$, c$, d$**

20 DATA "Steaua", "Dinamo", "Universitatea Cralova",
"Politehnica Timisoara"

P4.4. Se cere un program ce folosește instrucțiunile **RESTORE** pentru a se atribui succesiv valorile *10*, *100*, *1000* variabilelor *a*, *b*, *c* și respectiv *X*, *Y*, *Z*.

Rezolvare: **10 READ a, b, c**
20 DATA 10, 100, 1000
30 RESTURE 20
40 READ X, Y, Z

P4.5. Să se scrie programul care calculează și afișează volumul sferei. ($V = 4\pi r^3/3$)

Rezolvare: **10 CLS : INPUT "Introdu raza", r**
20 PRINT AT 10, 5; "Volumul sferei de raza"; AT 12,
10; "r="; r; AT 14, 8; "este :"; 4*r*r*r*PI/3

P4.6. Se cere programul pentru calculul sumei $S = X + Y + Z$ folosind numai instrucțiunile *INPUT* și *PRINT*.

Rezolvare:
```
10 PRINT AT 0, 4; "Calculul sumei S=X+Y+Z"
20 INPUT "Introdu X",X: PRINT "X="; X
30 INPUT "Introdu Y", Y: PRINT "Y="; Y
40 INPUT "Introdu Z", Z: PRINT "Z="; Z
50 PRINT AT 16, 7: "Suma S=X+Y+Z este: ";
AT 18, 11; "S="; X+Y+Z
```

P4.7. Să se calculeze expresia $y = ax^4 + bx + c$.

Rezolvare:
```
10 PRINT AT 3, 1; "Calculul expresiei y=ax ↑ 4+bx+c"
20 INPUT "Introdu a, b, c, x", a, b, c, x
30 PRINT AT 10, 11; "y="; a*x ↑ 4+b*x+c
```

P4.8. Să se calculeze mediile aritmetică, geometrică și armonică a trei numere nenule A, B, C (media aritmetică $(A+B+C)/3$; media geometrică $\sqrt[3]{ABC}$; media armonică $\frac{3}{\frac{1}{A}+\frac{1}{B}+\frac{1}{C}}$). Se vor folosi numai instrucțiunile *INPUT* și *PRINT*.

Rezolvare:
```
10 PRINT "Calculul:
        — mediei aritmetice
        — mediei geometrice
        — mediei armonice
          a trei numere A, B, C diferite de zero."
20 INPUT "Introdu A, B, C", a, b, c: PRINT AT 6, 14;
"A="; a; AT 7, 14; B"="; b; AT 8,14; "C="; c
30 PRINT AT 12, 0; "1) Media aritmetică: "; (a+b+c)/3
40 PRINT AT 14, 0 ;"2) Media geometrica :"; (a * b*c) ↑ (1/3)
50 PRINT AT 16, 0; "3) Media armonica :"; 3/((1/a) +
+(1/b)+(1/c))
```

P4.9. Să se tipărească pe ecran următoarele mesaje, scrise într-o singură linie de program:

„*ZX-SPECTRUM*" pe linia *0*, începînd din coloana *0*
„*INITIERE IN BASIC*" pe linia *11* poziționat la mijlocul liniei
„*MMP Software*" pe linia *21* în extremitatea dreaptă a ecranului.

Rezolvare:
```
10 CLS: PRINT "ZX-SPECTRUM"; AT 11, 7: "INITIERE
IN BASIC"; AT 21, 20; "MMP Software"
```

P4.10. Să se tipărească mesajul

„*APASATI O TASTA OARECARE*"

poziționat la mijlocul liniei *23* (prima linie din spațiul de editare).

Rezolvare:
```
10 PRINT # 0; AT 0, 4; "APASATI O TASTA
OARECARE": PAUSE 0
```

Capitolul 5

INSTRUCȚIUNI DE ATRIBUIRE, SALT ȘI PENTRU REZERVAREA MEMORIEI. INSTRUCȚIUNI LOGICE ȘI PENTRU TESTAREA CLAVIATURII.

5.1. INSTRUCȚIUNEA DE ATRIBUIRE LEI

Calculele aritmetice se realizează cu operatorii aritmetici

+ (*adnnare*) — (*scădere*) * (*înmulțire*)/(*împărțire*) ↑ (*ridicare la putere*)

iar rezultatul lor se atribuie unor variabile prin instrucțiunea/comanda **LET**.

Sintaxa: [nr. linie] **LET variabilă = expresie** unde [*nr. linie*] este opțional, *variabilă* —numele unei variabile simple sau indexate, *expresie* — expresie numerică sau alfanumerică

Exemple:

```
10 LET A = 1000
20 LET B9=C
30 LET z=a+21*2
40 LET K=(a+5)/(d+(27-i)*3)
50 LET N=N+1
60 LET a$="viteza"
```

Efect: determină evaluarea expresiei aritmetice/alfanumerice situate la dreapta semnului egal și atribuirea rezultatului variabilei situate în stînga egalului.

Observații: 1) Sînt interzișsi mai mulți *operatori consecutivi*. De exemplu $c(-d)$ se scrie $c*(-d)$ și nu $c*-d$.

2) Limbajul *BASIC* folosește numai *parantezele rotunde*. Parantezele drepte sau acoladele utilizate în algebră se înlocuiesc cu mai multe perechi de paranteze rotunde închise una în alta, *numărul de paranteze stînga trebuind să fie egal cu numărul de paranteze dreapta*. De exemplu expresia $[(x+a)(y-b)]^2$ se scrie ((x + a) * (y − b)) ↑ 2.

3) O fracție avînd la numitor mai mulți termeni reuniți prin operatori aritmetici se va scrie *delimitînd numitorul între paranteze.* De exemplu: *a/2b* se scrie **a/(2 * b)** și nu a/2 * b care înseamnă *ab/2.*

4) Ridicarea la o putere fracționară trebuie scrisă cu *puterea între paranteze.* De exemplu $\sqrt[3]{(ab)^2}$ se scrie **((a*b)*2) ↑ (1/3)** sau **(a * b) ↑ (2/3).**

5) Cu ajutorul instrucțiunii/comenzii **LET** se poate realiza și *concatenarea (alipirea) unor șiruri de caractere.*

De exemplu se dorește concatenarea șirurilor de caractere *„Mitică"* și *„Popescu"*. În acest scop se scriu instrucțiunile:

```
10 LET a$="Mitica_": LET bS="Popescu": LET c$=a$+b$:
   PRINT c$
```

6) **LET** permite și *manipularea subșirurilor* care se construiesc sub forma

s(n1 TO ns)

unde *„s"* este un șir de caractere sau o variabilă șir, *„n1"* și *„ns"* sînt numere nenegative ce reprezintă numărul de ordine al caracterului de început, respectiv de sfîrșit din subșir (dacă *n1* > *ns* rezultatul este *șirul vid* " "; dacă *n1* nu este precizat se ia implicit *1*, iar dacă nu este specificat *ns* atunci se ia implicit egal cu *lungimea șirului).*

De exemplu se dorește extragerea subșirului *„Pop"* din șirul *„Popovici"*. Scriind programul următor se realizează cerința exprimată:

```
10 LET a$="Popovici"
10 LET b$=a$(TO 3)
30 PRINT b$
```

În mod analog se poate transforma un *șir în alt șir.* De exemplu transformarea șirului *„abcdef"* în șirul *„abXYef"* se face cu ajutorul programului:

```
10 LET a$="abcdef"
20 LET a$ (3 TO 4)="XY"
30 PRINT a$
```

Exemplul 5.1:
```
10 LET A=2
20 PRINT A
30 LET B=A * A
40 PRINT B
50 LET C=B*B
60 PRINT C
```

Tastînd **RUN** și **CR** programul va afișa pe ecran, unul sub altul, cifrele *2, 4* și *16.*

Exemplul 5.2:
```
10 INPUT "Introdu valorile a, b, c", a, b, c
20 LET s = a+b+c
30 PRINT AT 12, 5; "Suma valorilor este:"; AT
   14, 8; "s="; s
```

Exemplul 5.3:
```
10 LET A=11 : LET B=201
20 LET C=A*B : LET D=A/B : LET E=A ↑ B :
LET F=(A*B) ↑ 0.5
30 PRINT C, D, E, F
```

Tastînd **RUN** și **CR** pe ecran se afișează

22.11 5.4726368

123.93652 4.7021272

Exemplul 5.4:
```
10 REM Scriind "10 la puterea" folosind litera e
20 READ A, B, C : RESTORE 30
30 DATA 3.2e1, 8.5, 45e2
40 LET d=10*A+B*C
50 PRINT D : PAUSE 0 : CLS
60 REM Scriind "10 la puterea" folosind litera E
70 RESTORE 80 : READ A, B, C
80 DATA 3.2E1, 8.5, 45E2
90 LET d=10*A+B*C : PRINT AT 20, 0; 0
```

Programul rulat va afișa același rezultat *38570*; se poate constata că nu se face distincție între literele mici și cele mari.

Exemplul 5.5: Într-un garaj se schimbă n piese avînd fiecare prețul p în h ore de lucru, cu un cost orar m și regie r. Se cere costul total calculat cu relația $c = npmh(1 + r)$.

```
10 REM Factura
20 RESTORE 40 : READ n, p, h, m, r
30 PRINT "Costul :"; n*p*m*h*(1+r); "_lei"
40 DATA 10, 100, 2, 22, .25
50 PAUSE 0 : CLS
60 INPUT "Introduceți valorile n, p, h, m, r", n,p, n, m, r
70 LET c=n*p*m*h*(1+r)
80 PRINT "Costul : "; c; "_lei"
```

Exemplul 5.6:
```
10 CLS : PRINT "Da-mi un nr. si-l ridic la cub"
20 INPUT "Nr. tau ?", n
30 LET c=n ↑ 3
40 PRINT AT 8, 0; n; "_ridicat la cub este";
AT 20, 8; c
```

Exemplul 5.7: Să se elaboreze un program care afișează zilele și orele care rezultă din vîrsta unei persoane.

```
10 CLS : PRINT "Citi ani ai ?"
20 INPUT A : PRINT AT 0, 16; A; "_ani"
30 LET Z= =A*365 : LET H=Z*24
40 PRINT :PRINT : "Aceasta face "; Z; "_zile_sau"
50 PRINT :PRINT : TAB 10; H "_ore" : PRINT :
   PRINT : PRINT AT 15, 8; "MULTI INAINTE !"
```

5.2. INSTRUCȚIUNI/COMENZI DE SALT

Execuția unui program *BASIC* se face linie cu linie, în ordinea crescătoare a numerelor lor de ordine (a etichetelor).

Există însă situații cînd se impune *schimbarea ordinii de execuție a liniilor programului*; o asemenea schimbare este realizată de instrucțiunile de salt. Din acest grup fac parte:

— instrucțiunea/comanda de *salt necondiționat* **GO TO**;
— instrucțiunea/comanda de *salt condiționat* **IF—THEN GO TO**;
— instrucțiunea/comanda de *ciclare* **FOR—TO—STEP—NEXT**.

5.2.1. Instrucțiunea/Comanda GO TO

Sintaxa: [nr. linie] **GO TO** nr. linie, unde *[nr. linie]* este opțional

Exemple: **100 GO TO 5000**

GO TO 600

Efecte: transferă execuția programului la numărul de linie specificat în instrucțiunea **GO TO**. Dacă în program nu există linia specificată, saltul se face la prima linie care există după numărul de linie indicat în instrucțiune. Cînd numărul de linie specificat conține o instrucțiune neexecutabilă (**REM, DATA**), saltul se face la instrucțiunea imediat următoare.
Transferul la o instrucțiune multiplă se poate da numai *primei instrucțiuni* din șirul celei multiple.
În calitate de comandă, **GO TO** se folosește pentru executarea programului de la linia avînd numărul introdus, fără a fi necesară ștergerea ecranului.

Deosebirile față de **RUN**: nu șterge variabilele din memorie și nu se șterge ecranul.

Exemplul 5.8: **10 PRINT "ROMANIA"**

20 GO TO 10

Datorită liniei *20*, linia *10* se va executa repetat pînă la umplerea ecranului; în acest moment în spațiul de editare apare mesajul

scroll?

care semnifică „*doriți afișarea unei pagini?*". Pentru un răspuns afirmativ se apasă pe orice tastă (cu excepția tastei *N* de la **NOT**) și afișarea se reia pînă la o nouă umplere a ecranului, cînd mesajul *scroll?* va reapare. Dacă însă se tastează *N*, răspunsul este negativ, pe ecran apare mesajul

D BREAK-CONT repeats, 10: 1

și execuția programului încetează.

Exemplul 5.9:

```
10 REM se reia exercițiul 5.5 sub alta forma
20 INPUT "Introdu valorile pentru n, p, h, m, r",
n, p, h, m, r
30 LET c=n*p*h*m*(1+r)
40 PRINT "Costul este c="; c; "␣lei"
50 CLS: PAUSE 200: GO TO 10
```

5.2.2. Instrucțiunea/Comanda IF-THEN

Sintaxa: [nr. linie] **IF** condiție **THEN** instrucțiuni, unde [*nr. linie*] este opțional, „*condiție*" reprezintă o expresie aritmetică, logică sau de relație cu valoare de adevăr sau fals, iar „*instrucțiuni*" sînt instrucțiunile ce se vor executa dacă este adevărată condiția; în caz contrar (condiție falsă) se execută instrucțiunile de după **IF** (din linia următoare)

Exemple:

```
100 IF b=0 THEN GO TO 300
200 IF A+B>0 THEN LET C=A
300 IF R$="DA" THEN CLS: STOP
IF delta < 0 THEN PRINT "Ecuatia are radacini
complexe"
```

Efect: Dacă condiția scrisă între cuvintele cheie **IF** și **THEN** este adevărată se execută instrucțiunile de după **THEN**, iar dacă această condiție nu este adevărată se execută instrucțiunile din linia următoare.

Observație) Instrucțiunea **IF–THEN** poate fi utilizată și sub formă prescurtată; de exemplu:

```
10 LET a=0
20 IF a THEN STOP
30 PRINT "BASIC"
```

Dacă $a = 1$ condiția este falsă, iar dacă $a = 0$ condiția este adevărată.

Exemplul 5.10:

```
9 REM Impartirea a 2 numere (varianta 1)
10 INPUT "Introdu valoarea împărțitorului A", A
15 INPUT "Introdu valoarea deîmpărțitului B", B
20 IF A=0 THEN GO TO 60
30 LET C= B/A
40 CLS : PRINT AT 12, 0; "B="; B; "␣împărțit
la A="; A; AT 14, 0; "face␣ :"; C
50 PAUSE 100: GO TO 10
60 CLS : PRINT AT 12, 0; "IMPARTIREA NU SE
POATE EFECTUA" : STOP
```

Exemplul 5.11:

```
9 REM Impartirea a 2 numere (varianta 2)
10 INPUT "Introdu numerele A și B", A, B
20 IF A=0 THEN GO TO 50
30 IF B=0 THEN GO TO 80
40 LET C=B/A : PRINT AT 12, 0; "B="; B;"
împărțit la A=" ; A ; AT 13, 2;"face␣" ; C: GO TO 90
50 IF B=0 THEN GO TO 70
60 PRINT "IMPARTIRE IMPOSIBILA" :GO TO 90
70 PRINT "NEDETERMINARE" (0/0)" : GO TO 90
80 PRINT "C=0"
90 STOP
```

Exemplul 5.12:

```
19 REM Jocul "Ghicește numărul !"
20 INPUT "Introdu numărul a", a : CLS
30 INPUT "Ghicește numărul", b
40 IF b=a THEN CLS : PRINT AT 11, 3; "Rezultat
corect ! BRAVO␣ !" : STOP
50 IF b<a THEN PRINT "Prea mic. Mai încearcă !" :
GOTO 30
60 IF b > a THEN PRINT "Prea mare. Mai
încearcă !" : GO TO 30
1000 STOP
```

5.2.3. Instrucțiuni de ciclare (FOR-TO-NEXT-STEP)

Un grup de instrucțiuni care se execută de mai multe ori reprezintă **un** *ciclu*. Fiecare parcurgere a ciclului se numește *pas*, iar evidența pașilor poate fi asigurată de un *contor* care reprezintă *variabila de ciclare*. Modificarea valorii variabilei de ciclare se face prin *incrementare* (creștere) sau

decrementare (descreștere). Instrucțiunea care asigură parcurgerea ciclurilor este **FOR –TO –NEXT(–STEP]**.

Sintaxa: [nr. linie] FOR v=vi TO vf [STEP r]
: corpul ciclului
.
nr. linie NEXT v

unde: *v* —nume de variabilă numerică formată dintr-o *singură literă* numită *contorul ciclului*;
vi —valoare numerică, variabilă simplă sau expresie aritmetică, care după evaluare exprimă *valoarea de început a contorului v*;
vf —idem, care după evaluare exprimă *valoarea finală a contorului v*;
r — idem, care după evaluare exprimă *pasul (rația) de creștere sau descreștere a contorului* (incrementul sau decrementul); cînd lipsește STEP r se consideră implicit că *r=1*.

Exemple:

```
10 FOR k=1 TO 10
.
.corpul ciclului
.
20 NEXT k
```

```
10 FOR L=3 TO 40 STEP 2
.
.
20 NEXT L
```

```
10 FOR z=A+B TO 100
.
.
20 NEXT z
```

```
10 FOR a=0 TO 2*PI STEP PI/200
.
.
20 NEXT a
```

```
10 FOR c=3 TO 1 STEP -1
.
.
20 NEXT c
```

```
10 FOR i= .03 TO .15 STEP .035
.
.
20 NEXT i
```

Efect: se definește o secvență de instrucțiuni la care corpul ciclului va fi executat de un număr de ori prestabilit.

Observații: 1) Sînt *permise)* a) ieșirea din ciclu cu o instrucțiune **GO TO**; b) cicluri suprapuse (cicluri în cicluri sau cicluri imbricate).

2) Sînt *interzise:* a) intrarea în ciclu; b) modificarea contorului ciclului în cadrul instrucțiunilor din corpul ciclului.

3) Ciclul poate să nu fie parcurs în două cazuri: $vi > vf$ și $r > 0$ sau $vi < vf$ și $r < 0$; în ambele situații se sare la prima instrucțiune după **NEXT**.

4) Se pot obține erori în următoarele situații:
— mesajul de eroare *1:* cînd se întîlnește **NEXT** și programul nu are instrucțiunea **FOR**;
— mesajul de eroare *2:* cînd se întîlnește **NEXT** și nu s-a trecut prin **FOR**-ul respectiv și nici nu există o variabilă definită anterior cu același nume cu variabila de control;
— mesajul de eroare *1:* cînd ciclul începe cu **FOR** dar nu se termină cu **NEXT**.

4) Pentru a părăsi un ciclu (o buclă) înainte de sfîrșitul ei normal se poate proceda în două moduri așa cum se indică în cele două programe echivalente ca efect, dintre care primul are o soluționare de preferat.

```
10 FOR i=1 TO 10                    10 FOR i=1 TO 10
20 INPUT "Continuăm?", a$           20 INPUT "Continuăm?", a$
30 IF a$="da" THEN LET i=11         30 IF a$="da" THEN GO TO 10
40 NEXT i                           40 NEXT i
```

Primul program folosește ieșirea forțată din buclă; astfel, la instrucțiunea *40* dacă $i = 11$, respectiv s-a răspuns „*da*" la întrebarea din **INPUT**, bucla este abandonată. Al doilea program realizează ieșirea directă din buclă, dar această ieșire nu este recomandabilă, deoarece dacă se întîlnește un alt **NEXT i** nepregătit de un alt **FOR**, programul va relua execuția de la instrucțiunea de după **FOR** (linia *20)*.

Exemplul 5.13:
```
10 FOR k=10 TO 50 STEP 10
30 PRINT k
30 NEXT k
```

Se afișează:
```
10
20
30
40
50
```

Exemplul 5.14:
```
10 FOR c=3 TO 1 STEP -1
20 PRINT C
30 NEXT C
```

Se afișează:
```
3
2
1
```

Exemplul 5.15:

```
10 FOR i = .03 TO .15 STEP .035
20 PRINT i
30 NEXT i
40 PRINT "Valoarea variabilei i după parcurgerea
   ciclului :"; i
```

Se afișează:

```
 .03
 .065
 .01
 0.135
Valoarea variabilei i după parcurgerea
ciclului : 0.17
```

Exemplul 5.16:

```
10 INPUT "Introdu numărul A", A
20 FOR L = 10 TO A : PRINT L : NEXT L
30 PRINT "Sfîrșit"
```

Tastînd valoarea 15 (la cererea instrucțiunii **INPUT**) se afișează:

```
10
11
12
13
14
15
Sfîrșit
```

Exemplul 5.17:

```
9 REM cicluri imbricate
10 PRINT "N"; TAB 10; "B"; TAB 20; "C"
20 LET N = 0
30 FOR B = 1 TO 3
40 FOR C = 1 TO 2
50 LET N = N + 1
60 PRINT N; TAB 10; B; TAB 20; C
70 NEXT C
80 NEXT B
```

Se afișează:

N	B	C
1	1	1
2	1	2
3	2	1
4	2	2
5	3	1
6	3	2

Se observă că pentru buclele imbricate variabilele de control trebuie să aibă *nume diferite.*

Exercițiul 5.18: se dă o listă de *10* numere și se cere să se determine dacă este conținut sau nu numărul *0.*

```
10 FOR i = 1 TO 10
20 READ x
30 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
40 IF x = 0 THEN GO TO 80
50 NEXT i
60 PRINT "Lista nu conține numărul 0"
70 STOP
80 PRINT "Lista conține numărul 0" :STOP
```

Exercițiul 5.19: să se afișeze primele 10 numere naturale impare

```
10 FOR i=1 TO 19 STEP 2
20 PRINT i
30 NEXT i
```

Exercițiul 5.20: să se afișeze în ordine descrescătoare primele 10 numere naturale

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

Exercițiul 5.21: să se afișeze tabla înmulțirii cu 10

```
10 FOR y=0 TO 9
20 PRINT y; "*10="; y*10
30 NEXT y
```

Exercițiul 5.22) să se calculeze energia electronilor folosind formula $en = 2\pi^2 me^4 Z^2/n^2h^2$ unde $h = 6{,}62619 \cdot 10^{-27}$ este constanta lui *Plank*, $m = 9{,}1 \cdot 10^{-28}$ *g* este masa electronului, $e = 4{,}8 \cdot 10^{-10}$ *u.e.s.* este sarcina electronului, n — numărul cuantic principal (numărul stratului) și Z — numărul de ordine al elementului în tabelul lui Mendeleev ($n = 1 \ldots 6$; $Z = 1 \ldots 104$).

```
10 LET h=6.62619 : LET m=9.1 :LET e=4.8
19 PRINT AT 0, 0; "Nr. ord."; AT 0,7; "Nr. cuantic"; AT 0, 19;
   "Energia"
20 FOR z=1 TO 104 :FOR n=1 TO 6
30 LET en=2*PI*PI*m*e↑4*z*z/(n*n*h*h)
40 PRINT z; TAB 8; n; TAB 18; en
50 NEXT n : NEXT z
```

5.3. INSTRUCȚIUNEA DIM PENTRU REZERVAREA MEMORIEI

În *BASIC* variabilele *A* și *A(4)* sînt diferite; prima este o *variabilă simplă* iar a doua o *variabilă indexată*. Variabilele indexate uzuale pot avea :

— *un singur indice* cînd se numesc **vectori** (sau *vectori coloană*, *variabile listă* sau *variabilă tip tablou unidimensional*) ; exemplu : *B(6)* ;

— *doi indici* cînd se numesc **matrice** (sau *variabile tip tablou bidimensional*), primul indice referindu-se la *linia* matricei iar al doilea la *coloana* ei ; exemplu *k(3, 4)*.

		1	*2*	*3*	*4* ⟶ *coloana*
linia ↓	*1*	k(1, 1)	k(1, 2)	k(1, 3)	k(1, 4)
	2	k(2, 1)	k(2, 2)	k(2, 3)	k(2, 4)
	3	k(3, 1)	k(3, 2)	k(3, 3)	k(3, 4)

In limbajul *BASIC* al calculatoarelor compatibile cu *ZX SPECTRUM* se pot defini variabile tip tablou cu *oricîte dimensiuni*. Dimensiunea fizică a unui tablou (masiv) se declară cu instrucțiunea **DIM**.

Sintaxa: nr. linie **DIM nume (indice1, indice2, . . .)**
unde *nume* — numele variabilei format dintr-o *singură literă; indice 1, indice 2, . . .* — numerele maxime de componente pentru fiecare dimensiune (indicile minim este 1)

Exemple: **10 DIM B(15)**
20 DIM A(10, 10)
30 DIM A$(75)

Efect: declară variabila cu numele precizat ca variabilă indexată, rezervă spațiu în memorie pentru toate componentele ei și inițializează fiecare componentă cu valoarea zero, respectiv cu șirul nul (" ").

Observații: 1) Instrucțiunea **DIM** trebuie amplasată în program *înainte de prima utilizare a variabilei indexate*, iar printr-o asemenea instrucțiune se poate defini *o singură variabilă tip tablou*.

2) Într-un program pot coexista fără să apară confuzii, un tablou și o variabilă simplă cu același nume, dar nu pot coexista un tablou de șiruri și o variabilă simplă cu același nume.

3) Definirea unui tablou *a$* avînd *n* șiruri necesita stabilirea lungimii șirului (respectiv numărul de caractere) :

DIM a$(n, k)

De exemplu: 6 **DIM aS(5, 10)** definește un tablou *a$* cu *5* șiruri avînd fiecare lungimea de *10* caractere. *Fiecare rînd poate fi interpretat ca*

un șir; de exemplu *a$(1)* este format din *a$(1, 1), a$(1, 2) ... a$(1, 10).* Utilizînd un singur indice se obține un *șir de lungime fixă* [de exemplu *a$(2)*], iar dacă sînt folosite două dimensiuni se obține *un singur caracter* [de exemplu : *a$(3, 6)* este al șaselea caracter din șirul *a$(3);* același element se poate nota *a$(3)(6)*].

4) Ultimul indice dintr-un tablou de șiruri poate juca rol de *selector de subșir* dacă se scrie sub forma

a$(m) (n1 TO n2)

unde *m* — ordinul șirului iar *n1* și *n2* sînt numere întregi nenegative ce reprezintă ordinul caracterului de început și respectiv de sfîrșit de șir (dacă nu se precizează *n1* și/sau *n2*, se iau implicit valorile *1* respectiv *lungimea șirului)*, iar **TO** este o instrucțiune care întoarce caracterele șirului situate între pozițiile *n1* și *n2*.

De exemplu dacă *a$(2)* = "*1234567890*", atunci *a$(2)(4 TO 8)* = = "*45678*"

5) Dacă vreunul din indicii instrucțiunii

DIM nume (indice1, indice2, ...)

nu este întreg sau pozitiv sau depășește *65535*, ori numărul total de dimensiuni depășește *255*, apare mesajul de eroare

B Integer out of range

6) Se obține mesajul de eroare

3 Subscrit wrong

în cazurile cînd :

— se apelează o variabilă indexată nedimensionată anterior ;

— numărul de indici de apelare este mai mic decît cel de la dimensionare ; de exemplu : dacă **DIM a(10, 5), atunci PRINT** a(2) sau **PRINT a(1, 1, 15)** vor conduce la afișarea mesajului de eroare precizat anterior ;

— la apelare valoarea unui indice este negativă sau depășește valoarea aceluiași indice de la dimensionare ; de exemplu : dacă **DIM a(2, 3)** atunci **PRINT a(2, —3)** sau **LET** w = a(3, 3) va conduce la afișarea mesajului de eroare.

7) Se poate opera cu un caracter al unui șir ca și cum ar fi un șir de sine stătător. De exemplu programul

```
10 LET a$="sîmbătă"
20 PRINT a$
30 LET a$(1)="resurse"
40 PRINT a$
```

determină afișarea următoare :

sîmbătă
rîmbătă

Se observă din linia *30* că s-a reținut doar primul caracter din șirul *asignat* (atribut) unui singur caracter.

În mod analog, programul

```
10 LET a$="roșu" :PRINT a$
20 LET a$=a$(1) : PRINT a$
30 LET a$=a$+a$(1) : PRINT a$
```

determină următoarea afișare :

roșu
r
rr

8) Următorul exemplu *rotește un șir de caractere* și anume cele care ies prin dreapta ecranului apar în stînga acestuia :

```
10 LET a$="Programat de M. M. POPOVICI 1992␣ ␣"
20 PRINT # 1; AT 1, 0; a$
30 LET a$=a$(32)+a$(TO 31)
```

Ieșirea din acest program care se repetă se face introducînd în linia *40* o instrucțiune care încă nu a fost prezentată și anume **INKEY$**. Astfel :

```
40 PAUSE 5: IF INKEY$=" "THEN GO TO 20
50 CLS :STOP
```

Exemplul 5.23 :

```
9 REM Inițializarea unui vector
10 DIM v(3) :REM vectorul v are 3 componente
20 INPUT "Introdu primul termen v(1)", v(1)
30 LET v(2)=v(1) ↑ 2
40 LET v(3)=v(1)+v(2)
50 PRINT "Termenii vectorului v(3) :": PRINT
   TAB 3; v(1); TAB 13; v(2); TAB 23; v(3)
```

Tastînd *2* (ca răspuns la cererea dată de **INPUT**) se afișează

2 4 6

Exemplul 5.24 :

```
9 REM Inițializarea unei matrice cu 2 linii și
  3 coloane denumită T(2, 3)
10 DIM T(2, 3)
20 INPUT "Introdu primul termen T(1, 1)", T(1, 1)
30 INPUT "Introdu al doilea termen T(2, 1)", T(2, 1)
40 LET T(1, 2)=T(1, 1) ↑ 2 : LET T(2, 2) =
   T(2, 1) ↑ 2
50 LET T(1, 3)=T(1. 1)+T(1, 2) :LET T(2, 3)=
   T(2, 1)+T(2, 2)
60 PRINT AT 1, 4; "MATRICEA T(2, 3) ESTE :"
   :PRINT :PRINT :PRINT
70 PRINT T(1, 1); TAB 10; T(1, 2); TAB 20;
   T(1, 3)
80 PRINT T(2, 1); TAB 10; T(2, 2); TAB 20;
   T(2, 3)
```

Tastînd succesiv *2* și apoi *2*, se afișează :

MATRICEA T(2, 3) ESTE:

2 4 6
2 4 6

Exercițiul 5.25: să se calculeze suma cantităților vîndute din două produse *X* și *Y* în timp de *12* luni.

```
10 DIM X(12) : DIM X(12) : DIM S(12)

20 FOR I=1 TO 12 : INPUT X(I) : INPUT Y(I) :NEXT I

25 PRINT AT 0, 3 : "CANTITATILE VINDUTE LUNAR :"

30 FOR I=1 TI 12 ; LET S(I)=X(I)+Y(I) :PRINT AT i+1, 18 ;
   S(I) :NEXT I

35 PRINT AT 2, 1 : "Luna ianuarie_ _ :" ; AT 3, 1 ; "Luna febru-
   rie_ :" ; AT 4, 1 ; "Luna martie_ _ _ _ :" ; AT 5, 1 ; "Luna
   aprilie_ _ _ :" ; AT 6, 1 ; "Luna mai_ _ _ _ _ _ _ :" ;
   AT 7, 1 ; "Luna iunie_ _ _ _ _ :" ; AT 8, 1 ; "Luna iulie
   _ _ _ _ _ :" ; AT 9, 1 ; "Luna august_ _ _ _ :" ; AT 10,
   1 ; "Luna septembrie :" ; AT 11, 1 ; "Luna octombrie_ :" ; AT
   12, 1 : "Luna noiembrie_ :" ; AT 13, 1 ; "Luna decembrie_ :"

40 LET total = 0 : For i=1 TO 12 : LET Total=total+ S(I) :
   NEXT i

45 PRINT AT 15, 1 ; "TOTAL_ _ _ _ _ _ _ _ _ _ :" ; AT
   15, 18 ; total
```

Exercițiul 5.26: să se afișeze numele, mediile și media generală a unui elev (se presupun max. *18* discipline școlare)

```
10 CLS :DIM n$(26) : REM variabila n$ conține numele elevului
   (cu max. 26 caractere)

20 INPUT "Numele elevului ?", n$

25 CLS :PRINT "Nume :_" ; n$

30 INPUT "Introdu nr. disciplinelor <=18", d :DIM m(d)

40 FOR j=1 TO d :INPUT "Introdu media_" ; m(j) :NEXT j

45 PRINT AT 2, 1 ; "Media_" : FOR j=1 TO d :PRINT AT j+2,
   1 ; - _obiectului_" ; j ; AT j+2, 20 ; " :" ; AT j+2, 22 ;
   m(j) :NEXT j

50 LET media=0 :FOR j=1 TO d :LET media=(media+m(j)) :
   NEXT j

55 LET medgen=media/d :PRINT # 1 ; AT 1, 1 ; "MEDIA GENE-
   RALA : " ; medgen : PAUSE 0
```

Exercițiul 5.27: să se elaboreze un program care caută o valoare în coloana *1* a tabelului următor și afișează valoarea corespunzătoare din coloana *2:*

	Linii	
	10	7
C	30	6
o	40	12
l	60	8
o	70	9
a	90	4
n	100	5
e	120	9
	130	10
	150	30

```
6 CLS :PRINT AT 6, 2; "TABELUL T"; AT 8, 3; "Linii"; AT
  11,0; "C"; AT 12, 0; "o"; AT 13, 0; "l"; AT 14, 0; "o";
  AT 15, 0; "a"; AT 16, 0; "n"; AT 17, 0; "e"
7 PRINT AT 10, 2; "10_ _ _ _7"; AT 11, 2; "30_ _ _ _6";
  AT 12, 2; "40_ _ _ 12"; AT 13, 2; "60_ _ _ _8"; AT
  14, 2; "70_ _ _ _9"; AT 15, 2; "100_ _ _4"; AT 16, 2;
  "120_ _ _ _9"; AT 17, 2; "130_ _10"; AT 18, 2;
  "150_ _30"
9 PRINT AT 16, 16; "APASATI O TASTA"; PAUSE 0: RES-
  TORE 15
10 DIM T(10, 2) :FOR I=1 TO 10 : FOR J=1 TO 2 : READ T(I,J)
   NEXT J :NEXT I
15 DATA 10, 7, 30, 6, 40, 12, 60, 8, 70, 9, 90, 4,100, 5, 120, 9, 130,
   10, 150, 30 :REM valorile din tabel citite pe orizontală
20 INPUT "INTRODU NUMARUL DORIT DIN COLOANA 1", R
55 FOR L=1 TO 10
56 IF (TL, 1)=R THEN GO TO 100
60 NEXT L
70 PRINT AT 16, 16; "VALOAREA"; AT 17, 16; R; "_NU
   EXISTA" :GO TO 120
110 PRINT AT 16, 16; "VALOAREA CAUTATA"; AT 17, 16;
    "ESTE_ :"; T(L, 2) ;"_ _ _ _ _ _ _ _"
120 STOP
```

Exercițiul 5.28: să se elaboreze un program care să afișeze valoarea corespunzătoare unui interval dat conform tabelului următor:

Intervalul	*Valoarea*
<5	3
5...15	5
15...50	7
50...170	8
170...210	9
210...240	10
240...260	13
260...270	15
>270	18

```
10 PRINT AT 4, 1; "Intervalul          Valoarea"; AT 6,4;
   "<5                    3"; AT 7, 2; "5...15
                          5"; AT 8, 1; "15...50
                        7"; AT 9, 1; "50...170
                      8"; AT 10, 1; "170...210
                    9"; AT 11, 1; "210...240      10"; AT
   12, 1; "240...260              13"; AT 13, 1;
   "260...270          15"; AT 14, 3; ">270
                  18"
15 DIM s(9, 2) : RESTORE 25
20 FOR i=1 TO 9 :FOR j=1 TO 2 :READ s(i, j) :NEXT j :NEXT i
25 DATA 5, 3, 15, 5, 30, 7, 170, 8, 210, 9, 240, 10,260, 13, 270, 15
   271, 18 :REM s-au introdus valorile extreme ale intervalelor
40 INPUT "Introduceți numărul", R
50 FOR L=1 TO 8
60 IF R<s(L, 1) THEN GO TO 80
70 NEXT L
80 LET t=s(L, 2)
90 PRINT AT 20, 3; "Valoarea căutată :"; t
```

Exercițiul 5.29: în calculele de proiectare este necesară alegerea unor *valori standardizate;* să se elaboreze un program care afișează lungimea standardizată a penei în urma dimensionării ei prin calcule de rezistență, folosind relația $l_{STAS} > l_{calculat}$ (pentru a nu lungi programul se vor lua doar patru valori standardizate: *70, 80, 90, 100 mm*).

```
 5 CLS :LET n=4 :REM n—numărul valorilor standardizate
10 DIM l(n) :RESTORE 30
20 FOR i=1 TO n :READ l(i) : NEXT i
30 DATA 70, 80, 90, 100 : REM șirul valorilor standardizare scrise
   în ordine crescătoare
40 INPUT "Introdu lungimea calculată a penei (mm)", l
50 FOR i=1 TO n
55 IF l>l(n) THEN PRINT "NU EXISTA VALOARE STANDARDI-
   ZATA" :STOP
56 IF l<l(1) THEN LET l=l(1) :GO TO 200
60 IF l=l(i) THEN GO TO 200
70 IF l<l(i+1) THEN LET l=l(i+1) :GO TO 200
80 NEXT i
200 PRINT "Lungimea standardizată l="; l; " mm"
```

5.4. INSTRUCȚIUNI LOGICE ȘI PENTRU TESTAREA CLAVIATURII *(NOT, AND, OR, INKEY$)*

● *Operațiile logice* se realizează folosind instrucțiunile **NOT, AND** și **OR.**

Sintaxa:	**NOT condiție**
	condiție 1 AND condiție 2
	condiție 1 OR condiție 2

● **NOT** *(nu)* este un *operator unar* deoarece formează o nouă condiție dintr-o singură condiție veche. Condiția nouă este opusă condiției inițiale. Astfel instrucțiunea

10 IF NOT k=0 THEN GO TO 50

va determina calculatorul să treacă la linia 50 numai dacă $k = 0$ (ceea ce se mai poate scrie **IF k<⊲⊳0 THEN GO TO 50**).

● **AND** *(și)* este un *operator binar* care conduce la un rezultat adevărat numai dacă ambele condiții *1* și *2* sînt adevărate. De pildă

10 IF a=b AND b=c NOT c=0 THEN CLS

determină ștergerea ecranului numai dacă $a = b$ *și* $c = 0$.

● **OR** *(sau)* este un *operator binar* care conduce la un rezultat adevărat numai dacă una din cele două condiții *1* și *2* este adevărată. De exemplu:

10 IF a=1 OR a=3 THEN NEW

produce distrugerea programului numai dacă $a = 1$ sau $a = 3$.

Este ușor de înțeles că se pot crea condiții oricît de complicate; astfel instrucțiunea multiplă următoare

10 IF(a=0 OR a=1) AND (b=0 OR b=1) AND (NOT a=b) THEN STOP

va determina oprirea programului doar în două cazuri: cînd $a = 1$ și $b = 0$ sau $a = 0$ și $b = 1$.

Observații: 1) Instrucțiunea **AND** dispune și de varianta cînd unul din parametri este un șir. Astfel, modul său de lucru cînd primul parametru este un șir este

a$ AND b= $\begin{cases} \text{" "} & \text{dacă } b=0 \\ \text{a\$} & \text{dacă } b<>0 \end{cases}$, respectiv rezultatul este un șir

O aplicație imediată este realizarea acordului de număr. De pildă instrucțiunile următoare vor afișa o sumă de bani ce poate varia între 1 leu și o sumă oarecare:

10 PRINT "Datorezi _"; a; "_le"; ("u" AND a=1); ("i" AND a>1)

Dacă $a = 1$ se va afișa

Datorezi 1 leu

iar dacă $a > 1$ (de ex. *100*), se va afișa

Datorezi 100 lei

Se va reține că folosirea parantezelor este obligatorie, pentru a se preciza calculatorului începutul și sfîrșitul condiției în vederea evaluării ei corecte. Prin urmare, expresia dintre paranteze este considerată ca o condiție.

2) Condițiile logice pot fi folosite în cadrul multor instrucțiuni. De exemplu instrucțiunea

5 PRINT (X=10)

va afișa *0* dacă parametrului x i se atribuie valori diferite de *10*, respectiv va afișa *1* dacă $x = 10$. Explicația constă în faptul că expresia dintre paranteze este considerată de calculator ca o condiție și i se evaluează valoarea de adevăr.

În mod similar se poate proceda și în cazul altor instrucțiuni. De pildă, dacă parametrul x este *1*; *2*; *3* și *4* și se dorește ca să se atribuie variabilei y valorile *10* (pt. $x = 1$); *20* (pt. $x = 2$); *30* (pt. $x = 3$) sau *40* (pt. $x = 4$), se scrie

```
10 LET y=10*(x=1)+20*(x=2)+30*(x=3)+*40(x=4)
15 PRINT y
```

Tastînd LET x = 1 : GO TO 10 se va afișa *10*, iar tastînd orice valoare pentru x în afară de *1*; *2*; *3* sau *4*, se va afișa *0*.

Metoda este mai avantajoasă decît dacă s-ar folosi instrucțiunile IF –THEN, deoarece se economisește memoria.

● Claviatura poate fi testată cu instrucțiunile INKEY$.

Sintaxa : INKEY$
Exemplu : 10 IF INKEY$="d" THEN GO TO 100
Efect : citește și returnează tasta apăsată în momentul execuției instrucțiunii ; dacă nu s-a apăsat nici o tastă întoarce șirul nu (" ")

Evident contează cursorul la editare; astfel în *CAPS LOCK* (*CS* și *2*) sau la apăsarea cu *CS*, instrucțiunea/comanda *INKEY$* întoarce o majusculă.

Observații : 1) Între instrucțiunile INKEY$ și INPUT se pot face următoarele comparații :

— INPUT așteaptă introducerea datei și programul nu continuă pînă cînd această operație nu a fost executată (condiție obligatorie), pe cînd la INKEY$ această condiție poate să nu existe;

— INPUT poate citi oricîte caractere, pe cînd INKEY$ numai un caracter;

— la INPUT se apasă ENTER pentru validarea introducerii datelor în vreme ce INKEY$ nu necesită această operație.

2) De multe ori instrucțiunea INKEY$ este utilizată pentru comunicarea unei opțiuni. De exemplu :

```
10 IF INKEY$="d" OR INKEY$="D" THEN GO TO nr. linie1
20 IF INKEY$="n" OR INKEY$="N" THEN GO TO nr. linie2
```

Această secvență de instrucțiuni poate fi scrisă și sub forma prescurtată

```
10 GO TO nr. linie1 *(INKEY$="d" OR INKEY$="D")+nr. linie 2
   *(INKEY$="n" OR INKEY$="N")
```

În mod analog se procedează pentru alegerea unei opțiuni dintr-un număr de variante oferite:

```
200 GOTO nr. linie1*(INKEY$="1")+nr. linie2*(INKEY$="2")+
    nr. linie 3*(INKEY$="3"+ etc.
```

înlocuind forma clasică

```
200 IF INKEY$="1" THEN GO TO nr. linie1
210 IF INKEY$="2" THEN GO TO nr. linie2
220 IF INKEY$="3" THEN GO TO nr. linie3
```

Exemplul 5.30:

```
5 REM deplasarea unui caracter cu tastele 5, 6, 7, 8
9 CLS :PRINT # 0; AT 0, 0; "Cu tastele 5, 6, 7, 8
  deplaseaza ! !"
10 LET X=0; LET Y=0
15 PRINT AT X, Y; "■" :REM tasta 8 în modul
  grafic
20 LET a=X :LET b=Y
25 LET X=X+(INKEY$="6" AND X<21)-
  (INKEY$="7" AND X>0) :LET Y=Y+
  (INKEY$="8" AND Y<31)-(INKEY$="5"
  AND Y>0) :PRINT AT a, b; " "
30 IF X=31 OR Y=21 THEN GO TO 15
35 IF INKEY$="s" THEN STOP
40 GO TO 15
```

Din acest exemplu se observă *principiul realizării mișcării* care constă în tipărirea unui blanc pe poziția anterioară a caracterului care execută mișcarea.

Exemplul 5.31:

```
8 REM alegerea tastelor care comandă mișcarea
9 CLS
10 DIM k$(4) :PRINT AT 5, 0; "INTRODU
  TASTELE" ' ' "STINGA ;" ' "DREAPTA :"
  ' ' "SUS :" ' "JOS :"
20 FOR f=1 TO 4 :PRINT AT f*2+5, 11; "?"
30 LET R$=INKEY$ :IF R$ < > " "THEN
  GO TO 30
40 LET R$=INKEY$ :IF R$=" " THEN GO TO
  40
50 PRINT AT f*2+5, 11; R$ :LET k$(f)=R$
55 NEXT f
60 LET X=0 :LET Y=0; CLS
70 PRINT AT X, Y; ,"■" :REM tasta 8 în modul
  grafic
```

```
80 LET a=X :LET b=Y
90 LET X=X+(INKEY$=k$(4) AND X<21)-
   (INKEY$=k$(3) AND X>0) : LET Y=Y+
   (INKEY$=k$(2) AND Y<31)-(INKEY$=
   k$(1) AND Y>0) :PRINT AT a, b; "␣"
100 IF X=31 OR Y=21 THEN GO TO 70
110 IF INKEY$="s" THEN STOP
120 GO TO 70
```

5.5. PROBLEME

P5.1. : Se cere un program care cere și afișează componentele unui vector.

Rezolvare :

```
9 CLS :INPUT "Introdu nr. componentelor k", k
10 DIM d(k)
20 FOR i=1 TO k
30 INPUT "Introdu fiecare componentă". d(i)
40 PRINT d(i) ; "=d("; i;")"
50 NEXT i
```

P5.2. : Să se elaboreze un program care cere și afișează componentele unei matrice

Rezolvare :

```
9 CLS : INPUT "Introdu nr. liniilor (l) și al coloa-
  nelor (c)", l, c
10 DIM p(l, c)
20 FOR m=1 TO l
30 FOR n=1 TO c
40 INPUT "Introdu fiecare componentă", p(l, c)
50 PRINT p(l, c) ; "=p ("; m; ", ";n ;")"
60 NEXT m
70 NEXT n
```

P5.3 : Să se calculeze suma primelor N numere naturale

Rezolvare :

```
10 CLS : INPUT "Introdu nr. numelor N", N
20 LET S=0
30 FOR i=1 TO N
40 LET S=S+1
50 NEXT i
60 PRINT "SUMA PRIMELOR N NR. NATURALE :",s
```

P5.4 : Se cere să se calculeze factorial dé n (adică valoarea $k = n! = 1 * 2 * 3 * \ldots * n$).

Rezolvare :

```
10 CLS : INPUT "Introdu nr. N", N
20 LET k=1
30 FOR i=2 TO N
40 LET k=k * i
50 NEXT i
60 PRINT "Factorial de_"; N; "_este :", k
```

P5.5 : Să se afle valoarea maximă dintr-un șir de numere.

Rezolvare :

```
 5 CLS : INPUT "Introdu nr. de valori n", n
10 DIM a(n)
15 LET q=1
20 INPUT "Următorul element", a(q) : PRINT "elem."; q;"="; a(q)
25 IF q<n THEN LET q=q+1 :GO TO 20
30 LET max=a(1) :LET q=2
35 IF a(q)>max THEN LET max=a(q)
40 IF q<n THEN LET q=q+1 :GO TO 35
50 PRINT "Maximul este : "; max
```

P5.6 : Se cere valoarea unui determinant de ordinul 2.

Rezolvare : un determinant de ordinul 2 are forma

$$\begin{vmatrix} a11 & a12 \\ a21 & a22 \end{vmatrix}$$

și valoarea : $A = a11 * a22 - a12 * a21$.

```
10 CLS : DIM p(2, 2)
20 FOR m=1 TO 2
30 FOR n=1  TO 2
40 INPUT "Introdu fiecare valoare", p (m, n)
50 PRINT p (m, n) ; "=p("; m", "; n;")"
60 NEXT m
70 NEXT n
80 LET A=p (1, 1) * p(2, 2) -p(1, 2) * p(2, 1)
90 PRINT AT 10, 1 ; "Valoarea determinantului :";
   AT 14, 0 ; "A="; A
```

P5.7 : Să se elaboreze un program care afișează valorile standardizate pentru secțiunile penelor (pentru a nu se lungi programul se vor lua *4* secțiuni conform tabelului următor :

Diametrul arborelui d [mm]	Secțiunea penelor	
	b[mm]	h[mm]
6...8	2	2
>8...10	3	3
>10...12	4	4
>12...17	5	5

```
5 CLS : LET n=4 : REM n—numărul valorilor standardizate
10 DIM l(n) : DIM r(n) : DIM b(n) : DIM h(n) : REM l—valoarea
   minimă a intervalelor diametrelor arborelui ; r—valoarea maximă
   a intervalului diametrelor ; b, h—dimensiunile secțiunii penei
20 FOR i=1 TO n : READ l(i), r(i), b(i), h(i) : NEXT i
30 DATA 6, 8, 2, 2
40 DATA 9, 10, 3, 3
50 DATA 11, 12, 4, 4
60 DATA 13, 17, 5, 5
70 INPUT "Introdu diametrul arborelui d[mm]", d
80 FOR i=1 TO n
90 IF d>l(i) AND d<=r(i) THEN GO TO 200
91 IF d<=l(1) THEN LET d=l(1) : GO TO 200
92 IF d>r(n) GO TO 100
93 NEXT i
100 PRINT "NU EXISTA VALOARE STANDARDIZATA" : STOP
200 PRINT AT 16, 4 ; "Diametrul arborelui d=" ;d ;"_mm" ; AT
   18, 1, "Dimensiunile secțiunii penei :" ; AT 19, 10 ; "b=" ;b(i) ;
   "_mm" ; AT 19, 20 ; "h=" ;h(i) ; "_mm"
```

Observație : se poate introduce și linia **15 RESTORE 30**

P5.8 : Să se aleagă materialul optim pentru arbori (pentru scurtarea programului se vor folosi numai *4* date conform tabelului următor :

Materialul	Tensiunea la rupere prin oboseală σ_{-1} [daN/cm²]
OL50, OT50	2400
OLC45	2500
OL60	2800
20C08	3000

Se va alege materialul pe baza relației $\sigma_{-1necesar} \geqslant \sigma_{-1}$

```
5 CLS : LET n=4 :REM n—numărul materialelor
10 DIM s(n) :DIM s$(n, 20) : REM s—valorile tensiunilor la rupere prin oboseală ; s$—denumirile materialelor cu max. 20 caractere
20 FOR i=1 TO n : READ s(i) : NEXT i
22 DATA 2400, 2500, 2800, 3000 : REM valorile tensiunilor la rupere prin oboseala ordonate crescător
25 FOR i=1 TO n :READ s$(i) :NEXT i
27 DATA "OL50, OT50", "OLC45", "OL60", "20C08" : REM denumirile materialelor
40 INPUT "Introdu tensiunea la oboseală necesară materialului arborelui în daN/cm↑2" 'sig
50 FOR i=1 TO n
55 IF sig>s(n) THEN GO TO 100
60 IF sig=s(i) THEN GO TO 200
65 IF sig<s(i) THEN LET sig=s(1) : GO TO 200
70 IF sig<s(i+1) THEN LET i=i+1 : GO TO 200
80 NEXT i
100 PRINT "NU EXISTA MATERIAL" ; STOP
200 PRINT AT 14, 4; "Materialul arborelui :"; AT 16, 12; s$(i); AT 18, 4; "Tensiunea la oboseală :"; AT 19, 6, s(i); "_daN/cm↑2"
```

P5.9 : Să se aleagă materialul optim pentru un cuzinet (aplicație restrînsă pentru datele din tabelul următor) :

Denumire	Ungere	v[m/s]	p[daN/cm²]	pv[daN/(cm²·s)]
Materiale sinterizate	U, M	1	60	50
Poliamida	U, M	2	60	60
Lignofon	U, M	2	20	50
FCA-1	M, F	2	90	18

Alegerea se face pe baza relațiilor : $v_{calculat} \leqslant v_{calculat}$; $p_{calculat} \leqslant p$ și $pv_{calculat} \leqslant pv$ (U — uscată ; M — mixtă ; F — fluidă).

```
5 CLS : LET n=4 :REM n—nr. materialelor
10 DIM m$(n, 30) :REM 30—nr. caracterelor din denumirile materialelor și felul ungerii
11 FOR i=1 TO n :READ m$(i) : NEXT i
```

```
12 DATA "Materiale sinterizate U, M", "Poliamida U, M",
   "Lignofon U, M", "FeA-1 U, M"
13 DIM v(n) :DIM p(n) :DIM s(n) :REM v—viteza fusului ; p—presiunea de contact ; s=pv—puterea specifică de frecare
14 FOR i=1 TO n :READ v(i), p(i), s(i) :NEXT i
15 DATA 1, 60, 50, 2, 60, 60, 2, 20, 50, 2, 90, 18 :REM valorile numerice din tabel citite pe orizontală
16 CLS
21 INPUT "Introdu viteza, presiunea și puterea specifică de frecare (valori calculate)", v, p, s
22 FOR i=1 TO n
25 IF v<v(1) AND p<p(1) AND s<s(1) THEN LET v=v(1) : LET p=p(1) :LET s=s(1)
26 TF v>v(n) AND p>p(n) AND s>s(n) THEN PRINT "NU EXISTA MATERIAL" : STOP
30 IF v(i) ≥ v AND p(i) ≥ p AND s(i) >[s THEN PRINT "␣ ␣ ␣ ␣ ␣
   ␣ ␣ ␣ ␣ ␣ ␣ ␣" ' ' 'm$(i) ' "v="; v(i) ; "␣m/s"; ' ' "p="; p(i) ; "␣daN/cm ↑ 2"; "" "pv=" : s(i) :"␣daN/cm↑ 2s" ' ' ' ' ',
   ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ ␣ "
40 NEXT i
```

P5.10 : Să se afișeze pe întregul ecran semnul #.

Rezolvare :

```
10 CLS :FOR x=0 TO 21 :FOR y=0 TO 31
20 PRINT AT x, y; "#"
30 NEXT y : NEXT x
```

Capitolul 6

FUNCȚII STANDARD, FUNCȚII UTILIZATOR ȘI SUBRUTINE

6.1. FUNCȚII STANDARD

Funcțiile sînt instrucțiuni generalizate pentru rezolvarea de probleme. Se disting funcții standard :

— *aritmetice:* **PI, SIN, COS, TAN, ASN, ACS, ATN, LN, EXP, SQR, ABS;**

— *de sistem:* **INT, SGN, LEN, VAL, STR$, CHR$**

— *pentru generarea numerelor aleatoare:* **RND, RANDOMIZE.**

O funcție standard este o *procedură denumită* care, pornind de la valoarea argumentului ei, generează o *valoare calculată.*

Sintaxa: **nume (e . a)**
unde *(e.a)* este un număr, o variabilă simplă sau o expresie aritmetică

În tabelul 6.1 sînt prezentate funcțiile standard.

TABELUL 6.1

Nr. crt.	Numele funcției	Semnifiația	Exemple sau precizări
1	**ABS***(e.a)*	Valoare absolută	**ABS(B)**=20 (dacă $B = 20$ sau $B = -20$) **PRINT ABS** $(8*(-3)) = 24$
2	**ACS** *(e.a)*	Arc cosinus	Calculează arc cosinusul valorii *(e.a)* cu rezultatul în radiani ; $-1 \leqslant e.a \leqslant 1$
3	**ASN** *(e.a)*	Arc sinus	Calculează arc sinusul valorii *(e.a)* cu rezultatul în radiani ; $-1 \leqslant e.a \leqslant 1$

4	**ATN** *(e.a)*	Arc tangentă	Calculează arc tangenta valorii *(e.a)* cu rezultatul în radiani ; $-\pi/2 \leqslant e.a. \leqslant \pi/2$ **ATN(1)** = *0*.7853981
5	**COS** *(e.a)*	Cosinus	Calculează cosinusul valorii *(e.a)* în radiani
6	**EXP** *(e.a)*	Exponențiala	Calculează $e^{(e.a.)}$ **PRINT EXP(1)** = 2.828818
7	**INT** *(e.a)*	Parte întreagă (întoarce cel mai mare întreg $\leqslant$ *(e.a)*	**INT (4.9)** = *4* **INT(− 4.9)** = − 5
8	**LEN** *(e.a)*	Lungimea unui șir	Calculează lungimea unui șir inclusiv spațiile și semnele incluse. **PRINT LEN "BASIC"**=5
9	**LN** *(e.a)*	Logaritm natural	Calculează logaritmul natural al unui argument pozitiv **PRINT LN**(*2.72*) = 1.*0*006319
10	**PI**	Numărul π	**PRINT PI** = 3.1415927
11	**RND**	Numere aleatoare	Generează numere aleatoare cuprinse între *0* și *1* (poate lua valoarea *0* dar niciodată 1). Pentru numere aleatoare în alte intervale *6****RND** (între *0* și *6*) *2.2*+*0.8****RND** (între *2.2* și 3) *2*+**INT** (**RND****7*) (întregi între *2* și *7*)
12	**RANDOMIZE** (RAND pe claviatură)	Numere aleatoare	Determină ca **RND** să genereze numere aleatoare dintr-un punct definit al secvenței de numere. Are sintaxa : **nr. linie RANDOMIZE k** unde $k \in [1; 65535]$ și repre-

			zintă numărul de ordine al viitorului apel al funcției **RND**. Cînd $k = 0$ sau lipsește baza de pornire o constituie timpul scurs de la punerea în funcțiune a calculatorului.
13	**SGN** (*e.a*)	Semnul *e.a*	1 cînd $e.a > 0$ -1 cînd $e.a < 0$ 0 cînd $e.a = 0$
14	**SIN** (*e.a*)	Sinus	Calculează sinusul valorii (*e.a*) în radiani **SIN(PI) = 0**
15	**SOR** (*e.a*)	Rădăcină pătrată	Condiție : $e.a > 0$ **PRINT SOR(9) = 3** **PRINT SOR(− 4)** = eroare
16	**STR$**	Convertește numere în șir	**STR$(100) = "100"**
17	**TAN** (*e.a*)	Tangenta	Calculează tangenta valorii (*e.a*) în radiani **PRINT TAN(PI/4) = 1**
18	**VAL** (*e.a*)	Convertește șir de numere	**PRINT VAL** *"6.6"* = 6.6
19	**CHR$***x*	$x \in (0..255)$	Oferă caracterul al cărui cod *ASCII* este x; rezultatul este un șir. **PRINT CHR$** (*65*) = A

Observații : În legătură cu funcția **CHRS** un interes aparte îl prezintă numerele între *0* și *32*, care reprezintă caracterele de comandă. Astfel :

1) *CHR$ 6* este echivalent cu **PRINT** virgulă

Exemplu : *10* **PRINT** *10* ; **CHR$***6*; *20* determină scrierea cifrei *20* în coloana *16* ;

2) *CHR$ 8* mută cursorul la stînga cu un caracter, iar *CHR$ 9* mută cursorul la dreapta cu un caracter.

Exemplu : *10* **PRINT** *"5432"* ; **CHR$***8* ; *"6"* **tipărește** *5436*

20 **PRINT** *"5432"* ; **CHR$***9* ; *"6"* **tipărește** *54326*

3) *CHR$ 22* este echivalent cu **AT**

Exemplul: **10 PRINT CHR$22+CHR$5+CHR$5; "ROMANIA"**

este echivalent cu **10 PRINT AT 5,5; "ROMANIA"**

4) *CHR$ 13* trece pe un rînd nou.

Tabelul 3.1 oferă toate posibilitățile de folosire a funcției **CHRS**

Cunoscînd aceste date se pot realiza mici efecte la salvarea programelor pe casetă. După cum se știe, calculatorul scrie automat pe linia *0* și coloana *0* textul

Program : nume (cu max. *10* caractere)

Acest titlu poate fi modificat sau completat așa cum se indică în exemplele următoare :

a) **SAVE CHR$8+CHR$4" at de MP"** care determină mesajul *„Programat de MP"*

b) **SAVE CHR$8+CHR$8+"ul__PENE"** care determină mesajul *„Programul PENE"*

8 caractere

c) Titlu scintilat (flash) :**SAVE CHR$18+CHR$1+"nume 8 caractere"**

d) Titlu în mijlocul ecranului : **SAVE CHR$22+CHR$10+CHR$13 + +"nume 7 car."**

e) Titlu între ghilimele : **SAVE CHR$34+"nume 8 car."+CHR$34**

f) Combinații de litere folosind tastele calculatorului

SAVE CHR$22+CHR$1+CHR$0+"SA"+CHR$180+CHR$255+ "©" determină mesajul

SATAN COPY ©

9) Titlu în partea din dreapta jos a ecranului :

SAVE CHR$22+CHR$21+CHR$25+"nume 7 car."

Exemplul 6.1:

```
9 REM simularea aruncării unui ziar
10 LET zar=INT(RND*6)+1
20 PRINT "S-a aruncat :";zar :PAUSE 0 :CLS :
   GO TO 10
```

Exemplul 6.2:

```
9 REM tipărire de numere aleatoare crescătoare
10 RANDOMIZE 0
20 PRINT RND : GO TO 10
```

Dacă se doresc numere aleatoare diferite atunci linia *20* devine

```
20 PRINT RND : GO TO 20
```

Exemplul 6.3: Programul determină frecvența de apariție a „capului" și a „pajurei" la aruncarea unei monezi ; dacă timpul de rulare este mare,

frecvența devine 1 deoarece numerele aleatoare generate sînt uniform repartizate în intervalul *0 . . . 1.*

```
10 CLS
15 LET cap=0 : LET pajura=0
20 LET moneda=INT(RND*2)
25 IF moneda=0 THEN LET cap=cap+1
30 IF moneda=1 THEN LET pajura=pajura+1
35 IF pajura=0 THEN GO TO 15
50 PRINT AT 0, 0; "Cap"; AT 0, 4; "pajura"; AT 0, 11; "Frecventa de aparitie"
55 PRINT TAB 2; cap; TAB 7; pajura; TAB 15, cap/pajura : PAUSE 50
60 PRINT : GO TO 20
```

Exemplul 6.4:

```
9 REM folosirea funcției SQR
10 RESTORE 20 : READ A, B, C
20 DATA 4, 5, −2
30 LET D=b ↑ 2−4*A*C
40 LET x1=(−B+SQR(D))/(2*A)
50 LET x2=(−B−SQR(D))/(2*A)
60 PRINT x1, x2
```

Se afișează : *0 .3187293* *− 1 .5687293*

Exemplul 6.5: valorile funcțiilor trigonometrice sinus și cosinus

```
8 CLS : PRINT AT 0, 0; "Grade"; AT 0, 8; "Sinus"; AT 0, 21; "Cosinus"
10 FOR g=0 TO 360
12 LET gr=g*PI/180 : REM transformarea gradelor în radiani
15 LET s=SIN (gr) :LET c=COS (gr)
20 PRINT TAB 1; g; TAB 6; s; TAB 20; c :PAUSE 40
25 NEXT g
```

Exemplul 6.6: exponențiala

```
9 CLS :PRINT AT 0, 0; "Argumentul x"; AT 0, 14; "Exponențiala e ↑ x"
10 FOR x=0 TO 75
20 LET a=EXP (x)
30 PRINT TAB 5; x; TAB 16; a; PAUSE 40
40 NEXT x
```

Exemplul 6.7: logaritmul natural și cel zecimal

```
10 CLS :PRINT AT 0, 0; "Arg.i"; AT 0, 6; "Log. natural"; AT 0,
   19; "Log. zecimal"
20 FOR i=1 TO 100
30 LET a=LN (i) :LET b=LN (i)/2.3025851 :REM transformarea
   logaritmului natural în logaritm zecimal
40 PRINT TAB 1; i; TAB 7; a; TAB 20; b: PAUSE 40
50 NEXT i
```

Exercițiul 6.8: să se numere cîte caractere de un anumit tip se află într-un șir dat:

```
5 CLS :INPUT "Introdu șirul (max. 32 caractere)", a$ :IF a$=" "
  THEN GO TO 5
7 INPUT "Introdu caracterul căutat", b$ :IF b$=" " THEN
  GO TO 7
8 LET b$=b$(1) :LET k=0 :REM k—contor
10 FOR i=1 TO LEN a$
15 IF a$(i)=b$ THEN LET k=k+1
20 PRINT a$(i);
25 NEXT i
30 PRINT AT 20, 0; "Caracterul_"; b$; "_figurează de_";
   AT 21, 8; k; "_ori"
```

Observație: cu modificările

```
9 PRINT AT 0, 6; "Sirul de analizat :"; AT 2, 0; a$
```

linia *20* se scoate.

```
30 PRINT AT 20, 0; "Caracterul_"; b$; "_figurează de_";
   AT 21, 8; k; "_ori"
```

rezultă un program mai clar.

6.2. FUNCȚII UTILIZATOR DEF(FN, FN)

Sînt definite de utilizator cu scopul folosirii lor într-un program în același mod ca și funcțiile standard.

Sintaxa: **nr. linie DEF FN nume([p1][,p2],. . ., [pn]) = expresie FN nume (v1, v2, . . ., vx)**

unde: —*nume* — numele funcției format dintr-o *singură literă* (pentru funcții numerice) sau o *literă urmată de semnul $* (pentru funcțiile șiruri de caractere); fiind 26 de caractere ale alfabetului, într-un program pot exista 26 de funcții;

— *parametrii p* — nume de variabile ce constituie parametrii *formali* ai funcției (max. *26*); la apelarea funcției, parametrii se înlocuiesc cu *valori concrete* (*v*);
— *expresie* — expresie matematică formată cu constante, parametri și operatori aritmetici sau de concatenare, putînd conține și funcții standard dar nu și funcții utilizator; prin formula acesteia se calculează valoarea funcției.

Exemple:
```
20 DEF FN A(B)=B↑6
30 DEF FN R(X)=(INT(x+0.05)*100)/100
40 DEF FN W(X)=SIN(X)/COS(X)
50 DEF FN c$(a$)=a$(2 TO)+a$(TO 1)
60 DEF FN r(x)=INT(x+0.5)
```
— funcția rotunjire

Efect: permite utilizarea unei funcții speciale într-un program, care se *scrie oriunde în cadrul programului*. La întîlnirea funcției FN se evaluează valoarea funcției definite în raport cu valorile *v1, v2, ..., vx* care se atribuie variabilelor simbolice *p1, p2, ..., pn*. Dacă *x* nu este egal cu *n* sau locul unui parametru șir s-a introdus unul numeric (sau în invers), apare mesajul de eroare *Q*; de asemenea, la încercarea de a se evalua o funcție nedefinită se va răspunde cu mesajul de eroare *P*.

Exemplul 6.9:
```
10 CLS :LET x=0 :LET y=0 :LET a=10
20 DEF FN p(x, y)=a+x*y
30 DEF FN q( )=a+x*y
40 DEF FN r(a)=FN p(FN q( ), 6)
50 PRINT FN p(2, 3), FN q( ), FN r(0)
```

Programul afișează:

```
 16          10
 70
```

Iată cum s-a calculat: **FN p(2, 3)** = $a + 2*3 = 10 + 6 = 16$ (folosind $a = 10$)

FN q( ) = $a + x*y = 10 + 0 = 10$ (funcție fără parametri)

FN r(0) = FN p(10, 6) = $a + 10*6 = 70$

Exemplul 6.10:
```
10 CLS :LET k=30
20 DEF FN T(x)=x/2+10
30 LET z=FN T(k) :PRINT z
```

Se afișează *25*.

Exemplul 6.11: calculul unui polinom de gradul doi ($p = ax^2 + bx + c$)

```
10 CLS :DEF FN f(x)=a*x*x+b*x+c
20 INPUT "Introdu valorile a, b, c, x", a,, b,, c,,x
30 PRINT "Valoarea funcției f(x)="; FN f(x)
40 PRINT AT 21,9; "Reluăm (d/n)?"
50 PAUSE 4e4 :GOTO 10*(INKEY$="d")+100*(INKEY$="n")
100 CLS :STOP
```

Observație: în unele probleme apare necesitatea rotunjirii unui rezultat numeric la valoarea întreagă imediat superioară; în acest scop se folosește instrucțiunea

```
10 DEF FN R(x)=INT(x)+(x<>INT(x))
```

Exemplu:

```
10 DEF FN r(x)=INT(x)+(x<>INT(x))
100 PRINT AT 13, 2; "Dal=m(z1+2+2×1)";
    AT 13, 20; "Dal="; FN r(m*(z1+2+2*x1);
    "␣mm"
```

Linia *100* calculează diametrul cercului de cap la o roată dințată (afișînd relația de calcul), pe baza valorilor date anterior în program pentru numărul de dinți *z1* și deplasarea specifică *x1*; valoarea numerică este rotunjită la numărul întreg de milimetri imediat superior.
Reducerea la *n* zecimale a unui număr real *x*: **INT((10 ↑ n)*x)/(10 ↑ n)**

6.3. SUBRUTINE *(GOSUB—RETURN)*

Subrutinele sînt secvențe de instrucțiuni care se execută de mai multe ori și în locuri diferite din program. Ele pot fi asimilate cu o instrucțiune **GO TO** cu retur.

Sintaxa:

```
nr. linie GOSUB etichetă
      :
nr. linie RETURN
```

Exemple:

```
140 GOSUB 400
  :
400 LET x=v+s
410 LET z=SQR(x)      Secvența de instrucțiuni
420 PRINT z             a subrutinei
430 RETURN

100 REM programul principal
110 GOSUB 200
120 PRINT A, B, C :STOP
```

```
200 REM subrutina apelată exterioară
210 INPUT A, B
220 GOSUB 300
230 LET C=c ↑ 2
240 RETURN
300 REM subrutina apelată exterioară
310 LET C=A+B
320 RETURN
```

Efect: transferă controlul programului la instrucțiunea cu eticheta specificată în instrucțiunea **GOSUB** (începutul subrutinei) și comandă revenirea la prima instrucțiune după **GOSUB** atunci cînd se execută instrucțiunea **RETURN** (sfîrșitul subrutinei).

Observații: 1) Într-un program subrutinele se scriu, de regulă, la sfîrșitul programului. Forma cea mai elegantă de scriere, care aduce și economie de memorie, este următoarea:

```
10 LET s1=9000 :LET s2=9100 :LET s3=9200 :REM stabilirea
   numărului de linie de început pentru fiecare din cele trei subrutine
   s1, s2, s3
.
.
8999 STOP
9000 LET d=b*b-4*a*c :RETURN
9100 LET x1=(-b+SQR(d))/(2*a) :PRINT x1 :RETURN
9200 LET x2=(-b-SQR(d))/(2*a) :PRINT x2 :RETURN
```

Acest program a trecut în subrutinele menționate discriminantul și expresiile de calcul ale rădăcinilor (*x1*, *x2*) unei ecuații de gradul doi (cînd subrutinele sînt la sfîrșit, atunci trebuie **STOP** la sfîrșitul programului *BASIC* principal).

2) Memoria unde se memorează trimiterile **GOSUB** se numește „*memorie GOSUB*" (în engleză **GOSUB STACK** — stivă *GOSUB*). Funcționarea este următoarea: cînd se întîlnește o instrucțiune **GOSUB** se memorează locul din program unde această instrucțiune este scrisă, după care programul face un salt la linia cu etichetă precizată parcurgîndu-se instrucțiunile subrutinei pînă la întîlnirea lui **RETURN**. În acest moment calculatorul citește din „*memoria GOSUB*" locul de unde s-a dat *ultimul GOSUB* și sare la instrucțiunea imediat următoare acesteia, *ștergînd din memorie ultimul GOSUB* pentru ca nu cumva următorul **RETURN** să se întoarcă în același loc. Acest tip de a memora în care ultima informație intrată iese prima se numește *LIFO (Last In First Out)*.

Exemplul 6.12:

```
10 CLS :INPUT "Introdu Y", Y :IF Y=0 THEN
   GO TO 10
20 GOSUB 110
30 STOP
110 INPUT "Introdu x", x :IF x=0 THEN
   GO TO 110
120 LET x=2*y/x
130 PRINT "x="; x
140 RETURN
```

Programul cere la linia *10* să se introducă o valoare pentru y, după care comandă treecerea la subrutina de la linia *110* unde se cere introducerea unei valori pentru x; în continuare se calculează noua valoare a lui $x(=2y/x)$ care se afișează și apoi programul revine la linia *30* unde se oprește.

Exemplul 6.13:

```
9 REM folosirea mai multor subrutine
10 INPUT "Introdu Y", Y :IF Y=0 THEN CLS :
   GO TO 10
20 GOSUB 110
30 STOP
110 INPUT "Introdu x", x :IF x=0 THEN GO TO 110
120 IF x>0 THEN GO TO 170
130 LET x=2*y/x
140 PRINT "x=", x
150 RETURN
160 LET x=x+1
170 INPUT "Introdu "z", z
180 LET x=y/x+z
190 PRINT "x="; x
200 RETURN
```

6.4. PROBLEME

P6.1: Să se scrie un program care distribuie aleator (întîmplător) elementele unui șir de caractere (max. *32*).

Rezolvare: se generează un număr aleator între *1* și lungimea șirului; elementul care se află pe acest loc este-trecut la sfîrșitul șirului, după ce toate elementele de după el au fost trecute cu un pas înainte.

În continuare se consideră lungimea șirului ca fiind cu *1* mai mică și se reia rearanjarea elementelor.

```
10 CLS : RANDOMIZE
15 INPUT "Introdu nr. de caractere n", n :DIM a$(n) :INPUT
   "Introdu șirul de caractere", a$
20 FOR i=1 TO n-1
25 LET q=INT(RND*(n-i)+1)
30 LET b$=a$(q)
40 FOR a=q TO (n-1) :REM elementele se mută cu 1 pas înainte
50 LET a$(a)=a$(a+1)
60 NEXT a
70 LET a$(n)=b$
80 NEXT i
90 PRINT a$; AT 21, 4; "Apasa o tasta oarecare" : PAUSE 0:
   CLS :GO TO 20
```

Programul anterior poate servi ca model la elaborarea unui joc în care se amestecă un număr de cărți de joc, sau la conceperea unui joc de *SCRABLE* pentru aranjarea a *100* de litere, ș.a.

P6.2 : Să se conceapă un program care afișează vocalele unui text dat (de max. *32* caractere).

Rezolvare : se compară fiecare caracter al șirului cu cele *5* vocale (*a*, *e*, *i*, *o*, *u*) recunoscute de calculator.

```
10 CLS :INPUT "Introdu numărul caracterelor (max 32)", n :IF
   n<0 OR n>32 THEN GO TO 10
20 DIM t$(n) :INPUT "Introdu textul", t$ :CLS
25 PRINT "Vocalele din text sînt : 8blancuri Poziția 5blancuri
   Vocala"
40 FOR i=1 TO LEN t$
50 IF t$(i)="a" OR t$(i)="A" OR t$(i)="e" OR t$(i)="E" OR
   t$(i)="i" OR t$(i)="I" OR t$(i)="o" OR t$(i)="O" OR
   t$(i)="u" OR t$(i)="U" THEN PRINT i, t$(i)
60 NEXT i
70 PRINT 19, 0; t$ :PRINT # 0; AT 0, 9; "RELUATI (d/n) ?"
   : PAUSE 0
80 GOTO 10*(INKEY$="d" OR INKEY$="D")+90*(INKEY$
   ="n" OR INKEY$="N")
90 STOP
```

P6.3. Să se scrie rezolvarea ecuației de gradul doi

Rezolvare :

```
10 CLS :INPUT "Introdu coeficienții a, b, c", a, b, c
20 IF a=0 THEN GO TO 60
30 LET d=b ↑ 2-4*a*c
40 IF d<0 THEN PRINT "Ecuația nu are rădăcini
   reale" :GO TO 70
50 IF d=0 THEN LET x1=-b/(2*a) :LET x2=x1 :
   GOSUB 100 : GO TO 70
55 LET x1=(-b+SQR(d))/(2*a) :LET x2=(-b-
   SQR(d))/(2*a) :GOSUB 100 :GO TO 70
60 LET x=-c/b :PRINT "      ECUAȚIE
   DE GRADUL INTII" · · · "
   x="; x
70 STOP
100 PRINT AT 5, 3; "Radacinile ecuației" ";a;" x ↑ 2+
   (";b;")x+(";c;")" sînt :"; AT 9, 5; "x1=";
   x1; AT 11, 5; "x2="; x2 :RETURN
```

P6.4. Se cere însumarea elementelor unui vector

Rezolvare :

```
4 CLS :INPUT "Introdu numărul de elemente ale
  vectorului N", n :IF n=0 OR n<0 THEN GO TO 4
5 PRINT "   Vecto ul N  are ";n; " ele-
  mente" ' "cu valorile :"
6 DIM v(n)
10 FOR i=1 TO n
20 INPUT "Introdu valorile elementelor vectorului",
   v(i) :PRINT v(i),
30 NEXT i
40 GOSUB 100
50 PRINT AT 15, 0; "Suma elementelor vectorului :"
   ' "      s="; s
60 STOP
100 LET s=0
110 FOR i=1 TO n
120 LET s=s+v(i)
130 NEXT i
140 RETURN
```

P6.5 : Să se calculeze produsul scalar a 2 vectori cu n componente

Rezolvare :

```
5 CLS :INPUT "Introdu numărul de elemente al vectorilor", n :IF n=0 OR n<0 THEN GO TO 5
6 PRINT "Se efectuează produsul scalar a 2 vectori avînd fiecare␣"; AT 3, 8; "N="; n; "␣componente"
10 DIM x(n) :DIM y(n)
15 PRINT "Introdu elementele vectorului x"
20 FOR i=1 TO n :INPUT x(i) :NEXT i
25 PRINT "Introdu elementele vectorului y"
30 FOR i=1 TO n :INPUT y(i) :NEXT i
40 GOSUB 100
50 PRINT AT 19, 5; "Produsul scalar este :" ···
   ␣␣␣␣␣␣ p="; p :STOP
100 LET p=0
110 FOR i=1 TO n
120 LET p=p+x(i)*y(i)
130 NEXT i
140 RETURN
```

Capitolul 7

INSTRUCȚIUNI PENTRU PRODUCEREA SUNETELOR ȘI FOLOSIREA CULORILOR

7.1. PRODUCEREA SUNETELOR *(BEEP)*

Calculatoarele compatibile cu *ZX SPECTRUM* sînt prevăzute cu un difuzor pentru a produce sunete cu ajutorul instrucțiunii **BEEP**.

Sintaxa: **nr. linie BEEP d,i**

unde *d* — expresie numerică ce ndică *durata* sunetului în secunde (*d* = *0* sunet nul) putînd lua valori întregi sau fracționare;

i — expresie numerică ce reprezintă *înălțimea sunetului* măsurată în semitonuri relative la *do central* (care are valoarea *0* conform schemei următoare)

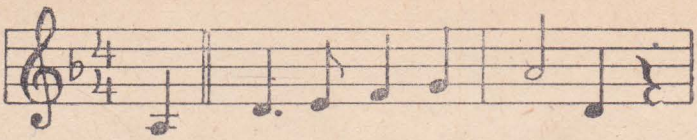
Nota	Do	Do #	Re	Re #	Mi	Fa	Fa #	Sol	Sol #	La	La #	Si	Do
i	0	1	2	3	4	5	6	7	8	9	10	11	12

Notele din octavele superioare sau inferioare se obțin adunînd sau scăzînd cifra *12* din valoarea *i*, pentru fiecare octavă; în general $i \in [-60; 69]$.

Durata notelor se stabilește ca în tabelul următor, iar codificarea unei partituri se face conform modelului din fig. 7.1.

Nota	Durata		Durata
întreagă	4		2
doimea	2	sau	1
pătrimea	1		1/2
optime	1/2		1/4
șaisprezecimea	1/4		1/8

Schimbarea cheii partiturii se face adunînd o variabilă (de ex : *c*) la înălțimea fiecărei note. Viteza de execuție se poate modifica folosind o variabilă (de ex : *p*) căreia i se atribuie valori diferite. Fiecărei partituri i se atașează un set de numere într-o instrucțiune **DATA**, reprezentînd datele referitoare la notele muzicale.



d	d/4	d/4+.5	d/8	d/4	d/4	d/2	d/4
1	-3	2	4	5	7	9	2

```
10 LET d=1:BEEP d/4,-3:BEEP(d/4+.5),2:BEEP d/8,4:BEEP d/4,5:
BEEP d/4,7:BEEP d/2,9:BEEP d/4,2
```

Fig. 7.1 Codificarea unei melodii

Exemplu : 5 BEEP 1,5 (se execută nota *fa* timp de o secundă)

Efect : instrucțiunea comandă executarea unui anumit sunet cu o anumită durată ; pentru fiecare sunet se scrie o instrucțiune BEEP.

Exemplul 7.1 :

```
9 CLS :PRINT AT 2, 9; "Gama centrală"
10 FOR i=0 TO 12 :BEEP .1, i :NEXT i :REM
   gama centrală parcursă din semiton în semiton
```

Exemplul 7.2 :

```
9 CLS :PRINT AT 2, 9; "Toate sunetele"
10 FOR i=-60 TO 69 :BEEP .1, i :NEXT i :REM
   toate sunetele emise de calculator
```

Exemplul 7.3 :

```
10 FOR I=1 TO 8 :READ H, J :BEEP H, J :NEXT I
20 DATA .1, 11, .1, 11, .3, 16, .05, 11, .05, 16, .05,
   11, .05, 16, 1, 20
```

Exemplul 7.4 :

```
9 REM schimbarea cheii și a duratei
14 PRINT AT 2, 4; "Audiere în do minor"
20 LET c=0 :BEEP 1, c+0 :BEEP 1, c+2 :BEEP
   1, c+3 :BEEP 1, c+2 : BEEP1, c+0
30 PAUSE 20 :CLS :PRINT AT 10, 4; "Audiere în
   re minor"
40 LET c=2 :BEEP 1, c+0 :BEEP 1, c+2 :BEEP 1,
   c+3 :BEEP 1, c+2 :BEEP 1, c+0
50 PAUSE 20 :CLS :PRINT AT 20, 2; "Audiere în
   la minor cu durată modificată (alternantă)"
60 LET c=9 :LET p=1 :BEEP p/5, c+0 :BEEP p/6
   c+2 :BEEP p/4, c+3 :BEEP p/2, c+2 :BEEP,
   p, c+0
```

Exemplul 7.5 :

```
10 CLS :BEEP .5, 14 :BEEP .25, 10 :BEEP .25,
   12 :BEEP .25, 9 :BEEP .25, 10 :BEEP .5, 7 :
   BEEP .5, 19
20 BEEP .25, 18 :BBEP .25, 19 :BEEP .25, 16 :BEEP
   .25, 18 :BEEP .5, 14 :BEEP .25, 19 :BEEP .25, 14
30 BEEP .5, 15 :BEEP .5, 12 :BEEP .5, 5 :BEEP .5,
   17 :BEEP .25, 14 :BEEP .25, 15 :BEEP .25, 12 :
   BEEP .25, 14 :BEEP .5, 10 :BEEP .25, 15 :BEEP
   .25, 10
40 BEEP .5, 12 :BEEP .5, 9 :BEEP .5, 2 :BEEP .5
   14 :BEEP .25, 10 :BEEP .25, 12 :BEEP .25,
   9 :BEEP .25, 10 :BEEP .5, 7
```

Exemplul 7.6 :

```
9 REM sunete în cascadă
10 FOR g=40 TO -10 STEP -1 :BEEP .04, g :
BEEP .04, g-1 ;BEEP .04, g+1 :NEXT g
```

Exemplul 7.8 :

```
9 REM încheierea frazei muticale
10 FOR x=12 TO 40 :BEEP .02, x :NEXT x
20 BEEP .3, 40 :PAUSE 38 :BEEP 2.5, 0 :STOP
```

Exemplul 7.9 :

```
9 REM sunete duble
10 LET sunete=9255 :GOSUB sunete :STOP
9255 RESTORE 9400
9260 FOR a=1 TO 20
9265 READ k
9270 FOR w=1 TO 4 :BEEP .01, k :NEXT w
9280 NEXT a
9400 DATA 24, 26, 28, 29, 24, 24, 26, 28, 29, 24
9405 DATA 16, 17, 19, 23, 21, 19, 17, 16, 19, 23
```

Exemplul 7.10 :

```
8 CLS :PRINT AT 12, 3; "VINZATOAREA DE
  FLORICELE"
10 DATA 11, 13, 10, 9, 7, 1, 11, 13, 13, 6, 1, 7, 2,
   10, 10, 6, 7, 10, -1, 13, 13, -1, 7, 10
20 DATA 11, 13, 10, 9, 7, 1, 11, 13, 13, 6, 1, 7, 2,
   10, 10, 6, 7, 10, -1, 13, 13, -1, 7, 10
```

```
30 DATA 13, 15, 15, 14, 10, 10, 11, 13, 10, 13, 15,
   15, 11, 13, 10, 13, 15, 15, 9, 8, 3, 11, 8, 12, 7, 1, 7,
   2, 10, 10, 7, 7, 1, 11, 13, 10, 11, 13, 10
100 FOR p=1 TO 2:RESTORE 10:FOR m=0 TO
   31:READ n:READ a1:READ a2:LET k=19-
   n:BEEP .03, n+(12*p):BEEP .09, -a1-12:
   BEEP .06, -a2:NEXT m:NEXT p:FOR p=1
   TO 2
110 RESTORE 10:FOR m=0 TO 31:READ n:READ
   a1:READ a2:BEEP .035, n+(12*p):NEXT m:
   NEXT p:FOR p=1 TO 2:RESTORE 10:FOR
   m=0 TO 31
120 READ n:READ a1:READ a2:BEEP .035, n+
   (12*p)-4:NEXT m:NEXT p
125 BEEP .3, n+(12*p)-4
```

Observații : 1) Calculatorul poate fi transformat în *orgă electronică* cu ajutorul următorului program :

```
9 CLS:PRINT AT 14, 1; "TASTELE IMITA ORGA ELECTRO-
   NICA"
10 LET p=CODE INKEY$
11 PRINT AT 21, 4; "Apasă ENTER pt. încetare": IF CODE
   INKEY$=13 THEN CLS:GO TO 100
15 IF p=0 THEN GO TO 10
20 BEEP .04, (p-30)/2
30 GO TO 10
100 STOP
```

2) În cazul programelor lungi, pentru a se evita lipsa de atenție ce intervine datorită rutinei de a tasta, este bine ca tastele calculatorului să fie sonorizate cu comanda

```
POKE 23609,30
```

7.2. INSTRUCȚIUNI PENTRU CULORI *(BORDER, PAPER, INK, FLASH, BRIGHT)*

Calculatoarele compatibile cu *ZX SPECTRUM* au facilități color putînd folosi *8* culori — numerotate de la *0* la *7* — și introduse pe următoarele taste numerice :

0 — negru
1 — albastru
2 — roșu
3 — purpuriu (magenta)
4 — verde
5 — albastru deschis (cyan)
6 — galben
7 — alb

În cazul televizoarelor alb-negru aceste numere corespund unor *nuanțe de gri*, ordonate de la închis spre deschis. La pornirea calculatorului sistemul lucrează în alb-negru cu caractere negre pe fond alb. Ecranul este împărțit în pătrățele de cîte *8* × *8 pixeli*, iar în fiecare pătrățel pot figura doar *două culori* diferite : una este culoarea fondului (**PAPER**) și alta este culoarea cernelei (**INK**). Tipărirea poate fi făcută *normal* sau prin *pîlpîire* (scintilare). Se pot colora marginea (**BORDER**-ul), fondul (**PAPER**-ul) și cerneala (**INK**-ul) și se poate utiliza pîlpîirea (**FLASH**) sau strălucirea (**BRIGHT**).

Sintaxa : **nr. linie BORDER** e

nr. linie PAPER n

nr. linie INK n

nr. linie FLASH m

nr. linie BRIGHT m

unde $n = 0 \ldots 9$ (culorile sînt date între *0* și *7* ; *8* — transparența ; *9* — contrast)

$c = 0 \ldots 7$

$m = 0, 1$ sau 8 (*0* — inactiv ; *1* — activ ; *8* — caracterele afișate clipitor rămîn așa, iar cele care urmează vor fi afișate normal)

Observații : 1) **INK** și **PAPER** folosite drept *comenzi* schimbă culoarea caracterelor și a fondului pînă la viitoarea modificare : Exemplu :

5 PAPER 5 :INK 1 :PRINT "GATA"

2) **INK** și **PAPER** pot fi folosite în cadrul instrucțiunilor **PRINT** și **INPUT**, cînd introduc *culori temporare :*

10 INPUT PAPER 5; INK 2: "Introdu valoarea a"; PAPER 7; INK 0; a :PRINT AT 10, 4; PAPER 2; INK 6; "Valoarea a="; a

3) Este posibilă scrierea directă în **FLASH** fără instrucțiuni folosind tastele *Caps Shift* și *Symbol* în felul următor :

CS și SS, CS și 9, text

Se revine tastînd

CS și SS, CS și 8.

4) *Listing-ul* poate fi asemenea de *colorat* tastînd după numărul de linie

CS și *SS* apoi tasta culorii pentru cerneală (*0* la *7*)

5) Pentru economisire de memorie culorile pot fi comandate ca în exemplele următoare :

a) BORDER RND :PAPER RND :INK RND :CLS

b) BORDER SIN PI :PAPER SIN PI :INK SIN PI :CLS

c) BORDER VAL "0" :PAPER VAL"0" :INK VAL"0" :CLS

d) BORDER NOT PI :PAPER NOT PI :INK NOT PI :CLS

e) BORDER SGN PI :PAPER SGN PI :INK SGN PI :CLS

Exemplul 7.11 :

```
8 REM schimbarea culorilor pt. BORDER și INK
9 BORDER 7 :PAPER 7 :INK 0 :CLS :REM inițializarea culorilor : BORDER-ul și PAPER-ul albe și INK-ul negru
10 FOR x=0 TO 7 :BORDER x :PAUSE 50 :PAPER 7-x :PAUSE 25 :CLS :NEXT x :BORDER 7 : PAPER 7 :INK 0 :CLS
```

Exemplul 7.12 :

```
9 REM text colorat aleator
10 FOR i=0 TO 38
12 BORDER 1 :INK RND * 7 :REM cerneală colorată aleator
20 PAPER RND * 7 :REM fond colorat aleator
30 PRINT "LECTIA DE BASIC␣ ␣ ␣"
35 NEXT i
```

Exemplul 7.13 :

```
7 REM mozaic colorat
8 CLS : BORDER 5
10 FOR i=1 TO 99
12 LET a$=" "
20 FOR x=1 TO 7
30 LET a$=a$+CHR$(RND * 14+129) :REM folosirea caracterelor grafice (v. tabelul 3.1)
40 NEXT x
45 INK RND * 7
50 PRINT a$;
60 NEXT i
70 PRINT AT 21, 25; "5 caractere grafice de pe tastele 1 la 8"
```

Exemplul 7.14 :

```
4 REM mozaic fără sfîrșit
5 CLS :BORDER 0 :CLS
10 LET h=16 :LET v=11
20 LET x=INT(RND * 3-1) :LET y=INT (RND * 3-1)
30 INK RND * 7
```

```
40 FOR z=1 TO 20
50 PRINT AT v, h; CHR$ (143) :REM  caracterul
   pătrat
60 LET h=h+x
70 LET v=v+y
80 IF h<0 THEN LET h=31
90 IF h>31 THEN LET h=0
100 IF v<0 THEN LET v=21
110 IF v>21 THEN LET v=0
120 NEXT z
130 GO TO 20
```

Exemplul 7.15 :

```
9 REM combinații de culori
10 FOR b=0 TO 7
20 BORDER b :PAPER b :CLS
30 PRINT AT 6, 12; INK 9; b
40 FOR p=0 TO 7
50 PRINT AT p+8, 8; INK p; PAPER 9; p;
60 BEEP .5, b * x-20+p
70 FOR i=0 TO 7
80 PRINT INK i; PAPER p; "_"; i;
90 BEEP .01, i * 5
100 NEXT i
110 NEXT p
120 NEXT b
```

Exercițiul 7.16 : Să se realizeze un BORDER colorat cu dungi late

```
9 PAPER 5 :INK 1 :CLS : :PRINT # 0; AT 0;4;
  "APASATI O TASTA OARECARE"
10 IF INKEY$=" " THEN BORDER 1 :BORDER 3
  :BORDER 5 : BORDER 6 :BORDER 4 :
  BORDER 0 :PAUSE 1 :GO TO 10
20 IF INKEY$=CHR$(13) THEN STOP
```

Exercițiul 7.17 : Să se obțină straturi orizontale colorate

```
10 CLS :FOR f=0 TO 21 :PRINT AT f, 0; PAPER
   RND  7; "32 blancuri" :NEXT f
```

Exercițiul 7.18 : Să se realizeze un chenar colorat care să încadreze fondul (PAPER-ul) ecranului.

```
10 BORDER 0 :PAPER 0 :INK 7 :CLS
20 FOR n=0 TO 31 : PRINT AT 0, n; PAPER
   INT(RND * 7) ; FLASH 1 ; "_" ; AT 21, n ;
   PAPER INT(RND * 7) ; FLASH 1 ; "_" :NEXT n
30 FOR n=0 TO 21 :PRINT AT n, 0 ; PAPER INT
   (RND * 7) ; FLASH 1 ; "_" ; AT n, 31 ; PAPER
   INT(RND * 7) ; FLASH 1 ; "_" :NEXT n
```

Exercițiul 7.19 : Să se coloreze o zonă din ecran și apoi tot ecranul.

```
10 CLS :LET l=4 : LET s=8 :LET c=3 :LET m=
   =29 :REM l—nr. liniei de început ; s—nr. liniei
   de sfîrșit a zonei colorate ; c—nr. coloanei din
   stînga de unde începe colorarea ; m—nr. coloanei
   din dreapta unde se termină colorarea
20 FOR i=l TO s :PRINT AT i, c ; PAPER 2 ;
   TAB m :NEXT i
30 PAUSE 0 :CLS
100 LET l=0 :LET m=31 :LET s=21 :LET c=0
110 FOR i=l TO s :PRINT AT i, c ; PAPER 5 ;
   TAB m :NEXT i :REM colorarea întregului ecran
```

Exercițiul 7.20 : se cere o cortină trasă în jos sau în sus.

```
10 CLS :LET z=NOT PI :LET u=NOT z :PRINT
   AT z, z :FOR i=u TO 21 :PRINT PAPER 5 ;
   AT i, u ; TAB 31 :NEXT i
15 PAUSE 0 :CLS :REM cortină în jos
20 FOR i=21 TO u STEP —1 : PRINT PAPER 0 ;
   AT i, z+1 : TAB 0 ; NEXT i :REM cortină în sus
```

Se vor reține exercițiile *7.19* și *7.20* ca fiind utile pentru diverse programe.

7.3. PROBLEME

P7.1 : Să se elaboreze un program care redă sunetele mașinii poliției

Rezolvare :

```
9 CLS :PRINT AT 12, 3 ; "SUNET 'MASINA POLI-
  TIEI' "
10 FOR f=6 TO 4 STEP —.25 :BEPP .1, 10 :BEEP .1,
   10 :BEEP .1, 20 :NEXT f
```

P7.2: Să se redea sunetul produs la lovirea a două corpuri

Rezolvare:
```
9 CLS :PRINT AT 12, 3; "SUNET 'LOVIREA
  CORPURILOR' "
10 FOR n=0 TO 10 :BEEP 1/20, -10 :BEEP 1/20, 0 :
  NEXT n
```

P7.3: Să se imite sunetul greierilor

Rezolvare:
```
9 CLS :PRINT AT 12, 8 : "SUNET 'GREIERI' "
10 FOR c=144 TO 153 :PAUSE 1 :FOR a=0 TO 7 :
  PAUSE 1 :BEEP .001, c-100 :NEXT a :NEXT c
```

P7.4: Se cere un sunet care să ilustreze într-un program eroarea făcută de utilizator.

Rezolvare:
```
9 CLS :PRINT AT 12, 5 ; "SUNETE PENTRU 'ERORI"
10 FOR i=30 TO 0 STEP -1 : BEEP .003, i :NEXT i
```

P7.5: Să se imite sunetul telefonului.

Rezolvare:
```
9 CLS :PRINT AT 12, 6; "SUNETUL TELEFONU-
  LUI !"
10 FOR f=PI TO 2 * PI-PI/24 STEP .04 :BEEP .006,
  20 :NEXT f
```

P7.6: Să se umple ecranul cu pătrățele divers colorate

Rezolvare:
```
5 FOR i=0 TO 21
6 FOR j=0 TO 31
10 BORDER 1 ; INK RND * 7
20 PRINT AT i, j ; FLASH 1 ; "■";
30 NEXT j :NEXT i
```

P7.7: Să se realizeze o scară colorată

Rezolvare:
```
5 BORDER 0 :PAPER 5 :CLS
10 LET x=1 : FOR l=1 TO 21
20 FOR c=1 TO 6 :PRINT INK c; AT l, c+x; "■"
  :NEXT c
30 LET x=x+1 :NEXT l
```

P7.8: Se cere o mină colorată cu dungi verticale cuprinzînd cele 8 culori oferite de calculator.

Rezolvare:
```
10 CLS :FOR i=0 TO 1 :FOR k=0 TO 10 :FOR j=0
  TO 7 :PAPER j :BRIGHT i :PRINT "  ▁ ▁";
  :NEXT j :NEXT k :NEXT i :REM mira
```

```
20 FOR w=1 TO 80: BORDER 1:BORDER 3:
   BORDER 5:BORDER 7:BORDER 2:BORDER 4:
   BORDER 6:PAUSE 1:NEXT w:REM efecte pe
   BORDER
30 PRINT # 0; AT 0,4; "APASATI O TASTA
   OARECARE":BEEP .02,45:PAUSE 2:BEEP .02,
   10:PAUSE 0:CLS:STOP
```

P7.9: Să se realizeze un chenar pentru fondul ecranului

Rezolvare:

```
5 BORDER 0:PAPER 0:INK 7:CLS
10 LET p$="32 semne diez":LET q$="30 blancuri"
20 FOR n=1 TO 20:PRINT AT n,0; p$; AT n-1,1;
   q$; AT 20-n,0; p$; AT 21-n,1: q$:NEXT n
```

P7.10: Folosind caracterele grafice ale calculatorului (tastele *1*...*8*), să se deseneze un robot.

Rezolvare:

```
3 BORDER 2:PAPER 1:CLS
10 PRINT INK 2; AT 3,15; "▛▀▀▀▜" :REM în modul
   grafie tasta 8
15 PRINT INK 2; AT 4,15; "▛▀▀▀▜"
20 PRINT INK 2; AT 5,16; "▛▀▜"
30 PRINT INK 5; AT 6,13; "▛▀▀▀▀▀▀▀▜"
40 FOR l=7 TO 10; PRINT INK 5; AT l,13; "▛▀▜
   ▛▀▀▀▀▀▜ ▛▀▜" :NEXT l :REM în modul grafie
   tastele 5 și 8
45 PRINT INK 6; PAPER 0; AT 8,14; "HC-85"
50 PRINT INK 2; AT 11,13; "▛▀ ▀ ▀ ▀ ▀ ▀▜"
60 FOR l=11 TO 15:PRINT INK 6; AT l,14; "▛▀▜
   ▀ ▀ ▀ ▛▀▜" :NEXT l
70 PRINT INK 3; AT 16,13; "▛▀▀▀▜"; TAB 17;
   "▛▀▀▀▜"
72 PRINT AT 3,15; PAPER 2; INK 0; "0 0"
80 FOR l=17 TO 21:FOR c=0 TO 31:PRINT INK 4;
   AT l,c; "▛▀▜" :NEXT c:NEXT l
```

```
90 PRINT AT 19, 1 ; PAPER 6 ; FLASH 1 ; "ROBOTUL
   HC-85 ASIGURA SERVICII" ; AT 20, 1 ; "MENA-
   JERE␣ ␣ ␣ (CURATENIE, TELEFON)"
100 FOR n=0 TO 31 :PRINT AT 0, n ; PAPER INT
   (RND * 7) ; FLASH 1 ; "␣" ; AT 21, n ; PAPER
   INT(RND * 8) ; FLASH 1 ; "␣" :NEXT n
110 FOR n=0 TO 21 : PRINT PAPER INT (RND * 7) ;
   FLASH 1 ; "␣" ; AT n, 31 ; PAPER INT(RND * 7) ;
   FLASH 1 ; "␣" :NEXT n
```

Capitolul 8

INSTRUCȚIUNI GRAFICE

Partea utilizabilă a ecranului *TV* are *22* linii și *32* de coloane (fig. 8.1, a), reprezentînd $22 \times 32 = 704$ poziții de caractere (*grila de joasă rezoluție*).

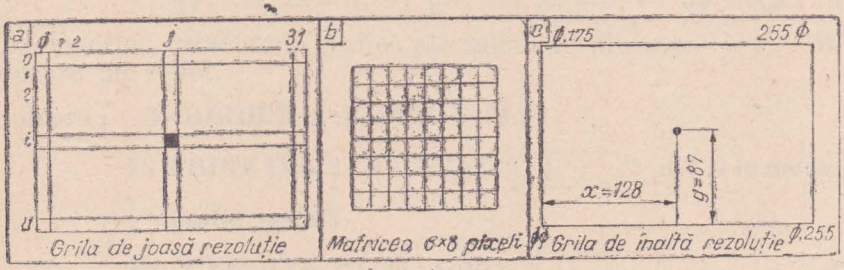


Fig.8.1 Grila ecran

La rîndul ei, fiecare poziție de caracter este un pătrat format din 8×8 puncte numite *pixeli (grila de înaltă rezoluție* — fig. 8.1, b); deci pe ecran se pot desena $704 \times 64 = 45056$ puncte (pixeli)).

Un pixel este indicat prin coordonatele sale x, y (fig. 8.1, c) unde:

x — distanța față de marginea din stînga a **BORDER**-ului;
y — distanța față de marginea de jos a **BORDER**-ului.

De exemplu, punctul de la mijlocul ecranului are coordonatele $x = 128$, $y = 87$. Rezultă că x și y trebuie să respecte condițiile: $0 \leqslant x \leqslant 255$; $0 \leqslant y \leqslant 175$.

Controlul la nivel de pixeli al ecranului este dat de instrucțiunile **PLOT, DRAW, CIRCLE** și **POINT**. Ele pot fi folosite în combinare cu instrucțiunile **PRINT, INVERSE, OVER, ATTR, FLASH, INK** și **BRIGHT**. De asemenea prezintă interes și comenzile **SCREEN$, TRUE VIDEO** și **INVERSE VIDEO**.

8.1. INSTRUCȚIUNI GRAFICE PROPRIU ZISE *(PLOT, DRAW, CIRCLE)*

a) Instrucțiunea PLOT

Sintaxa : **[nr. linie] PLOT x, y**
unde $x \in [0; 255]$ și $y \in [0; 175]$

Exemple : **10 PLOT 0, 0**
20 PLOT 255, 0
30 PLOT INK 2; 5, 113

Efect : desenează un punct de coordonate x, y

Exemplul 8.1:
```
9 REM desenarea unui punct
10 INPUT "Introdu coordonatele punctului x, y", x,y
20 IF x<0 OR x>255 AND y<0 OR y>175
   THEN GO TO 10
30 PLOT x, y
```

Dacă se dorește un punct colorat, instrucțiunea *30* se modifică introducînd culoarea dorită (de exemplu culoarea roșie : **30 PLOT INK 2 ; x, y**).

b) **Instrucțiunea DRAW**

Sintaxa : **[nr. linie] DRAW x, y [, a]**
unde $x \in [-255; 255]$, $y \in [-175; 175]$; a — lungimea arcului în radiani

Exemple :
```
10 DRAW 90, -40
20 DRAW INK 4; 50, 50
30 DRAW 8, 16, PI
```

Efect : trasează o linie dreaptă (pentru forma **DRAW x, y**) din poziția curentă a cursorului (dată de ultima instrucțiune grafică **PLOT, DRAW** sau **CIRCLE**), pînă la punctul de coordonate x, y (deci lucrează în coordonate relative la ultimul punct).
Cu instrucțiunea **DRAW x, y, a** se desenează *porțiuni de curbe*, unde x, y sînt coordonatele *punctului final*, iar a reprezintă numărul de radiani corespunzător arcului de curbă care se trasează (dacă $a > 0$ arcul se trasează în *sens trigonometric*, iar dacă $a < 0$ trasarea are loc în *sens orar*)

Observații : 1 Dacă :
$x > 0$ deplasarea se face la *dreapta*
$x < 0$ deplasarea se face la *stînga*
$y > 0$ deplasarea se face în *sus*
$y < 0$ deplasarea se face în *jos*.

2) Dacă înaintea lui **DRAW** nu a fost altă instrucțiune grafică prin care să se poziționeze cursorul, acesta se consideră inițializat pe poziția $x = 0$, $y = 0$; prin urmare instrucțiunea **DRAW** necesită precizarea întotdeauna a punctului de unde începe trasarea printr-o instrucțiune **PLOT**.

Exemplul 8.2 :

```
9 REM trasarea fondului ecranului (spațiul de lucru)
10 CLS :PLOT 0, 0 :REM poziționarea cursorului în punctul din stînga jos a ecranului — vezi fig. 8.1,
20 DRAW 255, 0 :REM trasarea dreptei orizontale inferioare
30 DRAW 0, 175 :REM trasarea dreptei verticale dreapta
40 DRAW —255, 0 :REM trasarea dreptei orizontale superioare
50 DRAW 0, —175 :REM trasarea dreptei verticale stînga
```

Exemplul 8.3 : programele următoare trasează dreapta și semicercul AB cu sensul de la A la B în conformitate cu situațiile prezentate în fig. 8.2 (axele de coordonate și liniile ajutătoare nu apar ele fiind trasate pentru înțelegere).

c) Instrucțiunea CIRCLE

Sintaxa : **[nr. linie] CIRCLE x, y, r**
unde x, y, r — numere naturale sau expresii ce dau ca rezultat numere naturale (x, y — coordonatele centrului cercului, r — raza cercului)

$$0 < x + r \leqslant 255; \quad 0 < y + r \leqslant 175; \quad r > 0$$

Exemple :

```
10 CIRCLE 130, 100, 50
20 CIRCLE INK 4; 120, 80, 70
```

Efect : desenează un cerc cu centrul în punctul de coordonate x, y și raza r

Exemplul 8.4 : programele următoare trasează cercuri în conformitate cu situațiile indicate în fig. 8.3 (axele de coordonate și liniile ajutătoare nu apar ele fiind trasate pentru înțelegere).

8.2. INSTRUCȚIUNI AJUTĂTOARE *(OVER, ATTR, POINT, INVERSE, SCREEN$, TRUE VIDEO* ȘI *INVERS VIDEO)*

a) Instrucțiunea OVER

Uneori apar situații în care este necesară modificarea unei părți dintr-o figură desenată cu instrucțiunile **PLOT, DRAW** și **CIRCLE**. Pentru asemenea situații a fost concepută instrucțiunea **OVER** care oferă posibilitatea ștergerii unui punct, unei drepte, a unui arc de cerc sau a unui cerc, după cum este folosită alături de **PLOT, DRAW** și **CIRCLE**. De asemenea, **OVER** poate fi folosită și în operații cu diferite caractere, pentru

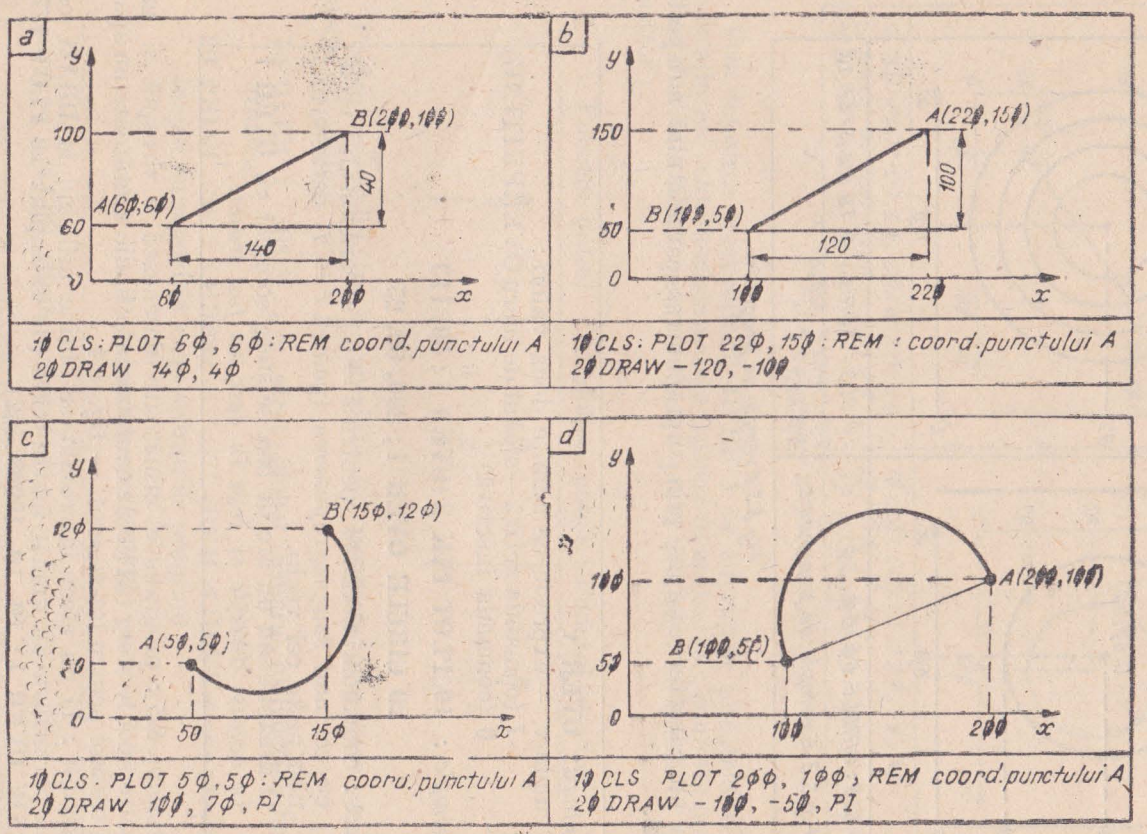


Fig 8.2 Exemple de programe

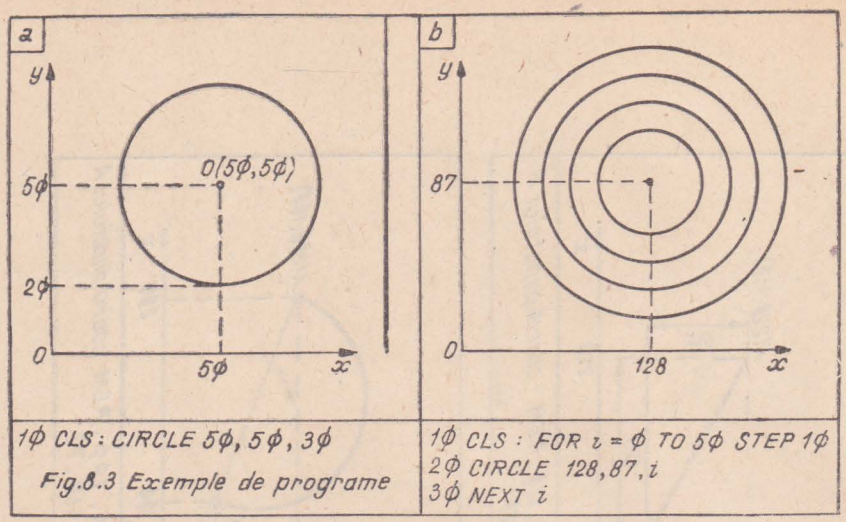


Fig.8.3 Exemple de programe

Fig.8.3 Exemple de programe

a forma un caracter compus prin suprapunerea caracterului nou peste cel vechi.

Sintaxa: **OVER v**
unde *v* — expresie ce admite două valori :
1 (comanda activă —modul de lucru OVERPRITING)
0 (comanda inactivă)

Exemple : **10 PLOT INK 2 :OVEB 1 ; 50, 113**

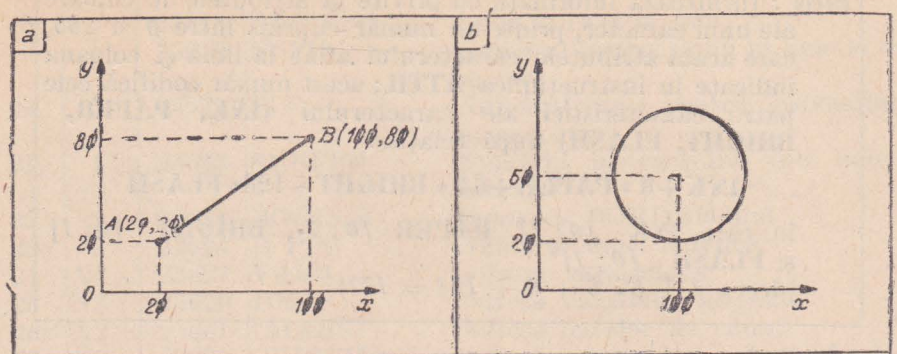
20 CIRCLE OVER 1 ; 200, 60, 25

Efect : comandă *înlocuirea* (pentru $v = 0$) a caracterului vechi cu cel nou sau *suprapunerea* (pentru $v = 1$) caracterului nou peste cel vechi.
O dreaptă sau un arc de cerc se șterg complet cu **OVER 1** *în aceeași direcție și sens în care au fost trasate.*

Exemplul 8.5 : programele următoare desenează o dreaptă și un cerc pe care apoi le șterg (axele de coordonate și liniile ajutătoare nu apar ele fiind trasate pentru înțelegere). — fig. 8.4.

Observații : 1) un efect interesant se obține utilizînd la **DRAW** ca al treilea parametru un număr foarte mare (de exemplu : **10 PLOT 177, 30 : DRAW OVER 1 ; 10, 10, 19999.562**)

2) **OVER 1** specifică faptul că se lucrează în modul *OVERPRITING*. Pentru a explica aceasta se presupune că într-un anumit loc pe ecran este tipărit un caracter ; trecînd în modul *OVERPRITING* se va tipări în același loc un alt caracter. Ambele caractere sînt formate dintr-o matrice 8×8 *pixeli* în care unele puncte (pixeli) sînt colorate în culoarea cernelii (notate *1*) iar alte puncte sînt tipărite în culoarea fondului (notate ∅), reprezentînd *„puncte albe"*.




```
10 CLS : PLOT 20,20; DRAW 80,60
20 PAUSE 0: PLOT OVER 1; 20; 20
30 DRAW OVER 1; 80,60
```

```
10 CLS : CIRCLE 100,50,30
20 PAUSE 0
30 CIRCLE OVER 1; 100,50,30
```

Fig. 8.4 Exemple de programe

Prin folosirea instrucțiunii **OVER 1** se suprapun punctele colorate și cele albe ale celor două caractere tipărite în același loc, conform operației de *SAU EXCLUSIV* pentru fiecare pixel așa cum se arată în tabelul următor :

Punctul inițial	Al doilea punct	Punctul rezultat
0	0	0
0	1	1
1	0	1
1	1	0

Pentru exemplificare se prezintă un program prin care se obține litera u cu umlaut (adică ü) :

```
10 OVER 1  :REM are efect global asupra instrucțiunilor care urmează
20 PRINT "u"; CHR$8; " " " ", :REM : CHR$ 8 comandă un salt înapoi cu o poziție a cursorului de program
```

b) **Instrucțiunea ATTR**

Sintaxa : **ATTR (linie, coloană)**

unde *linie* și *coloană* reprezintă coordonatele unui punct ce urmează a fi testat

$0 \leqslant linie \leqslant 23$; $0 \leqslant coloană \leqslant 31$

Exemplu :
```
10 PRINT AT 0, 0; INK 1; PAPER 6; BRIGHT 0; FLASH 1; "x"
20 PRINT ATTR(0, 0) : REM se vor afișa x și 177.
```

Efect : furnizează informații cu privire la atributele de culoare ale unui caracter, printr-un număr cuprins între *0* și *255*, care arată atributele caracterului aflat la linia și coloana indicate în instrucțiunea **ATTR** ; acest număr codifică cele patru caracteristici ale caracterului (**INK, PAPER, BRIGHT, FLASH**) după relația :

INK+8 * PAPER+64 * BRIGHT+128* FLASH

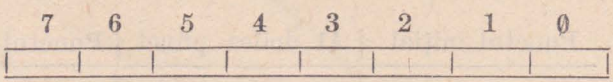
în care **INK** *[0; 7]*, **PAPER** *[0; 7]*, **BRIGHT** *[0; 1]* și **FLASH** *[0; 1]*
(deci : *1 + 8 * 6 + 0 + 128 = 177*)

Acest mod de codificare se va înțelege dacă se consideră modalitatea de calcul în baza *2*. Astfel :

— cerneala (**INK**) și hîrtia (**PAPER**) au *8* valori posibile și deci valorile lor se pot exprima prin *3* cifre binare (deoarece $2^3 = 8$) ;

— strălucirea (*BRIGHT*) și pîlpîierea (*FLASH*) pot avea *2* valori și deci se vor reprezenta pe cîte un bit.

În total rezultă *3 + 3 + 1 + 1 = 8* biți pentru un pătrățel *8 × 8 pixeli*. Acești biți sînt cifrele unui *număr în baza 2* conform schemei următoare :



Bitul *7* reprezintă **FLASH**, bitul *6* reprezintă **BRIGHT**, biții *5—4—3* indică **PAPER** și biții *2—1—0* indică **INK**.

Bitul *7* este *bitul cel mai semnificativ* (*MSB* — Most Significant Bit) denumit și *bitul superior* (notat uneori *h*), iar bitul *0* este *bitul cel mai puțin semnificativ (LSB — Less Significant Bit)* denumit și *bitul inferior* (notat uneori *l*).

Valoarea maximă pe care o poate lua acest număr oferit de **ATTR** este

$$7 + 8 \times 7 + 1 \times 64 + 1 \times 128 = 255$$

și ea corespunde unui pătrățel alb complet, strălucitor și pîlpîietor.

Se observă că **ATTR** poate testa și pătrățelele de pe cele două linii ale spațiului de editare (liniile *22* și *23*), care de obicei au culoarea **BORDER**-ului :

```
5 BORDER 4

10 PRINT#0; AT 0, 12; INK 1; PAPER 6; BRIGHT 1;
   FLASH 1; "x"

15 PRINT#0; AT 0, 15; ATTR(22, 12) :PAUSE 0 : REM se afișează
   pe linia 22 x și 177.
```

Funcția **ATTR** este utilizabilă mai ales în jocurile de divertisment.

c) Instrucțiunea POINT

Sintaxa : **POINT (x, y)**
unde x, y sînt coordonatele în pixeli ale unui punct de pe ecran ($0 \leqslant x \leqslant 255$; $0 \leqslant y \leqslant 175$)

Exemplu :
```
10 BORDER 2
20 PLOT PAPER 0; INK 7; 100, 100
30 PRINT POINT (100, 100) : PRINT POINT (100, 101) : REM se afișează un punct și cifrele 1 și respectiv 0
```

Efect : arată dacă un punct de coordonate x, y (în pixeli) de pe ecran este *aprins* (are culoarea cernelii) sau este *stins* (are culoarea fondului), prin valoarea sa care poate fi :
1 pentru punct aprins (punct în **INK**)
0 pentru punct stins (punct în **PAPER**)

d) Instrucțiunea INVERSE

Această nstrucțiune *inversează* culoarea fondului (**PAPER**) cu culoarea cernelii (**INK**) în cazul instrucțiunilor **PRINT, PRINT AT** și **PRINT TAB**. Atunci cînd se folosește împreună cu instrucțiunile grafice (**PLOT, DRAW, CIRCLE**) inversează culoarea cernelii (**INK**) cu cea a fondului (**PAPER**), efectul fiind de *ștergere*.

Sintaxa : **INVERSE v**
unde v — expresie ce admite două valori :
1 pentru comandă activă (invers video)
0 pentru comandă inactivă

Exemplu :
```
10 PLOT 0, 0 :DRAW 255, 175 :PAUSE 0 :REM diagonala ecranului
20 DRAW INVERSE 1, -255, -175 :REM se șterge diagonala ; dacă se înlocuiește INVERSE 0 ștergerea nu are loc
```

Efect : produce stingerea punctelor aprinse (active) de pe ecran

Observație : instrucțiunea **INVERSE** poate fi utilizată și ca linie-program caz în care va avea efect global supra instrucțiunilor care urmează, spre deosebire de folosirea sa în cadrul instrucțiunilor **PRINT, PRINT AT** și **PRINT TAB** unde are efect local.

e) Instrucțiunea/comandă SCREEN$

Sintaxa : a) **SCREEN$ (linie, coloană)**

Exemplu :
```
10 PRINT SCREEN$(4, 3)
```

Efect : întoarce caracterul tipărit pe ecran sub formă normală, la linia și coloana indicate, sau șirul nul dacă acest caracter nu este recunoscut; instrucțiunea nu recunoaște decît caracterele între *CHR$(32)* = blancul și *CHR$(127)* = ©. De pildă programul

```
10 PRINT AT 15, 5; "M.M.P" :PRINT SCREEN$
(15,5) :PRINT SCREEN$(15, 6)
```

va tipări

```
M.M.P.
M
.
```

b) **SAVE „nume" SCREEN$**

Exemplu : SAVE „Rulmenti" SCREEN$

Efect : salvează pe banda magnetică conținutul memoriei ecran (ce se vede pe ecran), inclusiv atributele; în acest fel se realizează „mirele" (genericele) programelor.

Exemplul 8.6 :

```
9 REM dreaptă orizontală trasată instantaneu
10 PLOT 0, 8 :DRAW 255, 0
```

Exemplul 8.7 :

```
9 REM dreaptă orizontală trasată cursiv
10 PLOT 0, 60 :FOR x=0 TO 255 :PLOT x, 60 :
NEXT x
```

Exemplul 8.8 :

```
9 REM orizontală trasată punctat
10 PLOT 0, 80 :FOR x=0 TO 255 STEP 3 :PLOT x,
80 :NEXT x
```

Exemplul 8.9 :

```
9 REM orizontală hașurată
10 PLOT 10, 40 :DRAW 226, 0 :REM trasarea
dreptei
20 FOR x=10 TO 236 STEP 5 :PLOT x, 40 :DRAW
−2, −2 :NEXT x :REM trasarea hașurilor
```

Exemplul 8.10 :

```
9 REM oritontală de o anumită grosime
10 For i=0 TO 255 :PLOT i, 70 :DRAW 0, 10 :
NEXT i :REM grosime 10 pixeli
```

Exemplul 8.11 :

```
9 REM axe de coordonate
10 PLOT 125, 5 :DRAW 0, 170 :PLOT 125, 175 :
DRAW −2, −2 :DRAW 4, 0 :PLOT 0, 85 :
DRAW 255, 0 :PLOT 255, 85 :DRAW −2, −2 :
PLOT 255, 85 :DRAW −2, −2 :PRINT AT 10,
31; "OVER 1;"x"; AT 0, 16; OVER 1;"y";
AT 12, 14; "0"
```

Exemplul 8.12 :

```
9 REM segmente orizontale îngroșate
10 FOR n=0 TO 11 :PLOT 0, 139-n :DRAW 111, 0 :PLOT 255, 139-n :DRAW -111, 0 :PLOT 255, 84+n :DRAW -11, 0 :PLOT 0, 84+n :DRAW 11, 0 :NEXT n
```

Exemplul 8.13 :

```
9 REM 4 segmente înclinate îngroșate
10 FOR n=0 TO 19 :PLOT 185+2*n, 140+n : DRAW 32, 0 :PLOT 200+2*n, 83-n :DRAW 16, 0 :PLOT 39-2*n, 83-n :DRAW 32, 0 : PLOT 54-2*n, 140+n :DRAW -16, 0 :NEXT n
```

Exemplul 8.14 :

```
9 REM cercuri cu raze crescătoare
10 FOR i=50 TO 200 STEP 10 :CIRCLE i, 88, i/5+10 :NEXT i
```

Exemplul 8.15 :

```
9 REM cornet
10 LET x=-1 :FOR j=15 TO 70 STEP 2 :PLOT 128, 85 :DRAW j, j, ((2*PI)*x)-PI :PAUSE 25 : LET x=-x :NEXT x
```

Exemplul 8.16 :

```
9 REM șir de romburi pe orizontală
10 PLOT 0, 100
15 FOR i=1 TO 16 :DRAW 8, 8 :DRAW 8, -8 : DRAW -8, -8 :DRAW -8, 8 :DRAW 15, 0
20 NEXT i
```

Exemplul arată că avantajul trasării în coordonate relative și anume faptul că modificînd doar coordonatele punctului de pornire se schimbă locul figurii pe ecran. Dacă la acest program se adaugă

```
9 FOR n=1 TO 4
10 PLOT 0, 100+n
.
.
25 NEXT n
```

liniile desenului se îngroașe rezultînd o figură geometrică interesantă.

● Pot fi reprezentate diverse curbe matematice folosind ecuațiile lor parametrice. Cu programul următor se reprezintă *ovoida*, *cardioida*, *nefroida*, *cuartica piriformă*, *astroida*, *deltoida*, *melcul lui Pascal*.

Exemplul 8.17.:

```
3 BORDER 2 :PAPER 0 :INK 7 :CLS
4 LET c=1000 :LET p=2000
5 PRINT AT 12, 2; "_ _ _ REPREZENTAREA GRAFICA A _ _ _ _ _ _ UNOR CURBE MATEMATICE FOLO-_ _ _ _ _ SIND ECUATIILE PARAMETRICE"
```

```
 8 PAUSE 50 :CLS
10 CLS :PRINT AT 12, 13; "OVOIDA" :PAUSE 50 :CLS
11 GOSUB c
15 FOR a=0 TO PI STEP PI/200 :LET x=2 * COSa * COSa * COSa * COSa :LET y=2 * COSa * COSa * COSa * SINa :PLOT 50+75 * x, 85+75 * y :NEXT a
16 PRINT AT 1, 1; "OVOIDA"; AT 2, 1; "x=k(COSc) ↑ 4"; AT 3, 1; OVER 0; "y=k(COSa) ↑ 3 * SINa"; AT 4, 1; "k>0"
17 BEEP .02, 45 :PAUSE 2 :BEEP .02, 10
18 GOSUB p
20 PRINT AT 12, 11; "CADIOIDA" :PAUSE 50 : CLS
21 GOSUB c
25 FOR a=0 TO 2 * PI STEP PI/200 :LET x=3 * (1+COSa) * COSa : LET y=3 * (1+COSa) * SINa : PLOT 125+18 * x, 85+18 * y : NEXT a
26 PRINT AT 1, 1; "CARDIOIDA"; AT 2, 1; OVER 0; "x=k(1+ COSa)COSa"; AT 3, 1; "y=k(1+COSa)SINa"; AT 4, 1; "k>0"
27 BEEP .02, 45 :PAUSE 2 :BEEP .02, 10
28 GOSUB p
30 PRINT AT 12, 11; "NEFROIDA" : PAUSE 50 :CLS
31 GOSUB c
35 FOR a=-PI TO PI STEP PI/100 :LET x=2 * ((3 * COSa)-SIN (3 * a)) :LET y=2 * ((3 * COSa)-COS(3 * a)) :PLOT 125+10 * x, 85+10 * y :NEXT a
36 PRINT AT 1, 1; "NEFROIDA"; AT 2, 1; OVER 0;"x=k(3SINa) -SIN3a"; AT 3, 1; OVER 0; "y=k(3COSa)-COS3a"; AT 4, 1; "k>0"
38 GOSUB p
40 PRINT AT 12, 7; "CUARTICA PIRIFORMA" :PAUSE 50 :CLS
41 GOSUB c
45 FOR a=0 TO PI STEP PI/200 :LET x=6 * COSa * COSa :LET y=10 * COSa * COSa * COSa * SINa : PLOT 125+20 * x, 85+30 * y :NEXT a
46 PRINT AT 1, 1; "CUARTICA" AT 2, 1; OVER 0; "x=k(COSa) ↑ 4"; AT 3, 1; OVER 0; "y=l(COSa) ↑ 3-SINa"; AT 4, 1; "k, l>0"
48 GOSUB p
50 PRINT wT 12, 12: "ASTROIDA" :PAUSE 50 :CLS
51 GOSUB c
```

```
55 FOR a=0 TO 2*PI STEP PI/200:LET x=5*COSa*COSa*
   COSa:LET y=5*SINa*SINa*SINa:PLOT 125+16*x, 85+
   16*y:NEXT y

56 PRINT AT 1, 1; "ASTROIDA"; AT 2, 1; OVER 0; "x=y(COSa) ↑
   3"; AT 2, 1; OVER 0; "y=k(SINa) ↑ 3"; AT 4, 1; "k>0"

58 GOSUB p

60 PRINT AT 12, 12; "DELTOIDA":PAUSE 50:CLS

61 GOSUB c

65 FOR a=0 TO 2*PI STEP PI/100:LETx=30*(2*COS(2*a)):
   LET y=30*(2*SINa-SIN(2*a)):PLOT 125+x, 85+y:NEXT a

66 PRINT AT 1, 1; "DELTOIDA"; AT 2, 1; OVER 0; "x=k(2COSa+
   COS2a)"; AT 3, 1; OVER 0; "y=k(2SINa-SIN2a)"; AT 4, 1;
   "k>0"

68 GOSUB p

70 PRINT AT 12, 7; "MELCUL LUI PASCAL":PAUSE 50:CLS

71 GOSUB c

75 FOR a=0 TO 2*PI STEP PI/200:LET x=(10*COSa+2)*COSa:
   LET y=(10*COSa+2)*SINa:PLOT 125+9*x, 85+9*y:
   NEXT a

76 PRINT AT 1, 1; OVER 0; "MELCUL LUI PASCAL"; AT 2, 1;
   OVER 0; "x=(kCOSa+2)COSa"; AT 3, 1; OVER 0; "y=
   (kCOSa+2)SINa"; AT 4, 1; "k>0"

78 GOSUB p

80 PRINT AT 12, 4; "RELUATI PROGRAMUL (d/n)?":PAUSE 0

85 GO TO 10*(INKEY$="d")+100*(INKEY$="n")

100 CLS:STOP

1000 PLOT 0, 0:DRAW 255, 0:DRAW 0, 175:DRAW -255, 0:DRAW
     0, -175:PLOT 125. 0:DRAW 0, 175:PLOT 125, 175:DRAW -2,
     -2:DRAW 4, 0:PLOT 0, 85:DRAW 255, 0:OVER 1; "y"; AT
     12, 16; "0":RETURN: REM axele de coordonate

2000 PRINT AT 20, 18; "APASA O TASTA":PAUSE 0:CLS:RETURN
```

f) Comenzile TRUE VIDEO și INVERSE VIDEO

Aceste comenzi permit *scriere de tip contrast* și anume:

— scriere *negru pe alb* cu comanda **TRUE VIDEO**, care se obține tastînd simultan *CS* și *3*;

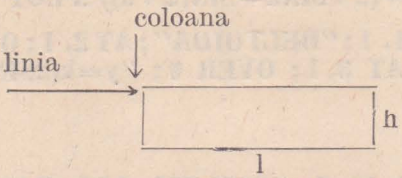
— scriere *alb pe negru* cu comanda **INVERSE VIDEO**, care se obține tastînd simultan *CS* și *4*.

Prin folosirea lor pentru scrierea unui text în care o literă se tipărește în **TRUE VIDEO** iar litera următoare în **INVERSE VIDEO** se obțin *efecte de scriere* foarte plăcute vederii.

Exemplul 8.18 :

```
10 BORDER 0:PAPER 0:INK 7::CLS
20 FOR k=19 TO 21:PRINT INK k-1; AT K, 4;
FLASH 1; BRIGHT 1; "23 semne ■ ":NEXT
k:PRINT AT 18, 4; "APASATI O TASTA
OARECARE": PAUSE 0: REM literele din text
se tastează succesiv în TRUE VIDEO, respectiv
în INVERSE VIDEO
```

Programele atractive au în interiorul lor incluse „*ferestre*" colorate în care se scriu elemente importante ale programului (definiții, formule de calcul, titluri, citate, etc.). Programul următor realizează acest deziderat folosind variabila șir (string) *a$* = „*10 cifre*" distribuite conform schemei :



a$ = "␣␣ ␣␣ ␣␣ ␣␣ ␣␣ ␣␣ ␣␣ ␣␣ ␣␣ ␣␣"

linia coloana INK PAPER lungimea l înălțimea h

exemplu : *a$* = "*0902222706*" — linia *9*, coloana *2*, **INK 2, PAPER 2,** *l* = *27*, *h* = *6*

unde *l* este lungimea ferestrei și *h* înălțimea ei (în caractere)

Exemplul 8.19 :

```
1 REM ferestre colorate
2 PRINT AT 2, 1; "Programul realizează ferestre
␣␣␣ colorate în zone diferite pe TV"
3 PLOT 0, 0:DRAW 255, 0:DRAW 0, 175:DRAW
-255, 0:DRAW 0, -175:PLOT 17, 56:DRAW
213, 0:DRAW 0, 46:DRAW -213, 0:DRAW
0, -46
5 PRINT AT 20, 4; "APASATI O TASTA OARE-
CARE"
9 PAUSE 0
10 LET bx=9900:LET a$="0902222706":GOSUB
bx
11 PLOT 17, 56:DRAW 213, 0:DRAW 0,46:
DRAW -213, 0:DRAW 0, -46
12 STOP
9900 LET x=VAL a$(3 TO 4)
9905 PRINT AT VAL a$(TO 2), x;
9910 FOR i=1 TO VAL a$(9 TO):PRINT OVER 1;
BRIGHT 8; FLASH 8; PAPER 8; INK 8;
TAB x; PAPER VAL a$(6); INK VAL a$(5);
TAB x+VAL a$(7 TO 8)-1; "_":NEXT i:
RETURN
```

Exemplul 8.20 :

```
 4 REM graficul funcției de o singură variabilă
 5 BORDER 6; PAPER 6 :INK 1 :CLS
10 INPUT "Introduceți funcția f(x) = "; q$ :IF LEN
   q$=0 THEN GO TO 10
12 INPUT "Domeniul de reprezentare : limita
   stîngă :"; ls ; ",limita dreaptă :"; ld ; "␣" : IF
   ls >=ld THEN GO TO 12
13 PRINT AT 1, 2 ; "Domeniul f :[" ; ls; "," ; ld ;" ]→
   R"; AT 0, 2; PAPER 7; INK 0; "Funcția
   f(x) = "; q$
14 PRINT AT 20, 9; "CORECT (d/n) ?" :PAUSE 0
15 GOTO 10*(INKEY$="n")+30*(INKEY$="d")
30 CLS :LET rap=255/(ld+ls)
40 LET orig=-ls*rap
50 BEEP .02, 45 : PAUSE 2 :BEEP .02, 10
60 REM axele de coordonate
70 PLOT 0, 87 : DRAW 255, 0 : DRAW -4, 2 :
   DRAW 0, -4 :DRAW 3, 2
80 PRxNT OVER 1 ; AT 9, 0 ; ls ; AT 9, 25 ; ld :
   PRINT AT 10, 0 :"␣ ␣"
90 IF ls >0 OR ld<0 THEN GO TO 120
100 PLOT orig, 0 :DRAW 0, 170
120 LET fin=1 ; LET cv=1
140 LET ym=87/rap/cv : LET ym=INT(ym*100)/
   100
150 PRINT AT 0, 0 ; ym ; AT 21, 0 ; -ym
199 REM trasarea graficului
200 FOR x=ls TO ld STEP fin/rap
205 LET xr=x * rap+orig :REM coordonata reală
210 LET y=VAL q$*rap*cv+87
220 IF y<0 OR y>175 THEN GO TO 240
230 PLOT xr, y
240 NEXT x
250 PRINT # 1 ; AT 0, 7 ; "ALTA FUNCTIE (d/n) ?" :
   PAUSE 0
260 GOTO 10*(INKEY$="d")+300*(INKEY$
   ="n")
300 CLS :STOP
```

În exemplele următoare sînt realizate o serie de desene cromatice

Exemplul 8.21 :

```
10 BORDER 1 :PAPER 3 :INK 6 :CLS
20 FOR i=0 TO 255 STEP 3 :PLOT 0, 0 :DRAW i,
   175 :NEXT i
30 FOR i=0 TO 255 STEP 3 :PLOT 255, 175 :
   DRAW i-255, -175
40 NEXT i
```

Exemplul 8.22 :

```
10 BORDER 2 :PAPER 6 :INK 1 :CLS
20 LET p=70 :LET q=p :LET x=126 :LET y=160
22 PLOT x, y :DRAW -p, -q :DRAW p, -q :
   DRAW p, q :DRAW -p, q
25 LET p=p-4 :IF p<=-120 THEN PRINT # 1 ;
   AT 0, 4; FLASH 1; "APASATI O TASTA
   OARECARE" : PAUSE 0 :GO TO 30
27 GO TO 22
30 CLS :PAPER 0 :INK 7 :CLS :PLOT 0, 0 :DRAW 0,
   175 :DRAW 255, 0 :DRAW 0, -175 :DRAW
   -255, 0
32 FOR i=1 TO 120 STEP 5 :PLOT 0, i :DRAW
   255-i, -i :PLOT i, 0 :DRAW 255-i, i :PLOT 0,
   i-175 :DRAW 255-i, i : PLOT i, 175 :DRAW
   255-i, -i :NEXT i
```

Exemplul 8.23 :

```
10 BORDER NOT PI :PAPER NOT PI :INK 7 :CLS
20 LET s=120
30 PLOT 0, 0 :DRAW 255, 0 :DRAW 0, 175 :DRAW
   -255, 0 :DRAW 0, -175
40 FOR i=0 TO PI STEP PI/32 :PLOT 128, 0 :
   DRAW INK 3; INT(s*COSi), INT(s*SINi) :
   NEXT i
```

Exemplul 8.24 :

```
10 BORDER SGN PI : PAPER SGN PI : INK 7 :
   CLS :BRIGHT 1 :FLASH 1 :CLS
20 FOR i=0 TO 6.25 STEP .05 :PLOT 128, 0 :DRAW
   OVER 1 ; 127*COS(i), 175*SIN(i/4) :NEXT i :
   FLASH 0
```

Exemplul 8.25 :

```
10 BORDER VAL "1" :PAPER VAL "1" :INK
   VAL "7" :CLS
20 FOR n=VAL "1" TO VAL "176" STEP VAL "2" :
   PLOT n, n : DRAW VAL "255"-(n*2), VAL
   "0" :DRAW VAL "0", VAL "175"-(n*2) :
   DRAW VAL "-255"+(n*2), VAL "0" :DRAW
   VAL "0", VAL "-175"+(n*2) :NEXT n :FOR
```

```
n=VAL "1" TO VAL "40" :BORDER VAL "0" :
BORDER VAL "6" :BORDER VAL "2" :BOR-
DER VAL "7" :BORDER VAL "3" :BORDER
VAL "5" :BORDER VAL "0" :BEEP .05, n :
NEXT n
```

Exemplul 8.26 :

```
10 BORDER 0 :PAPER 0 :INK 7 :CLS
20 FOR i=0 TO 175 STEP 14/3 :PLOT 127, 0 :
DRAW -127, i :PLOT 128, 0 :DRAW 127, i :
NEXT i
30 FOR i=2 TO 127 STEP 11/3 :PLOT 127, 0 :
DRAW i-127, 175 :PLOT 128, 0 :DRAW 127-i,
175 :NEXT i
```

Exemplul 8.27 :

```
10 BORDER 0 :PAPER 0 :INK 7 :CLS
20 FOR f=40 TO 30 STEP -2 :PLOT f-15, f-30 :
DRAW 293-2*f, 0 :DRAW 0, 150 : DRAW
-293+2*f, 0 : DRAW 0, -150 :NEXT f
```

Exemplul 8.28 :

```
10 BORDER 0 :PAPER 0 :INK 7 :CLS
20 PLOT 0, 0 :DRAW 255. 0 :DRAW 0, 175 :DRAW
-255, 0 :DRAW 0, -175
30 FOR t=0 TO 90 :LET a=t/45*PI :LET ax=85*
SIN(a) :LET ay=85*COS(a) :PLOT 128, 88 :
DRAW ax, ay :NEXT t
```

Exemplul 8.29 :

```
1  BORDER 2 :PAPER 0 :INK 7 :CLS
20 FOR j=0 TO 359 :LET x=j*PI/180 :PLOT 127,
87 :DRAW SINx*125, COSx*85 :NEXT j
```

Exemplul 8.30 :

```
10 BORDER 0 :PAPER 0 :CLS
40 LET t=0 :CLS
45 FOR j=0 TO 6
50 LET q=INT(RND*70)+100
60 LET w=INT(RND*60)
70 LET e=INT(RND*3)-4
80 FOR s=q TO w STEP e
85 IF INKEY$=CHR$(13) THEN CLS : STOP :
REM programul se opreşte tastînd ENTER
90 PLOT s, s
100 DRAW 0, 175-2*s
115 DRAW 255-2*s, 0
120 DRAW 0, -175+2*s
130 DRAW -255+2*s, 0
```

```
140 INK RND*8
150 OVER 1
160 NEXT s
170 LET t=t+1
180 IF t=5 THEN OVER 0
185 NEXT j
190 GO TO 40
```

Acest exemplu poate fi reținut pentru frumusețea desenelor aleatoare pe care programul le realizează.

● De regulă un program bine realizat începe cu un *„generic"*, respectiv cu o imagine sugestivă prin intermediul căreia se face introducerea în tematica tratată. Fără îndoială se pot imagina generice foarte diverse, în funcție de fantezia programatorului; programele care urmează oferă cîteva soluții.

Exemplul 8.31 :

```
10 BORDER 0 :PAPER 0 :INK 7 :CLS
20 FOR f=0 TO 21 : PRINT AT f, 0 :INVERSE 1 ;
   "_ _ _ _ _ _ _ PROGRAME INFORMATICE
   _ _ _ _ _ _ _ ": BEEP .004, f :PRINT AT f,
   0 ; "32 blancuri" :NEXT f
30 PRINT AT 21, 0 ; INVERSE 1 ; "_ _ _ _ _ _ _
   PROGRAME INFORMATICE _ _ _ _ _ _ _ ";
   AT 10, 5/ ; "de dr. ing. M. M. POPOVICI"
40 FOR f=0 TO 15 :BEEP .01, 10 :NEXT f
50 FOR f=21 TO 0 STEP -1 :PRINT AT f, 0 :
   INVERSE 1 ; "_ _ _ _ _ _ _ PROGRAME IN-
   FORMATICE _ _ _ _ _ _ _ "; BEEP .004, f :
   "32 blancuri"; NEXT f
60 PLOT 32, 64 : DRAW 24*8, 0, 24 : DRAW
   -24*8, 0 :DRAW 0, -24 :PRINT AT 12, 6 ;
   INVERSE 1 ; FLASH 1 ; "PROGRAME INFOR-
   MATICE" : REM textul se scrie alternînd o literă
   în TRUE VIDEO cu alta în INVERSE VIDEO
70 FOR f=0 TO 15 :BEEP .01, 15 :NEXT f
```

Exemplul 8.32 :

```
10 BORDER 1 :PAPER 1 :INK 7 :CLS :PLOT 0, 0 :
   DRAW 255, 0 :DRAW 0, 175 :DRAW -255, 0 :
   DRAW 0, -175
20 PRINT AT 1, 4 ; PAPER 5 ; INK 2 ; FLASH 1 ;
   "M. M. POPOVICI SOFTWARE ©"; AT 2, 8 ;
   PAPER 3 ; INK 7 ; FLASH 0 ; "INITIERE IN
   BASIC"
30 FOR f=0 TO 120 STEP 4 :PLOT 10+f/3, 10-
   f/3 :DRAW f, 0 :DRAW 0, f :DRAW -f, 0 :
   DRAW 0, -f :NEXT f
```

Exemplul 8.33 :

```
10 CLS :FOR f=0 TO 100 STEP 4:PLOT 10+f/2,
10+f/2 :DRAW INK 5; f, 0: DRAW INK 5, 0,
f :DRAW INK 5; -f, 0 :DRAW INK 5; 0,
-f :NEXT f

20 FOR f=100 TO 0 STEP -4 :PLOT 60-(f+f/2),
60-(f+f/2) :DRAW INK 1-(f+f/2), 60-(f+
f/2) :DRAW INK 1; (100-f), 0 :DRAW INK 1;
0, (100-f) :DRAW INK 1; -(100-f), 0 :DRAW
INK 1; 0, -(100-f) :NEXT f

30 OVER 0 :PLOT 0, 0 :DRAW 255, 0 :DRAW 0,
175 :DRAW -255, 0 :DRAW 0, -175

40 PRINT AT 8, 24; FLASH 1; "JOCURI"; FLASH
0; AT 01, 24; "create"; AT 12, 26; "de"; AT
15, 25; "elevi"
```

Exemplul 8.34 :

```
10 BORDER 0 :PAPER 0 :INK 7 :CLS

20 FOR n=1 TO 67 STEP 2 :PLOT 128-n, 88-n :
DRAW 0, n*2 :DRAW n*2, 0 :DRAW 0, -n*2 :
DRAW -n*2, 0 :NEXT n

30 PRINT AT 2, 2: "C"; AT 3, 2; "E"; AT 4, 2;
"R"; AT 5, 2; "C"; AT 7, 2; "D"; AT 8, 2;
"E"; AT 10, 2; "B"; AT 11, 2; "A"; AT 12,
2; "S"; AT 13, 2; "I"; AT 14, 2; "C"; AT 16,
2; "1"; AT 17, 2; "9"; AT 18, 2; "9"; AT 19,
2; "2"

40 PRINT AT 2, 30; "P"; AT 3, 10; "R"; AT 4,
30"; "O"; AT 5, 30; "G"; AT 6, 30; "R"; AT
7, 30; "A"; AT 8, 30; "M"; AT 10, 30; "I",
AT 11, 10; "N"; AT 12, 30; "F"; AT 13, 30;
"O"; AT 14, 30; "R"; AT 15, 30; "M"; AT
16, 30; "A"; AT 17, 30; "T"; AT 18, 30; "I";
AT 19, 40; "C" : REM o literă alternează cu alta
în scriere TRUE VIDEO, respectiv INVERSE
VIDEO

50 FOR a=1 TO 10 :BORDER 7 :FOR x=1 TO
10 :BORDER 0 :PAPER 1 :BORDER 1 :PAPER
6 :NEXT x :BORDER 1 :PAPER 6 :INK 9 :
NEXT a
```

Exemplul 8.35 :

```
10 CLS :LET c=0 :BORDER 6 :PAPER 6 :CLS :
GOSUB 100

82 FOR i=5 TO 28 :PRINT AT 3, i : "□" : NEXT i

84 FOR i=5 TO 10 : PRINT AT i, 6 ; "□" : NEXT i :
PRINT AT 7, 7; "□"; AT 8, 8; "□"; AT 7, 9;
"□" :FOR i=5 TO 10 :PRINT AT i, 10; "□" :
NEXT i :PRINT AT 10, 12; "□"

86 PRINT AT 5, 15; "□□□" :FOR i=6 TO 9 :
PRINT AT i, 14; "□" :NEXT i :PRINT AT 10,
15; "□□□" :FOR i=6 TO 9 : PRINT AT i, 18;
"□" :NEXT i :PRINT AT 10, 20; "□"
```

```
88 FOR i=5 TO 10 :PRINT AT i, 22; "□" :NEXT
   i :PRINT AT 7, 23; "□"; AT 8, 24; "□"; AT
   7, 25; "□" :FOR i=5 TO 10 :PRINT AT i,
   26; "□" :NEXT i :PRINT AT 10, 28; "□"
   :FOR i=5 TO 28 :PRINT AT 12, i; "□" :NEXT i

90 LET c=c+1 :IF c=8 THEN LET c=d

92 IF c=6 THEN GO TO 90

94 INK c :RETURN

100 INK 0 :PRINT AT 16, 7; PAPER 1; INK 7;
    FLASH 1; "©M. M. POPOVICI_ _1992";
    FLASH 0; AT 18, 7; PAPER 7; INK 0; "Colecția de programe"

102 IF INKEY$ < >"" THEN GOSUB 82 : GO TO
    102

104 IF INKEY$="" THEN GOSUB 82 :GO TO 104

106 REM se folosește caracterul grafic □ de pe
    tasta 8
```

8.3. PROBLEME

P8.1 : Să se realizeze un generic în formă de panou dreptunghiular la care se vede umbra lui.

Rezolvare :

```
10 CLS :FOR f=6 TO 16 :PRINT PAPER 5; INK 2;
   AT f, 6; "21 blanc" :NEXT f :REM panoul dreptunghiular

20 FOR f=7 TO 17 :PRINT PAPER 1; INK 1; AT f, 5;
   "_" :NEXT f: PRINT PAPER 1; INK 1; AT 17,
   6; "20 blanc" :REM umbra panoului

30 PRINT PAPER 5; INK 2; AT 7, 6; "M. M. POPOVICI SOFTWARE"; AT 19, 14; "©" 1992"; AT 13,
   9; FLASH 1; "NU OPRITI BANDA"
```

P8.2 : Se cere mira de control a televizorului calculatorului, folosind instrucțiunile de desenat și caracterul grafic de pe tasta *8*.

Rezolvare :

```
10 BORDER 0 :PAPER 0 :INK 7 :CLS

20 PRINT"= = = = =MIRA= = = = ="

30 FOR i=0 TO 255 STEP 32 :PLOT i, 0 :DRAW 0,
   159 : NEXT i

40 FOR i=0 TO 172 STEP 32 :PLOT 0, i :DRAW 224,
   0 :NEXT i

50 CIRCLE 112, 80, 80

60 FOR i=2 TO 5 :PRINT AT i, 0; INK 7; "□□□□"
   :NEXT i
```

```
70 FOR i=18 TO 21: PRINT AT i, 0; INK 6;
"□□□□□":NEXT i

80 FOR i=18 TO 21:PRINT AT i, 24; INK 5;
"□□□□□":NEXT i

90 FOR i=2 TO 5:PRINT AT i,24; INK 4;"□□□□□":
NEXT i

100 FOR i=6 TO 9:PRINT AT i, 6; : FOR f=0 TO 7:
PRINT INK f; "□□":NEXT f:NEXT i:REM modul
grafic tasta 8

110 FOR i=14 TO 17:PRINT AT i, 6; : FOR j=7 TO 0
STEP-1:PRINT BRIGHT 1; INK j; "□□"; :NEXT
i:REM tasta 8

120 PAPER 7;PRINT INK 0; AT 10, 6; "_↓ " "#$%
&'( )*+, -./"; INK 2; AT 11, 6; "0123456789:;
<=>?";INK 1;AT 12, 6;"@ABCDEFGHIJKL
MNO";INK 3; AT 13, 6;"[\] ↑ £ abcdefghij"
```

P8.3: Se cere un program care desenează o tablă școlară

Rezolvare:

```
20 BORDER 2:PAPER 2:INK 7:CLS

30 FOR n=2 TO 14:PRINT PAPER 0; INK 7; AT n,
8; "12blanc":NEXT n

40 FOR n=15 TO 17:PRINT INK 4; AT n, 10; "□";
AT n, 17; "□":NEXT n

45 FOR n=18 TO 20:PRINT INK 4; AT n, 10; "□";
AT n, 17; "□":NEXT n

50 PRINT INK 4; AT 17, 11; "□□□□□□"

60 OVER 1:INK 4:PRINT AT 15, 10; ".": AT 16, 10;
".": AT 17, 10; ".": AT 18, 10; ".": AT 19, 10; "."

70 PRINT AT 15, 17; ".": AT 16, 17; ".": AT 17,
17; ".": AT 18, 17; ".": AT 19, 17; "."
```

Alta variantă:

```
20 BORDER 2:PAPER 2:INK 7:CLS

30 FOR n=2 TO 14:PRINT PAPER 0; INK 7; AT n, 2:
"28blanc":NEXT n

40 FOR n=0 TO 5:PRINT INK 4; AT b+15, 6; "□";
AT n+15, 25; "□":NEXT n

50 PRINT AT 17, 7; INK 4; "18 caractere □"
```

P8.4: Să se deseneze un contur al ecranului având grosimea de un caracter, în interiorul căruia să se afle un desen gen *„foaie de matematică"* în care să se scrie, cu ajutorul caracterului grafic de pe tasta 8, titlul *„GRAFIC"*

Rezolvare : 5 BORDER 7 :PAPER 7 : INK 1 :CLS

10 FOR m=0 TO 31 :PRINT AT 0, m ; "□" : NEXT m :FOR n=1 TO 20 :PRINT AT n, 31 ; "□" :NEXT n :FOR m=31 TO 0 STEP −1 : PRINT AT 21, m ; "□" :NEXT m :FOR n=20 TO 1 STEP −1 :PRINT AT n, 0 ; "□" :NEXT n :REM conturul ecranului

15 FOR m=7 TO 247 STEP 8 :PLOT INK 0 ; m, 7 : DRAW INK 0; 0, 160 :NEXT m :FOR n=167 TO 7 STEP −8 :PLOT INK 0; 247, n :DRAW INK 0; −240, 0 :NEXT n :REM caroiaj în formă de "foaie de matematică"

20 PRINT INK 2; AT 3, 7 ; "□□□□□□" ; AT 4, 6; "□" ; AT 5, 5; "□" ; AT 6, 4; "□" ; AT 7, 3; "□" ; AT 8, 3; "□" ; AT 9, 3: "□" ; AT 10, 3; "□" ; AT 11, 3 ; "□" ; AT 11, 9 ; "□□□□□□□□" ; AT 12, 3 ; "□" ; AT 12, 2 ; "□□" ; AT 13, 3 ; "□" ; AT 13, 12 ; "□" ; AT 14, 3 ; "□" ; AT 14, 12 ; "□□" ; AT 15, 3 ; "□" ; AT 15, 12 ; "□□" ; AT 16, 4 ; "□" ; AT 16, 11 ; "□□" ; AT 17, 5 ; "□" ; AT 17, 10 ; "□□" ; AT 18, 6 ; "□□□"

30 PRINT AT 5, 10 ; "□□□" ; AT 6, 10 ; "□" ; AT 6, 13 ; "□" ; AT 7, 10, "□□□" ; AT 8, 10 ; "□" ; AT 8, 13 ; i'□"

40 PRINT AT 5, 15 ; "□□□" ; AT 6, 15 ; "□ □" ; AT 7, 15 ; "□□□" ; AT 8, 15 : "□ □"

50 PRINT AT 5, 20 ; "□□□" ; AT 6, 20 ; "□" ; AT 7, 20 ; "□□□" ; AT 8, 20 ; "□"

60 PRINT AT 5, 25 ; "□" ; AT 6, 25 ; "□" ; AT 7, 25 ; "□" ; AT 8, 25 ; "□"

70 PRINT AT 5, 27 ; "□□□" ; AT 6, 27 ; "□" ; AT 7, 27 ; "□" ; AT 8, 27 ; "□□□"

P8.5 : Să se deseneze un telefon

Rezolvare : 1 BORDER 7 :PAPER 7 :INK 0 :CLS

10 PRINT AT 1, 8 ; "16 caractere □" ; AT 2, 7 ; "18 caractere □"

15 PRINT AT 3, 6 ; "□□□" ; AT 3, 23 ; "□□□"

20 PRINT AT 4, 6 ; "□□□□□" ; AT 4, 11 ; INK 4 ; "□" ; AT 4, 20 ; INK 4 : "□", AT 4, 22 ; INK 0 ; "□□□□□" ; AT 5, 6 ; INK 0 ; "□□□□□" ; AT 5, 11 ; INK 4 ; "10 caractere □" ; AT 5, 22 ; INK 0 ; "□□□□"

30 PRINT AT 6, 11 ; INK 4 ; '10 caractere □" ; AT 7, 10 ; INK 4 ; "□□□□", AT 7 19 ; INK 4 ; "□□□"

```
35 INK 4
40 PRINT AT 8, 9; "□□□□"; AT 8, 20; "□□□□";
   AT 9, 8; "□□□□"; AT 9, 21; "□□□□"; AT 10, 9;
   "□□□□"; AT 10, 21; "□□□□"; AT 11, 9;
   "□□□□□□"; AT 11, 20; "□□□□□□"; AT 12, 8;
   "□□□□□□"; AT 12, 20; "□□□□□□"; AT 13, 8; "16
   caractere □"; AT 14, 8: "16 caractere □"
50 CIRCLE INK 1; 128, 78, 3: PRINT AT 7, 14;
   "5_ _ 4"; AT 8, 12; "6_ _ _ _ _ _3"; AT
   9, 11; "7_ _ _ _ _ _ _ 2"; AT 10, 11;
   "8_ _ _ _ _ _ _ 1"; AT 11, 12; "9"
```

Capitolul 9

INSTRUCȚIUNI DE LUCRU CU MEMORIA

9.1. VARIABILELE DE SISTEM

În capitolul 2 a fost prezentată organizarea memoriei (v. fig. 2.1); variabilele de sistem sînt dispuse între adresele 23552 și 23733 cu semnificațiile precizate în tabelul 9.1.

TABELUL 9.1

Nr. crt.	Tip	Adresa	Nume	Conținut
1	N8	23552	KSTATE	Folosită în citirea tastaturii
2	N1	23560	LAST K	Reține ultima tastă apăsată
3	1	23561	REPDEL	Durata (în 1/50 sec) cît trebuie ținută apăsată o tastă pt. a se repeta
4	1	23562	REPPER	Timpul (în 1/50 sec) după care se repetă o tastă apăsată
5	N2	23563	DEFADD	Adresa argumentelor funcțiilor definite de utilizator
6	N1	23565	K DATA	Al doilea octet pentru controlul culorii introdus de la tastatură
7	N2	23566	TVDATA	Controlul culorii, al lui AT și TAB pentru TV
8	X38	23568	STRMS	Adresa canalului atașat căii
9	2	23606	CHARS	Adresa generatorului de caractere minus 256
10	1	23608	RASP	Durata sunetului de eroare
11	1	23609	PIP	Durata sunetului la apăsarea unei taste
12	1	23610	ERR NR	Codul de mesaj minus 1
13	X1	23611	FLAGS	Diferiți indicatori de control ai sistemului BASIC
14	X1	23612	TVFLAG	Indicatorii asociați cu TV
15	X2	23613	ERR SP	Adresa elementului din stiva mașinii utilizat ca adresă de întoarcere în caz de eroare
16	N2	23615	LIST SP	Adresa de întoarcere la listările automate
17	N1	23617	MODE	Specifică cursorul (K, L, C, E, G)

18	2	23618	NEWPPC	Linia la care se sare
19	1	23620	NSPPC	Numărul instrucțiunii în linie la care se sare
20	2	23621	PPC	Numărul liniei pentru instrucțiunea în execuție
21	1	23623	SUBPPC	Numărul instrucțiunii din linie în execuție
22	1	23624	BORDCR	Culoarea BORDER-ului
23	2	23625	E PPC	Numărul liniei curente
24	X2	23627	VARS	Adresa variabilelor
25	N2	23629	DEST	Adresa variabilei asignate
26	X2	23631	CHANS	Adresa datelor de canal
27	X2	23633	CURCHL	Adresa informației curente folosite pentru intrare sau ieșire
28	X2	23635	PROG	Adresa programului BASIC
29	X2	23637	NXTLIN	Adresa următoarei linii din program
30	X2	23639	DATADD	Adresa ultimului element din lista DATA
31	X2	23641	E LINE	Adresa comenzii introduse
32	2	23643	K CUR	Adresa cursorului
33	X2	23645	CH ADD	Adresa următorului caracter care urmează să fie interpretat
34	2	23647	XPTR	Adresa caracterului după semnul întrebării
35	X2	23649	WORKSP	Adresa spațiului de lucru temporar
37	X2	23651	STKEND	Adresa inferioară a stivei calculator
38	X2	23653	STKBOT	Adresa de început a spațiului liber
39	N1	23655	BREG	Registrul B al calculatorului
40	N2	23656	MEM	Adresa spațiului folosit pentru memoria calculatorului
41	1	23658	FLAGS2	Alți indicatori
42	X1	23659	DF SZ	Adresa liniilor din partea de jos a ecranului
43	2	23660	S TOP	Numărul liniei de sus a programului la listarea automată
44	2	23662	OLDPPC	Numărul liniei la care sare CONTINUE
45	1	23664	OSPCC	Numărul din linie la care sare CONTINUE
46	N1	23665	FLAGX	Diverși indicatori
47	N2	23666	STRLEN	Lungimea asignată șirului
48	N2	23668	T ADDR	Adresa următorului element din tabela de sintaxă
49	2	23670	SEED	Variabila pentru RND
50	3	23672	FRAMES	Contorul de cadre
51	2	23675	UDG	Adresa primului grafic definit de utilizator

52	1	23677	COORDS	Coordonata x a ultimului punct PLOT-at
53	1	23678		Coordonata y a ultimului punct PLOT-at
54	1	23679	P POSN	Numărul poziției de scriere pe ecran
55	1	23680	PR CC	Octetul mai puțin semnificativ al adresei pentru noua poziție la care se imprimă prin LPRINT
60	1	23681		NEFOLOSIT
61	2	23682	ECHO E	Numărul coloanei și al liniei
62	2	23684	DF CC	Adresa de afișare pe ecran prin PRINT
63	2	23686	DFCCL	Același lucru pentru partea de jos a ecranului
64	X1	23688	S POSN	Numărul coloanei pentru PRINT
65	X1	23689		Numărul liniei pentru PRINT
66	X2	23690	SPOSNL	Ca S POSN pentru partea de jos a ecranului
67	1	23692	SCR CT	Numără defilările de ecrane
68	1	23693	ATTR P	Culoarea curentă
69	1	23694	MASK P	Folosit pentru culori transparente
70	N1	23696	MASK T	Ca MASK P dar temporar
71	1	23697	PFLAG	Alți indicatori
72	N30	23696	MEMBOT	Arie memorie calculator
73	2	23728		NEFOLOSIT
74	2	23730	RAMTOP	Adresa ultimului octet din aria sistemului BASIC
75	2	23732	P-RAMT	Adresa ultimului octet din RAM

Octeții din memorie de la adresa *23552* la adresa *23733* sînt rezervați pentru operații specifice ale sistemului ; ei pot fi citiți și cîțiva pot fi modificați. Octeții respectivi au numele indicat în coloana a patra a tabelului 9.1 care nu trebuie confundate cu variabilele utilizate în *BASIC* (variabilele formate din mai mulți octeți au primul octet mai puțin semnificativ), Abrevierile din coloana a doua au următoarele semnificații :

X — schimbarea valorii poate duce la blocarea sistemului
N — modificarea variabilei nu are efect asupra funcționării normale a sistemului
număr — numărul de octeți ocupat

O parte din aceste locații de memorie pot fi folosite cu ajutorul instrucțiunilor **POKE** și **PEEK** care, împreună cu **USR**, formează setul instrucțiunilor ce folosesc memoria.

9.2. INSTRUCȚIUNEA USR

Instrucțiunea **USR** are două funcții total diferite în raport cu parametrul său care poate fi un număr sau un șir

Sintaxa : **comandă USR n**

unde „*comandă*" = {**PRINT, RANDOMIZE, LET literă =, GO TO**}

$n \in 0; 65535$]

Exemple : **10 RANDOMIZE USR 64000**

20 PRINT USR 64000

30 LET k=USR 64000

40 PRINT USR 0 (resetare program)

Efect : pornește în execuție un program (rutină) în limbaj de asamblare aflat în memorie la adresa *n;* dacă programul se reîntoarce în *BASIC* (are ultima instrucțiune în limbaj de asamblare RET), atunci **USR** procedează ca o funcție returnînd conținutul perechii de registre *BC* al microprocesorului

Exemplul 9.1 : **10 CLS :FOR i=0 TO 703 : PRINT CHR$(RND*96+ +32) ; : NEXT i**

20 PAUSE 0 : FOR i : 0 TO 22 : RANDOMIZE USR 3190 : NEXT i : REM instrucțiunea RANDOMIZE USR 3490 ridică textul cu un rînd

Sintaxa : **USR „caracter"** (între *a* și *u*, sau *A* și *U* inclusiv, respectiv un caracter grafic definit de utilizator)

Exemplu: **USR "a"**

Efect : returnează o *adresă* (un număr cuprins între *0* și *65535*, care reprezintă adresa primului octet al caracterului grafic definit de utilizator, care se obține în modul grafic apăsînd tasta indicată în comanda USR (astfel **USR "a"** este adresa primului octet al caracterului UDG al tastei a ; tastînd **PRINT USR "a"** se afișează *65368*)

9.3. INSTRUCȚIUNILE PEEK și POKE

Instrucțiunea/comanda **PEEK** permite *citirea* conținutului unui octet de memorie, iar instrucțiunea/comanda **POKE** permite *modificarea* acestui conținut. Conținutul octetului de modificat este un număr cuprins între *0* și *255* (numărul maxim ce se poate reprezenta cu *8* biți). Pentru numere

mai mari ca *255* **se codifică adresei valoarea pe 2 octeți**, cel de al doilea avînd o pondere de *256* ori superioară primului octet. De pildă, dacă se dorește să se știe care este *RAMTOP*-ul la *HC-85*, se va citi conținutul adresei variabilei sistem *RAMTOP* (la adresa *23730* conform tabelului 9.1) care va indica adresa ultimului octet din zona sistemului *BASIC* (o valoare pe doi octeți) :

PRINT PEEK 23370+256*PEEK 23731

calculatorul afișînd cifra *65367*.

Se reamintește că în memorie adresa se păstrează cu octetul mai puțin semnificativ (*LSB* sau *l*) în *stînga* (primul octet). De exemplu :

$$29000 = 72 + 256 * 113$$

Această adresă se va găsi în memorie la adresa *adr* astfel :

— la *adr* : **72**

— la *adr* + *1* : **113**.

Comanda/instrucțiunea **PEEK** nu prezintă nici un pericol în derularea programelor, spre deosebire de **POKE** care modifică conținutul memoriei (o modificare neinspirată duce la crahul sistemului).

Sintaxa : **POKE adr, v**
unde $adr \in [0; 65535]$
$v \in [0; 255]$ (dacă *v* este negativ i se adaugă *256*)

Exemplu : **POKE 23620, 150**

Efect : înscrie în octetul cu adresa „adr" numărul v

Sintaxa : **PEEK adr**
unde $adr \in [0; 5535]$

Exemplu : **PEEK 23620**

Efect : afișează (returnează) conținutul locației de memorie de la adresa „*adr*".

Exemplul 9.2 : determinarea valorilor pentru octetul cel mai puțin semnificativ (*l*) și respectiv cel mai semnificativ (*h*), pentru o valoare *adr* memorată pe doi octeți se obține astfel :

```
10 LET adr=număr :LET h=INT (adr/256) : LET l=adr-256*h
:PRINT l, h, l+256*h
```

De exemplu : pentru $adr = 29000$ rezultă $l = 72$ și $h = 113$.

9.4. UNELE EFECTE REALIZATE CU INSTRUCȚIUNEA POKE

Aceste efecte se obțin *schimbînd conținutul unora dintre variabilele* de sistem indicate în tabelul 9.1 ; în cele ce urmează se prezintă acele efecte care interesează orice utilizator de calculatoare compatibile *SPECTRUM*.

1) **POKE 23561, 1** determină la apăsarea unei taste repetarea ei de minimum două ori ; este o modalitate de protecție a programelor dacă se scrie ca primă instrucțiune (se revine înlocuind *1* cu *33*)

2) **POKE 23568, 8** determină protecție la editarea liniei programului

3) **POKE 23562, 1** realizează o deplasare rapidă a cursorului (util la depanarea programelor)

4) **POKE 23607, 20** alterează scrisul listingului, reprezentînd o metodă de protecție a programelor (se revine înlocuind *20* cu *60*)

5) **POKE 23609, 15** taste sonorizate (se revine înlocuind *15* cu *0*)

6) **POKE 23610, 28** afișează numele firmei constructoare a calculatorului

7) **POKE 23613, 0 : POKE 23614, 0** protecție la **BREAK**

8) **POKE 23658, 8** determină scriere cu majuscule (se revine înlocuind *8* cu *0*)

9) **POKE 23659, n + 2** cu $n \in [3; 24]$ determină ștergerea a *n* linii începînd cu partea de jos a ecranului (se revine cu **POKE 23659, 2**)

10) **POKE 23659, 1** face ecranul dungat instantaneu

11) **POKE 23681** determină scrierea cu litere mari folosind următorul program :

```
10 FOR f=72 TO 79: POKE 23681,f:LPRINT TAB 8; "M. M.
   POPOVICI SOFTWARE": NEXT f: REM scrie la mijlocul
   ecranului
```

Dacă se dorește scrierea în treimea superioară, respectiv inferioară a ecranului se vor face următoarele înlocuiri :

— pentru *treimea superioară :* **FOR f = 64 TO 71 : POKE 23681, f** etc

— pentru *treimea inferioară :* **FOR f = 80 TO 87 : POKE 23681, f** etc : **PAUSE 0**

12) **POKE 23692, 255** inhibă *scroll*-pentru următoarele *255* linii de program

13) **SAVE "nume" POKE 23736, 181** determină salvarea automată a programului

14) **POKE 23739, 111** inhibă scrierea titlului programului la încărcare

15) **POKE 23743, 80** face listing-ul invizibil (se revine înlocuind *80* cu *83*)

16) **POKE 23756, 0** numerotează *0* prima linie a programului (modalitate de protecție a programelor ; se revine înlocuind *0* cu un număr de linie).

9.5. RUTINE ÎN COD MAȘINĂ APELABILE CU INSTRUCȚIUNEA USR

Deoarece instrucțiunea **USR** poate pune în execuție o rutină în cod mașină, se prezintă cîteva asemenea rutine care sînt incluse într-o instrucțiune **DATA**. Ele realizează unele efecte remarcabile ce pot fi folosite în programe proprii. Toate aceste rutine sînt relocabile (adică pot fi stocate la orice adresă ; se recomandă adrese cît mai ridicate).

Exemplul 9.3 :

```
8 REM dispariția imaginii ecranului în PAPER
9 LET adr=60000
10 FOR i=1 TO 23:READ a:POKE adr+i, a:
   NEXT i
```

```
20 DATA 6, 8, 197, 33, 0, 64, 6, 192, 197, 6, 32, 203,
   62, 35, 16, 251, 193, 16, 245, 193, 16, 236, 201
30 FOR d=32 TO 255 : PRINT CHR$(d) ; : NEXT d
40 RANDOMIZE USR adr
```

Exemplul 9.4 :

```
 8 REM scroll rapid spre dreapta a întregului ecran
 9 LRT adr=60100
10 FOR n=1 TO 25 : READ a : POKE adr+n, a :
   :NEXT n
20 DATA 6, 192, 17, 255, 87, 213, 225,43, 197, 1,
   31, 0, 26, 237, 184, 35, 54, 0, 43, 43, 27, 193,
   16, 240, 201
30 FOR i=32 TO 255 : PRINT CHR$(i) ; : NEXT i
40 FOR f=1 TO 32 : RANDOMIZE USR adr :
   NEXT f
```

Exemplul 9.5 :

```
 8 REM scroll rapid stînga a întregului ecran
 9 LET adr=60200
10 FOR n=1 TO 25 : READ a : POKE adr+n,
   a : NEXT n
20 DATA 6, 192, 17, 0, 64, 213, 225, 35, 197, 1, 31,
   0, 26, 237, 176, 43, 54, 0, 35, 35, 19, 193, 16,
   240, 201
30 FOR i=32 TO 255 : PRINT CHR$(i) ; :NEXT i
40 FOR x=1 TO 32 : RANDOMIZE USR adr : NEXT x
```

Exemplul 9.6 :

```
 8 REM scroll rapid stînga treimea superioară a
   ecranului
 9 LET adr=60300
10 FOR n=1 TO 24 :READ a :POKE adr+n,
   a :NEXT n
20 DATA 6, 64, 17, 0, 64, 213, 225, 35, 197, 1, 31,
   0, 26, 237, 176, 43, 119, 0, 35, 35, 19, 193, 16,
   240, 201
30 FOR i=32 TO 255 : PRINT CHR$(i) ; : NEXT i
40 FOR x=1 TO 32 : RANDOMIZE USR adr : NEXT x
```

Exemplul 9.7 :

```
 8 REM scroll rapid stînga partea mijlocie a ecra-
   nului
 9 LET adr=60400
10 FOR n=1 TO 25 :READ a : POKE adr+n,
   a : NEXT n
```

```
20 DATA 6, 64, 17, 0, 72, 213, 225, 35, 197, 1, 31,
   0, 26, 237, 176, 43, 119, 0, 35, 35, 19, 193, 16,
   240, 201

30 FOR i=32 TO 255:PRINT CHR$(i); :NEXT i

40 FOR x=1 TO 32:RANDOMIZE USR adr:NEXT x
```

Exemplul 9.8 :

```
8 REM scroll rapid stînga treimea de jos a ecranului

9 LET adr=60500

10 FOR n=1 TO 25:READ a:POKE adr+n,
   a:NEXT n

20 DATA 6, 64, 17, 0, 80, 213, 225, 35, 197, 1, 31,
   0, 26, 237, 176, 43, 119, 0, 35, 35, 19, 193, 16,
   240, 201

30 FOR i=32 TO 255:PRINT CHR$(i);:NEXT i

40 FOR x=1 TO 32: RANDOMIZE USR adr:NEXT x
```

Exemplul 9.9 :

```
8 REM inversarea instantanee a PAPER-ului cu
   INK-ul

9 LET adr=60600

10 FOR n=1 TO 19:READ a:POKE adr+n,
   a:NEXT n

20 DATA 33, 0, 64, 6, 24, 197, 6, 0, 126, 238, 255,
   119, 35, 16, 249, 193, 16, 243, 201

30 FOR i=32 TO 255:PRINT CHR$(i);:NEXT i

40 RANDOMIZE USR adr
```

Exemplul 9.10 :

```
8 REM sunet (multi-beep)

9 LET adr=60700

10 FOR n=1 TO 24: READ a:POKE adr+n,
   a:NEXT n

20 DATA 1, 4, 250, 33, 0, 2, 17, 15, 0, 229, 213,
   197, 205, 181, 3, 193, 209, 225, 125, 145, 111,
   16, 242, 201

30 RANDOMIZE USR adr
```

Introducînd linia 25 POKE adr + 2, cifră (între *0 și 255*), se pot obține alte efecte sonore interesante (de pildă cifrele : *100, 255, 15*, ș.a.)

Exemplul 9.11 :

```
8 REM sunet

9 LET adr=60800

10 FOR n=1 TO 31:READ a:POKE adr+n,
   a:NEXT n
```

```
20 DATA 58, 72, 92, 31, 31, 31, 6, 240, 14, 254, 37,
   32, 6, 238, 16, 237, 121, 38, 238, 45, 32, 244,
   238, 16, 237, 121, 46, 254, 16, 236, 201

30 RANDOMIZE USR adr
```

Exemplul 9.12 :

```
 8 REM sunet

 9 LET adr=60900

10 FOR n=1 TO 26:READ a:POKE adr+n,
   a:NEXT n

20 DATA 6, 10, 197, 33, 0, 3, 17, 1, 0, 229, 205,181,
   3, 225, 17, 16, 0, 167, 237, 82, 32, 240, 193,16,
   233, 201

30 RANDOMIZE USR adr
```

Exemplul 9.13 :

```
 8 REM efect grafic

 9 LET adr=61000

10 FOR n=1 TO 14:READ a:POKE adr+n,
   a:NEXT n

20 DATA 42, 120, 92, 38, 4, 17, 64, 89, 1, 128, 1,
   237, 176, 201

30 FOR i=1 TO 7:BEEP .02, i*7:RANDOMIZE
   USR adr:PAUSE 2:NEXT i:BEEP .1, -20:
   CLS
```

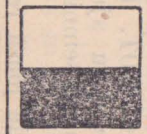
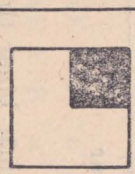
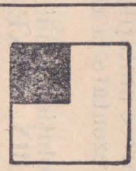
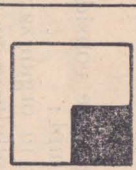
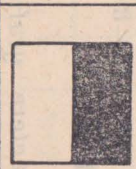
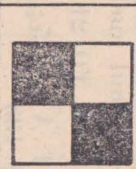
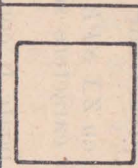
Exemplul 9.14 :

```
 8 REM efectul "perdea"-cortină spre dreapta

 9 LET adr=65100

10 FOR n=1 TO 45:READ a:POKE adr+n,
   a:NEXT n

20 DATA 14, 32, 33, 0, 88, 6, 24, 17, 32, 0, 229,54,
   18, 25, 16, 251, 197, 33, 208, 0, 17, 32,0, 205,
   181, 3, 193, 225, 229, 6, 24, 17, 32, 0, 54, 9,25,
   16, 251, 225, 35, 13, 32, 217, 201

30 FOR i=32 TO 255:PRINT CHR$(i);:NEXT i

40 RANDOMIZE USR adr
```

Exemplul 9.15 :

```
 8 REM scroll vertical rapid în sus cu oprire la
   locul dorit

 9 LET adr=64050

10 FOR n=1 TO 6:READ a:POKE adr+n,
   a:NEXT n

20 DATA 6, 17, 205, 0, 14, 201

30 FOR i=32 TO 255:PRINT CHR$(i);:NEXT i

40 FOR x=1 TO 10:RANDOMIZE USR adr:NEXT
   x:REM x—nr. de linii
```

Tabelul 9.2

Tasta / Mod de tastare	1	2	3	4	5	6	7	8
CS și 9 CS și tasta								
CS și 9 tasta								

Tabelul 9.3

Caracterul semigrafic																
Numărul n din funcția CHR$ n	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143

9.6. DEFINIREA DE NOI CARACTERE GRAFICE

9.6.1. Caractere semigrafice predefinite

Calculatoarele compatibile cu *ZX SPECTRUM* au definite pe tastele de la *1* la *8* un număr de *16* caractere semigrafice predefinite, care se obțin conform tabelului 9.2.

Evident, cu ajutorul lor poate fi construită o imagine oarecare pe ecran.

Plasarea unui anumit caracter semigrafic într-o poziție oarecare pe ecran se poate realiza cu ajutorul instrucțiunilor cunoscute **PRINT, PRINT AT** și **CHR$**, deoarece **CHR$(n)** generează caracterul corespunzător codului *n* așa cum se indică în tabelul 9.3.

Exemplul 9.16 :

```
9 REM desenul unei scări cu instrucțiunea CHR$
10 FOR n=0 TO 21
20 PRINT AT n, 5; CHR$ 138: PRINT AT n, 6; CHR$ 133
30 PLOT 40, 175-8*n: DRAW 15, 0
40 NEXT n
```

9.6.2. Definirea de noi caractere grafice folosind o matrice de 8×8 pixeli

După cum se știe ecranul este considerat ca o matrice de unități grafice cu următoarele dimensiuni :

a) *176 linii* × *256 coloane* în organizarea de **înaltă rezoluție,** fiecare punct din această matrice numindu-se *pixel* și corespunzînd unei poziții pe ecran a spotului luminos ; instrucțiunile **PLOT, DRAW, CIRCLE, POINT,** specifică coordonatele acestor puncte și cu ajutorul lor se pot realiza diferite caractere grafice noi (de exemplu semnul radicalului sau numărul *pi*) ;

b) *22 linii* × *32 coloane* în organizarea de **joasă rezoluție** denumită și *reprezentare alfanumerică* deoarece unitățile de reprezentare sînt caractere alfanumerice ; acestea sînt reprezentate la rîndul lor pe cîte o matrice de *8* × *8 pixeli* (puncte).

Organizarea de joasă rezoluție permite definirea de noi caractere grafice folosind instrucțiunile **BIN, POKE, USR** și **DATA.**

Sintaxa : **BIN număr binar**
Exemplu : **BIN 10010111**
Efect : convertește un număr binar (în baza *2*) într-un număr zecimal

Pentru a se defini un nou caracter, de exemplu litera grecească *pi*, se procedează astfel :

1) Se desenează pe hîrtie de caiet pentru matematică o matrice *8* × *8 pixeli* în care, prin înegrirea unor pătrățele, se desenează imaginea caracterului dorit (fig. 9.1).

Valorile numerice din instrucțiunea **BIN** se stabilesc pe baza următoarei reguli : „se notează *0* pentru pixel nedesenat (neînegrit) și *1* pentru pixel desenat (înegrit)" — v. fig. 9.1. Pentru simplificarea scrierii, în

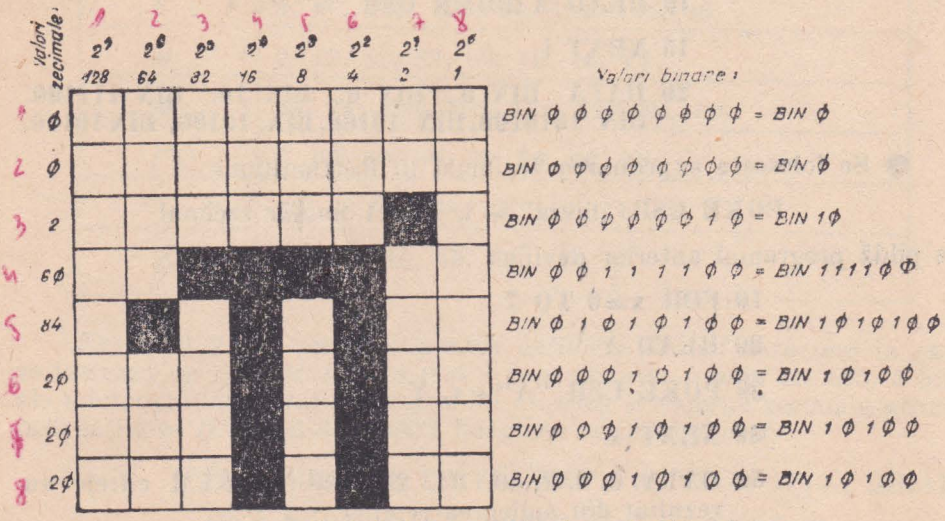


Fig 9.1 Definirea caracterului grafic π

instrucțiunea **BIN** se pot scrie *mai puțin de 8 cifre* și anume **începînd cu poziția unde apare prima cifră** *1* (dacă cele *8* cifre sînt *0* se scrie o singură cifră *0* după **BIN**). Este însă mai comod să se transforme valorile binare în zecimal conform relației

$$\sum 2^{poziție-1}$$

așa cum se indică în fig. 9.1 (de exemplu $2^7 = 128$).

2) Se programează caracterul nou definit folosind una din următoarele două metode :

● Se scriu *8* instrucțiuni de forma

POKE USR "litera" + i BIN număr binar

unde *„litera"* este o literă din șirul *a ... u*, aleasă pentru a defini noul caracter, iar $i \in [0; 7]$. De exemplu, pentru caracterul *pi* programul va fi :

```
10 POKE USR "a", BIN 0
20 POKE USR "a"+1, BIN 0
30 POKE USR "a"+2, BIN 10
40 POKE USR "a"+3, BIN 111100
50 POKE USR "a"+4, BIN 1010100
60 POKE USR "a"+5, BIN 10100
70 POKE USR "a"+6, BIN 10100
80 POKE USR "a"+7, BIN 10100
90 PRINT "A" :REM se apasă A în modul grafic
```

sau într-o formă simplificată

```
5 FOR i=0 TO 7
10 READ A :POKE USR "a"+i, A
15 NEXT i
20 DATA BIN 0, BIN 0, BIN 10, BIN 111100,
   BIN 1010100, BIN 10100, BIN 10100, BIN 10100,
```

● Se folosește exprimarea zecimală în instrucțiunile

POKE USR „litera" + i, DATA număr zecimal

De pildă programul anterior devine:

```
10 FOR x=0 TO 7
20 READ Y
30 POKE USR "A"+x, Y
40 NEXT x
50 DATA 0, 0, 2, 60, 84, 20, 20, 20 :REM cifrele au
   rezultat din aplicarea relației ∑ 2^(poziție−1)
60 PRINT "A" :REM se apasă A în modul grafic
```

În tabelul 9.4 sînt indicate valorile zecimale pentru definirea unor litere grecești și semne uzuale.

Tabelul 9.4.

Litera sau semnul grafic	Valorile zecimale pentru instrucțiunea DATA
σ	0, 62, 64, 64, 68, 68, 56, 0
τ	0, 62, 8, 16, 18, 17, 14, 0
α	0, 0, 0, 104, 144, 144, 104, 0
β	0, 28, 34, 60, 34, 34, 76, 0
γ	0, 0, 72, 48, 32, 32, 32, 0
μ	0, 0, 36, 36, 36, 58, 32, 64
θ	0, 48, 74, 124, 72, 72, 48, 0
ν	0, 0, 3, 36, 20, 24, 16, 0
ω	0, 68, 130, 130, 146, 146, 108, 0
ε	0, 60, 66, 64, 56, 64, 66, 60
Γ	254, 66, 66, 64, 64, 64, 64, 224

λ	0, 56, 4, 4, 28, 36, 34, 0
ρ	0, 0, 0, 2, 255, 2, 0, 0
δ	0, 96, 128, 128, 96, 144, 144, 96
φ	0, 12, 82, 82, 60, 16, 16, 16
Φ	60, 24, 126, 153, 153, 126, 24, 60
∩	0, 60, 66, 66, 66, 66, 66, 66
→	56, 68, 68, 68, 68, 120, 64, 64, 0
△	0, 0, 0, 16, 40, 68, 254, 0

Exemplul 9.17 : program pentru definirea de noi caractere în care se tastează un rînd de *8* cifre *0* și *1* care definesc o linie din cele *8* linii ale noului caracter grafic; programul afișează și valorile zecimale pentru instrucțiunea **DATA** și desenează imaginea noului caracter.

```
5 CLS :FOR n=0 TO 71 STEP 8 :PLOT 0, 175-n :DRAW
  63, 0 :PLOT n, 175 :DRAW 0, -63 :NEXT n
9 FOR i=0 TO 7
10 INPUT a$
11 IF LEN a$ < > 8 THEN GO TO 10
12 LET a=0
13 FOR j=1 TO 8
14 IF a$(j)="1" THEN PRINT AT i, j-1; "■" :PLOT
  120+j, 80-i
15 GOSUB 25
16 NEXT j :PRINT AT i, 10; a
18 NEXT i
19 PAUSE 0
20 GO TO 5
25 IF POINT (j*8-4, 171-i*8)=1 THEN LET a=a+2 ↑
  (8-j)
26 RETURN
```

9.6.3. Definirea de noi caractere grafice folosind mai multe matrice 8×8 pixeli

O figură mai complicată se realizează pe mai multe matrice 8×8 pixeli, fiecare matrice fiind tratată ca un nou caracter grafic. De asemenea

se pot defini max. *21* de caractere grafice folosind literele *a* . . . *u* cu ajutorul următoarelor programe :

a) **10 FOR f=USR "a" TO USR "ultima literă"+7 :READ a :POKE f, a :NEXT f :REM ultima literă semnifică litera ultimă folosită pentru definirea de noi caractere în șirul a . . .u**

20 DATA șir de 8 valori numerice pentru fiecare literă

b) **10 FOR x=1 TO nr. caractere : READ a$:FOR y=0 TO 7 : READ a :POKE USR a$+y, a :NEXT y :NEXT x**

20 DATA "a", șir de 8 valori numerice, "b", șir de 8 valori, etc.

c) **10 LET a$="abcdefghijklmnopqrstu" :REM șirul literelor a . . . u**

20 FOR i=1 TO LEN a$

30 FOR n=0 TO 7 :READ k : POKE USR a$(i)+n, k :NEXT n

40 NEXT i

50 DATA șir de 8 valori numerice pentru fiecare caracter

d) **10 FOR f=0 TO (8*nr. caractere—1) :READ a : POKE 65368+ f, a :NEXT f**

20 DATA șir de 8 valori numerice pentru fiecare caracter

Acest program de UDG-uri noi definite se salvează ca rutină în cod mașină cu comanda

SAVE "nume" CODE 65368, nr. caractere * 8

Exemplul 9.18 : definirea unui helicopter pe 3 și respectiv 6 caractere.

```
30 BORDER 1 :PAPER 1 :INK 6 :CLS

40 POKE USR "a", 0 :POKE USR "a"+1, BIN 11000000 :POKE
   USR "a"+2, BIN 11000000 :POKE USR "a"+3, 255 :POKE
   USR "a"+4, 255 :POKE USR "a"+5, BIN 11111 :POKE
   USR "a"+6, 0 :POKE USR "a"+7, 0

50 POKE USR "b", 255 :POKE USR "b"+1, BIN 1 :POKE USR
   "b"+2, BIN 1111 :POKE USR "b"+3, BIN 11111100 :POKE
   USR "b"+4, BIN 11111110 :POKE USR "b"+5, 255 :POKE
   USR "b"+6, 255 :POKE USR "b"+7, BIN 1111

60 POKE USR "c", BIN 11111110 :POKE USR "c"+1, 0 :POKE
   USR "c"+2, BIN 11100000 : POKE USR "c"+3, BIN 10000 :
   POKE USR "c"+4, BIN 10100000 :POKE USR "c"+5, BIN
   11110000 :POKE USR "c"+6, BIN 11110000 :POKE USR "c"
   +7, BIN 11100000

70 POKE USR "d", BIN 11 :POKE USR "d"+1, BIN 1 :POKE
   USR "d"+2, BIN 1111 :POKE USR "d"+3, BIN 11111100 :
   POKE USR "d"+4, BIN 11111100 :POKE USR "d"+5, 255 :
   POKE USR "d"+6, 255 :POKE USR "d"+7, BIN 1111
```

```
80 POKE USR "e", BIN 10000000 :POKE USR "e"+1, 0 :POKE
   USR "e"+2, BIN 11100000 :POKE USR "e"+3, BIN 10000 :
   POKE USR "e"+4, BIN 10100000 :POKE USR "e"+5, BIN
   11110000 :POKE USR "e"+6, BIN 11110000 :POKE USR
   "e"+7, BIN 11100000

100 FOR w=1 TO 3 :CLS :PRINT AT 20, 5; "HELICOPTER (3
    CARACTERE)" :FOR n=0 TO 28

110 BEEP 1/100, -5 :PRINT AT 3, n; "ADE" :BEEP 1/100, 5 :
    PRINT AT 2, n; "— ABC" : REM deplasarea helicopterului ;
    tastele ADE şi ABC se apasă în modul grafic

120 NEXT n

130 NEXT w

140 PRINT AT 20, 4; "Apăsaţi o tasta oarecare" :PAUSE 0 :BORDER
    2 :PAPER 0 :INK 7 :CLS

150 PRINT AT 21, 5;"HELICOPTER (6 CARACTERE)"

160 DATA 0, 127, 0, 0, 14, 49, 65, 129

170 DATA 8, 227, 20, 8, 8, 28, 188, 188

180 DATA 0, 255, 0, 0, 0, 0, 1, 6

190 DATA 129, 129, 129, 129, 127, 127, 63, 31

200 DATA 254, 254, 255, 255, 255, 255, 255, 252

210 DATA 6, 11, 3, 143, 255, 252, 224, 0

220 DATA 0, 255, 0, 0, 0, 0, 128, 96

230 DATA 16, 199, 40, 16, 16, 56, 61, 61

240 DATA 0, 254, 0, 0, 112, 140, 130, 129

250 DATA 96, 176, 192, 241, 255, 63, 7, 0

260 DATA 127, 127, 255, 255, 255, 255, 255, 43

270 DATA 129, 129, 129, 129, 254, 254, 252, 248

280 RESTORE 160

290 FOR I=0 TO 95 :READ X :POKE USR "G"+I, X :NEXT I

300 PRINT AT 12, 4; "GHI"; AT 13, 4; "JKL"; AT 12, 24;
    "MNO"; AT 13, 24; "PKR" :REM tastele GHI, JKL, MNO şi
    PKR se apasă în modul grafic

310 PRINT # 0; AT 0, 4; "Apăsaţi o tastă oarecare" :PAUSE 0 :
    BORDER 5 :PAPER 5 :INK 1 :CLS

320 FOR w=1 TO 3 :CLS :PRINT AT 20, 5; "HELICOPTER (6
    CARACTERE)" :FOR n=0 TO 28

330 BEEP 1/100, -5 :PRINT AT 3, n; "MNO"; AT 4, n; "PKR" :
    BEEP 1/100, 5 :PRINT AT 3, n; "—MNO"; AT 4, n; "—PKR"

340 NEXT n

350 NEXT w
```

9.6.4. Definirea de noi caractere grafice pe fiecare tastă cu un singur simbol

Pentru un număr mare de caractere grafice se pot folosi tastele cu singur simbol înscris pe ele (deci nu se pot folosi tastele cu două simboluri cum ar fi $<=$, $>=$, $<>$, etc). Tastele ce se definesc pentru literele din șirul *a ... u* se tratează așa cum s-a indicat anterior. Pentru restul tastelor programul este următorul:

```
10 LET B=4834
20 FOR I=15616 TO 165383
30 POKE (I+B), PEEK I :NEXT I
40 POKE 23607, 249 :RESTORE 110
50 FOR I=1 TO nr. caractere (taste)
60 READ A$
70 LET ADR=63744+8*CODE A$
80 FOR J=0 TO 7
90 READ A :POKE ADR+J, A
100 NEXT J :NEXT I
110 DATA "!", șir de 8 valori, etc :REM se iau toate tastele
    în ordine începînd cu rîndul de sus al claviaturii
```

În legătură cu acest program se vor reține următoarele:

— numărul maxim de noi caractere ce se pot defini: **43**;

— programul se va scrie cu **litere majuscule.**

— instrucțiunea **POKE 23607, 249** determină folosirea noilor caractere (cu **POKE 23407. 60** se revinela caracterele normale);

— programul care conține noile caractere se salvează ca rutină în cod mașină cu comanda

```
SAVE "nume" CODE 64000, 768
```

9.6.5. Definirea de noi caractere grafice folosind variabila de sistem UDG

Conform tabelului 9.1, variabila de sistem *UDG* de la locația de memorie *23675* conține adresa primului grafic definit de utilizator. Programul următor realizează definirea a maximum *21* caractere grafice în șirul literelor *a ... u*, fiecare tastă fiind apăsată în modul grafic.

```
10 LET a=PEEK 23675+256*PEEK 23676
20 FOR b=a TO a+nr. caractere*8-1
30 READ c :POKE b, c :NEXT b
40 DATA șir de 8 valori zecimale pentru fiecare nou caracter grafic
```

9.7. FOLOSIREA UNUI SET DE CARACTERE REALIZAT DE FIRME CREATOARE DE SOFT

Unele programe realizate de firme specializate oferă seturi complete de noi caractere avînd lungimea de *768* octeți realizate în cod mașină. În aceste cazuri se va proceda după cum urmează :

a) Se salvează pe casetă setul de caractere la adresa unde este stocat (exemplu :

SAVE "ALDINE" CODE 64572, 768.

b) În programul de apelare a acestui set de caractere el se încarcă la adresa dorită *adr* (exemplu :

LET adr=50000 :LOAD "ALDINE" CODE adr, 768).

c) În programul *BASIC* care folosește setul de caractere se introduc instrucțiunile următoare :

```
20 LET litere=9600
30 LET adr=50000 :GOSUB litere
9600 LET H = INT(adr/256) : LET L = adr−256*H
9610 POKE 23606, L :POKE 23607, H−1 :RETURN
```

9.8. FOLOSIREA MAI MULTOR SETURI DE CARACTERE GRAFICE DEFINITE PE LITERELE DIN SIRUL a . . . u

Programele științifice și tehnice operează de regulă cu mai multe seturi de caractere definite pe literele din șirul *a . . . u*. În astfel de cazuri se procedează după cum urmează :

a) Se salvează pe bandă toate seturile de noi caractere la *aceeași adresă 65368* cu titluri diferite (exemplu :

SAVE "udg1" CODE 65368, nr. caractere*8 : SAVE "udg2" CODE 65368, nr. caracter*8, etc).

b) În programul de apelare aceste *UDG*-uri se încarcă la adresa dorită (exemplu :

LOAD "udg1" CODE 64000 :LOAD "udg2" CODE 63000, etc.).

c) În programul BASIC care folosește aceste seturi de caractere se introduc instrucțiunile :

```
5 LET udg=9600
10 LET adr=64000 :GOSUB udg :REM apelarea primului set
   "udg1" încărcat la adresa 64000
.
.
.
9600 LET h=INT(adr/256) : LET l=adr−256*h : POKE 23675;
   l :POKE 23676, h :RETURN
```

În mod analog se apelează celălalt set de caractere „*udg2*" încărcat la adresa *63000* înlocuind în linia *5* cifra *64000* cu *63000*.

Exemplul 9.19 : Folosind datele de programare cunoscute se pot magina secvențe de grafică dinamică așa cum se prezintă în programul următor intitulat „*Extratereștrii la piramide*".

```
1 BORDER 2 :PAPER 1 :INK 7 :CLS :PRINT AT 12, 1 ; FLASH 1 ;
"EXTRATERESTRII_ _ LA PIRAMIDE ! !"
2 FOR n=25 TO 0 STEP -1 :BEEP .005, n :BEEP .005, n-1 :
BEEP .005, n+1 :NEXT n :PAUSE 50
4 REM desenarea unui extraterestru
5 FOR x=0 TO 7 :READ y :POKE USR "a"+x, y :NEXT x
6 DATA 0, 60, 126, 213, 126, 60, 66, 66
10 BORDER 0 :PAPER 1 :INK 6 :CLS
20 REM trasarea planșeului piramidelor
30 FOR y=0 TO 20 STEP 2
40 PLOT 0, y
50 DRAW 255, 0
60 NEXT y
65 REM desenarea piramidelor
70 FOR n=100 TO 220 STEP 30
80 FOR x=-10-n/10 TO 10+n/10
90 PLOT n, 35+n/10
100 DRAW x, -n/4
110 NEXT x ; NEXT n
112 REM lansarea extratereștrilor și urmărirea lor cu raze laser
114 LET h=RND*31
115 FOR v=0 TO 20
117 PRINT AT v, h ; "_" ; AT v+1, h ; INK 4 ; "A" :REM tasta A
apăsată în modul grafic pentru a reda imaginea extraterestrului
120 LET x=RND*255
130 LET y=RND*104+71
160 PLOT 0, 0 :DRAW OVER 1 ; x, y :REM raza laser ce pornește
din colțul din stînga jos al ecranului
170 BEEP .01, x/4
180 PLOT 0, 0 :DRAW OVER 1 ; x, y
190 IF ATTR (v+1, h)=14 THEN GO TO 500 :REM condiția de
marcare a lovirii extraterestrului cu raze laser
```

```
200 NEXT v

205 PRINT AT 21, h; FLASH 1; INK 2; PAPER 6; "A" :PRINT
    # 1; AT 0, 12; "ATERIZAT" :PAUSE 30 :PRINT # 1; AT
    0, 12; "_ _ _ _ _ _ _ _ _" :GO TO 114

500 PRINT AT v+1, h; "_"; FLASH 1; "ATINS" :PAUSE 20:
    PRINT AT 0, 13 ; FLASH 0 ; PAPER 1 ; INK 1 ; "_ _ _ _ _" :
    REM anunțul doborîrii unui extraterestru

510 PAUSE 20

520 GO TO 114
```

Exemplul 9.20 : salvarea unei mire ca SCREEN.

```
  1 BORDER 0 :PAPER 0 :INK 7 :CLS

  2 POKE USR "a", 60 :POKE USR "a"+1, 36 :POKE USR
    "a"+2, 102 :POKE USR "a"+3, 165 :POKE USR "a"+4,
    „165 :POKE USR "a"+5, 102 :POKE USR "a"+6, 36 :POKE
    USR "a"+7, 60 :REM definirea unui nou caracter grafic

  5 PLOT 40, 65 :DRAW 170, 0 :DRAW 0, −25 :DRAW −170,
    0 :DRAW 0, 25 :PLOT 35, 170 :DRAW 180, 0 :DRAW 0, −35 :
    DRAW −180, 0 :DRAW 0, 35

 10 PLOT PAPER 2; INK 4; 144, 100 :DRAW PAPER 2; INK 4;
    95, 0 :DRAW PAPER 2; INK 4; 0, −40 :DRAW PAPER 2;
    INK 4; −95, 0 :DRAW PAPER 2; INK 4; 0, 40

 20 PLOT INK 6; 3, 3 :DRAW INK 6; 0, 170 :DRAW INK 6;
    249, 0 :DRAW INK 6; 0, −170 :DRAW INK 6; −249, 0

 25 PRINT FLASH 1 ; PAPER 1 ; INK 5 ; AT 20, 6 ; "20 caractere
    □"; AT 20, 6; INVERSE 1; "PROGRAMUL SE INCARCA" :
    REM tasta 8 în modul grafic

 30 FOR f=2 TO 19 ; PRINT INK 5 ; PAPER 2 ; AT f, 1 ; "□□□" :
    NEXT f :PRINT PAPER 2; INK 5; AT 3, 2; "0"; AT 4, 2;
    "R"; AT 5, 2; "G"; AT 6, 2; "A"; AT 7, 2; "N"; AT 8, 2;
    "E"; AT 10, 2; "D"; AT 11, 2; "E"; AT 13, 2; "M"; AT 14,
    2; "A"; AT 15, 2; "S"; AT 16, 2; "I"; AT 17, 2; "N";
    AT 18, 2; "I"

 40 PRINT INK 2; AT 9, 30; "□"; INK 2; AT 10, 30; "□"

 50 INK 2 :FOR f=56 TO 87 :PLOT 240, f :FOR f=58 TO 87 :
    PLOT 241, f :NEXT f :FOR f=61 TO 87 :PLOT 242, f :NEXT
    f :FOR f=64 TO 87 :PLOT 243, f :NEXT f :FOR f=68 TO 87 :
    PLOT 245, f :NEXT f

 60 FOR f=70 TO 87 :PLOT 246, f :NEXT f :FOR f=72 TO 87 :
    PLOT 247, f :NEXT f

 70 FOR f=10 TO 13 :PRINT INK 6; FLASH 1; AT f, 19; "10
    caractere A în modul grafic" :NEXT f

 80 PRINT AT 2, 8; "R U L M E N T I"
```

```
90 PAUSE 0
100 SAVE "RULMENTI" SCREEN$
```

Se resetează calculatorul și apoi se tastează comanda

```
LOAD "RULMENTI" SCREEN$
```

banda fiind derulată la începutul programului salvat ca SCREEN.

Exemplul 9.21 : modificarea setului de caractere al calculatorului

```
10 CLEAR 59999 :LET a=0 :RESTORE 14
11 FOR i=60000 TO 60073 :READ p :POKE i, p; LET a=a+p
   :NEXT i
14 DATA 0, 33, 0, 61, 17, 0, 250, 1, 0, 3, 237, 176, 33, 0, 250, 17, 0,
   3, 203, 78, 40, 2, 203, 198, 203, 86, 40, 2, 203, 206, 203, 94, 40,
   2, 203, 214, 203, 102, 40, 2, 203, 222, 203, 110, 40, 2, 203, 230,
   203, 118, 40, 2, 203, 238, 203, 126, 40, 2, 203, 246, 35, 27, 122,
   179, 254, 0, 32, 206, 62, 249, 50, 55, 92, 201
15 RANDOMIZE USR 60000
20 CLS :POKE 23607, 249 :PRINT AT 12, 4: "M. M. POPOVICI
   SOFTWARE ©"
30 POKE 23607, 60
```

Rutina este lansată în execuție cu **RANDOMIZE USR 60000**, iar folosirea caracterelor modificate cu **POKE 23607, 249**. Revenirea la caracterele normale ale calculatorului se face cu comanda

```
POKE 23607, 60
```

9.9. PROBLEME

P 9.1 : Să se definească un *tanc* folosind *2* taste literale.

Rezolvare :

```
10 CLS :FOR f=USR "a" TO USR "b"+7 :READ
   a :POKE f, a :NEXT f
20 DATA BIN 1, BIN 1111, BIN 1111, BIN 111, BIN
   1111111, 255, 255, BIN 1111111
30 DATA BIN 1000, BIN 11010000, BIN 11100000,
   BIN 11000000, BIN 11111100, BIN 11111110, BIN
   11111110, BIN 11111100
40 PRINT AT 11, 15; "AB" :REM literele A și B se
   tastează în modul grafic.
```

P 9.2 : Să se deseneze un *camion* prevăzut cu un *tun*, definit pe *2* taste literale.

Rezolvare :

```
10 CLS :FOR x=1 TO 2 :READ a$ :FOR y=0 TO 7:
   READ a :POKE USR a$+y, a :NEXT y :NEXT x
20 DATA "a", 0, BIN 1111100, BIN 1000100, BIN
   11111100, 255, 255, BIN 1111111, BIN 11000000
```

```
30 DATA "b", BIN 100, BIN 1000, BIN 11110000, BIN
   11100000, BIN 11111100, BIN 11111100, BIN
   11111100, BIN 1110000
40 PRINT AT 11, 15; "AB" :REM literele A și B se
   tastează în modul grafic
```

P 9.3 : Dintre piesele jocului de șah să se deseneze „*calul*" pe patru taste literale.

Rezolvare :

```
 8 REM desenul "calului"
10 CLS :FOR x=1 TO 4
20 READ a$
30 FOR y=0 TO 7
40 READ a :POKE USR a$+y, a
50 NEXT y
70 DATA "a", 1, 3, 7, 15, 15, 15, 15, 7
80 DATA "b", 32, 96, 240, 248, 252, 254, 254, 230
90 DATA "c", 3, 3, 3, 3, 7, 15, 31, 0
100 DATA "d", 192, 192, 192, 192, 224. 240, 248, 0
110 PRINT AT 11, 25; "AB"; AT 12, 15; "CD" :REM
    tastele A, B, C și D se apasă în modul grafic
```

P 9.4 : Să se deseneze un *submarin în mișcare*

Rezolvare :

```
 8 REM submarin în mișcare
10 CLS :RESTORE 20
15 FOR f=USR"a" TO USR "c"+7 :READ a :POKE
   f, a :NEXT f
20 DATA 0, 0, 0, BIN 1111, BIN 10, BIN 1111111, BIN
   1111111, BIN 111111
30 DATA BIN 111, BIN 111, BIN 11111, BIN 11111,
   BIN 11111, 255, 255, 255
40 DATA 0, 0, 0, 0, 0, BIN 11111110, BIN 11111110,
   BIN 11111110
50 FOR x=0 TO 28
60 PRINT INK 1 ; AT 20, x ; "_ABC" :BEEP .01,
   -10 :PAUSE 10
70 IF x=28 THEN PRINT AT 20, 28 :"_ _ _"
80 NEXT x
90 GO TO 10
```

P 9.5 : Să se deseneze un *cartier de blocuri* pe malul unui *rîu.*

Rezolvare :

```
10 CLS :FOR f=0 TO 7 :READ a :POKE USR ,"a"+f,
„a :NEXT f
20 DATA 255, 255, 153, 153, 153, 153, 153, 255
30 FOR a=0 TO 31
40 FOR b=15+INT(RND*6) TO 21 :PRINT AT b,
a ; BRIGHT 1 ; "A" :NEXT b :REM tasta A se apasă
în modul grafic și rezultă un șir aleator de blocuri
50 NEXT a
60 FOR w=1 TO 10 :BEEP .01, 40 :BEEP .03, 60 :BEEP
.18, -8 :NEXT w :BEEP 1, -20
70 LET b=5 :FOR a=0 TO 255 :PLOT a, 120 :DRAW 0,
b :LET b=b+SGN (RND-.5)
80 IF b<0 THEN LET b=0
90 IF b>120 THEN LET b=120
100 NEXT a :REM trasarea reliefului de pe un mal
110 LET b=5 :FOR a=0 TO 255 :PLOT a, 80 :DRAW 0,
b :LET b=b+SGN (RND-.5)
120 IF b<0 THEN LET b=0
130 IF b>80 THEN LET b=80
140 NEXT a :REM trasarea reliefului de pe celălalt mal
```

Capitolul 10

INSTRUCȚIUNI PENTRU PORTURI, CĂI, CANALE, IMPRIMANTA, MICRODRIVE ȘI DISCHETE

10.1. INSTRUCȚIUNI PENTRU PORTURI, CĂI ȘI CANALE

10.1.1. Instrucțiuni pentru porturi, (IN OUT)

Instrucțiunile/comenzile **IN** și **OUT** reprezintă un cuplu de instrucțiuni opuse, asemănătoare cuplului **PEEK-POKE**, cu deosebirea că ele lucrează cu *porturi* în loc de locații de memorie. Calculatoarele compatibile cu *ZX SPECTRUM* au **65536 porturi de intrare/ieșire** care sînt utilizate pentru comunicarea cu perifericele. Ca și în cazul locațiilor de memorie, un port conține un număr $n \in [0; 255]$.

Sintaxa : **IN adr**

OUT adr, n unde $n \in [0; 255]$

Exemplu : **10 IF IN 254=191 THEN GO TO 500 :OUT 254, 0**

Efect **IN** *citește* valoarea înscrisă în *portul adr*, iar **OUT** *trimite* valoarea $n \in [0; 255]$ la portul *adr*.

Principalele porturi sînt cele ale tastaturii și difuzorului.

● *Tastatura* este împărțită în *4* rînduri și fiecare rînd este divizat în două jumătăți. Pentru tastatură sînt **8 porturi** repartizate cîte unul pentru fiecare semirînd, care au rolul de a *citi tastele apăsate*. Astfel :

IN 65278 citește tastele *CS ... V*
IN 65022 citește tastele *A ... G*
IN 65410 citește tastele Q ... *T*
IN 63486 citește tastele *1 ... 5*
IN 61438 citește tastele *0 ... 6*
IN 57342 citește tastele *P ...* Y
IN 49150 citește tastele *CR ... H*
IN 32766 citește tastele *SP ... B*

Relația pentru obținerea adresei portului este

$$254 + 256 * (255 - 2 \uparrow m) \text{ unde } m \in [0; 7]$$

(semirîndurile tastelor se consideră ca poziție *dinspre exteriorul tastaturii*)

Dacă nici o tastă nu este apăsată și casetofonul nu merge, valoarea citită va fi:

191

(biții *b0* ... *b4* sînt asociați celor 5 taste din semirînd, iar bitul *b6* conectorului de casetofon).

● *Portul difuzorului* este portul de ieșire cu adresa *254* care determină și culoarea **BORDER**-ului pe care îl face negru dacă $m = 0$

```
OUT 254, 0
```

(fără liniile din spațiul de editare).
Evident, pot fi apelate și restul culorilor întrucît $m \in [0; 7]$.

Exemplul 10.1 :

```
1 REM efecte pe BORDER cu instrucțiunea OUT —
  vezi linia 12

2 CLS

3 LET f=0 :LET t=50

10 PAPER 0 :BORDER 0 :INK 7 :CLS :PLOT 2, 8 ;
   DRAW 250, 0 : DRAW 0, 146 :DRAW —250,
   0 :DRAW 0, —146

11 GO TO 40

12 FOR f=0 TO 7 : BEEP .03, t—f : OUT 254,
   f :NEXT f :OUT 254, 1 :OUT 254, 0 :RETURN

40 PLOT 60, 90 :DRAW 0, 50 :DRAW 15, —30 :
   DRAW 15, 30 :DRAW 0, —50

45 LET t=t—5 :GOSUB 12

50 PLOT 110, 90 :DRAW 15,—30 :DRAW 15, 30 :
   DRAW 0, —50

55 LET t=t—5 :GOSUB 12

60 PLOT 160, 90 :DRAW 0, 50 :DRAW 30, 0 :
   DRAW 0, —30 :DRAW —30, 0

65 LET t=t—5 :GOSUB 12

70 PLOT 90, 60 :CIRCLE 96, 90, 2 :PLOT 114,
   90 :CIRCLE 144, 90, 2 : PLOT 168, 90 :CIRCLE
   168, 90, 2

75 LET t=t—5 :GOSUB 12

76 PRINT AT 12, 8 ; " S_O_F_T_W_A_R_E"
   AR 15, 1 ; FLASH 1 ; "ANGRENAJUL" ; AT 15,
   13 ; "CILINDRIC" ; AT 15, 23 ; "EXTERIOR" ;
```

```
REM fiecare literă din acest text se scrie alternînd
TRUE VIDEO cu INVERSE VIDEO

80 FOR t=20 TO 0 STEP -5:GOSUB 12:NEXT t
```

Exemplul 10.2 :

```
8 REM efect de "defilare a culorilor" folosind și
instrucțiunea IN

9 BORDER 1:PAPER 1:INK 0:CLS

10 LET b$="":IF b$="" THEN FOR i=0 TO
31:LET b$=b$+CHR$ 17+CHR$(i-8*INT
(i/8))+"_":NEXT i

20 PRINT # 0; OVER 1; AT 0,4; "APASATI
O TASTA OARECARE"

30 PRINT # 0; AT 1,0; OVER 1; INK 9; b$:IF
IN 254=191 THEN LET b$=b$(4 TO)+b$(TO
3):GO TO 30
```

10-2. INSTRUCȚIUNI PENTRU CĂI ȘI CANALE *(OPEN #, CLOSE #)*

Pentru fiecare echipament periferic sau port de intrare/ieșire este asigurată o linie de comunicație numită *canal*. Fiecărui canal i se asociază o componentă software numită *cale*.

La perifericele standard ale calculatorului (claviatura, TV și casetofonul) se mai pot cupla — folosind interfețe — și alte dispozitive cum sînt :

— *imprimanta* (mașină de scris comandată de calculator) ;
— *discheta* (suport fizic de informație cu o capacitate de stocare echivalentă cu cea a unei casete cu bandă magnetică) ;
— *microdrive* (suport fizic de informație intermediar între casetă și dischetă) ;
— *joystick* (manetă utilizată la derularea jocurilor, care înlocuiește de regulă 5 taste ale calculatorului, servind pentru deplasarea personajului principal al jocului — sus, jos, stînga, dreapta — , precum și pentru declanșarea „focului") ;
— *mouse* (dispozitiv de desenat urmărind trasee pe o hîrtie) ;
— *modem* (dispozitiv care permite cuplarea mai multor calculatoare).

Calculatoarele din familia *ZX SPECTRUM* dispun de trei canale :

— **canalul ecranului** codificat *s* care este *unidirecțional* (trimitere de informații) ;

— **canalul imprimantei** codificat *p*, de asemenea *unidirecțional* (primire de informații) ;

— **canalul tastaturii** codificat *k* care este *bidirecțional* (citirea și scrierea datelor) ; lui îi este atașat și spațiul de editare.

Pentru transmiterea de informații pe un canal oarecare este suficient să se transmită pe calea asigurată acestui canal. La punerea calculatorului în funcțiune se deschid automat căile *0* ... *3* (din cele *16* canale numerotate de la *0* la *15*) cu următoarea asigurare :

— *calea 0* pentru canalul tastaturii *k*
— *calea 1* pentru canalul tastaturii *k*
— *calea 2* pentru canalul ecranului *s*
— *calea 3* pentru canalul imprimantei *p*.

Asigurarea unei căi se face prin instrucțiunea **OPEN #**, iar închiderea unei căi prin instrucțiunea **CLOSE #**.

Sintaxa : **nr. linie OPEN # nr. cale, "tip"**

unde *nr. cale* ∈ *[0 ; 15]* și *tip* = {*s, k, p*}

Exemplu : **10 OPEN # 5, "k" :PRINT # 5; AT 1, 0;**
"HC-85"

Efect : deschide calea cu numărul precizat pentru un anumit canal

Sintaxa : **nr. linie CLOSE # nr. cale**
unde *nr. cale* ∈ *[0 ; 15[*

Exemplu : **40 CLOSE #5**

Efect : închide calea cu numărul precizat pentru un anumit canal.

Dacă *nr. cale* ⩾ *16* survine mesajul de eroare

0 Invalid stream, nr. linie ; nr. instrucțiune

Intenția de a închide o cale nedeschisă se soldează cu crahul sistemului.

INSTRUCȚIUNI/COMENZI PENTRU LUCRUL CU IMPRIMANTA (*LLIST, LPRINT, COPY*)

Sintaxa : **LLST [nr. linie]**

unde *nr. linie* este opțional

Exemplu : **LLIST 10**

LLIST

Listează la imprimantă programul de la numărul de linie precizat, sau de la prima linie dacă eticheta lipsește. Spre deosebire de comanda **LIST**, comanda **LLIST** nu afișează **Ok?**. Listarea poate fi oprită cu **BREAK**.

Sintaxa : **LPRINT [listă]**

Exemplu : **LPRINT a, b, c, d**

LPRINT

Efect : tipărește la imprimantă datele și rezultatele dorite; formatul general al instrucțiunii/comenzii **LPRINT** este același cu cel al instrucțiunii **PRINT**

Sintaxa : **COPY**

Exemplu : **COPY**

Efect : listează la imprimantă o copie a primelor *22* linii ale ecranului

Cele trei instrucțiuni sînt operante numai dacă ordinatorul este cuplat cu o imprimantă.

10.4. COMENZILE PENTRU MICRODRIVE ȘI DISCHETĂ *(FORMAT, CAT, ERASE, MOVE)*

După cum s-a menționat, microdrive-ul și discheta sînt suporturi fizice de mare viteză pentru stocarea programelor. Comenzile folosite sînt :

FORMAT — formatează un cartus microdrive sau o dischetă

CAT — listează fișierele microdrive-ului sau dischetei

ERASE — șterge un fișier

MOVE — transferă un fișier.

Capitolul 11

ARTIFICII PENTRU PERFECȚIONAREA ȘI PROTEJAREA PROGRAMELOR

Artificiile sînt modalități ingenioase de programare prin care programele în BASIC capătă un plus de atractivitate sau sînt protejate împotriva listării sau însușirii.

11.1. ARTIFICII PENTRU PERFECȚIONAREA PROGRAMELOR

11.1.1. Modalități de scriere

Programele care urmează sugerează o serie de modalități de scriere a unor texte (mesaje, enunțuri, concluzii, etc.).

Exemplul 11.1 :

```
 8 REM scriere de jos în sus
10 CLS :BORDER 0 :PAPER 0 :INK 7 :CLS
20 PRINT AT 21, 3; "MECANISMELE CU CAME
   PLANE" :GOSUB 100
30 PRINT AT 21, 0; "SINT MECANISME TRI-
   LATERE AVIND :" GOSUB 100
40 PRINT AT 21, 2; "—O CUPLA SUPERIOARA ;"
   :GOSUB 100
50 PRINT AT 21, 2; "—2 CUPLE INFERIOARE
   (R SAU T)." :GOSUB 100
60 STOP
100 PAUSE 40 : RANDOMIZE USR 3582 :RETURN :
    REM rutina de ridicare cu un rînd
```

Condiția scrierii de jos în sus este de a introduce într-o instrucțiune **PRINT AT** 21, **coloană** un mesaj de maximum *32* caractere, după care execuția programului este trimisă la linia *100*.

Exemplul 11.2 :

```
 8 REM scriere cu litere "rotite"
10 FOR f=7 TO 0 STEP —1 :POKE 23606, f;
```

```
PRINT AT 1, 3; "SINTEZA MECANISMELOR
PLANE" :BEEP .009, f :NEXT f
```

Exemplul 11.3 :

```
8 REM scriere gen "mașină de scris"
10 CLS :LET a$=" _ _ _ _ _ _ PROGRAME
INFORMATICE _ _ _ _ _ _ " :REM în va-
riabila șir a$ mesajul trebuie să aibă maximum
32 caractere
20 LET lin=5 :LET col=0 :GOSUB 170 :REM lin,
col reprezintă linia și coloana unde urmează
a se tipări mesajul
30 STOP
170 FOR n=32 TO 1 STEP -1 :PRINT AT lin,
col; a$(n TO) :BEEP .001, 60 :NEXT n :BEEP
.1, 0 :BEEP .1, 5 :RETURN
```

Exemplul 11.4 :

```
8 REM scriere gen "șenilă"
10 CLS :LET a$="_Artificii pt. programe în BASIC"
:REM variabila șir a$ poate conține maximum
32 caractere
20 GOSUB 100 :STOP
100 FOR w=1 TO 31 :PRINT AT 8, w-1; a$(w) :
BEEP .005, 10 :NEXT w :RETURN
```

Exemplul 11.5 :

```
8 REM scriere cu INK în schimbare
9 BORDER 0 : PAPER 0 :INK 7 :CLS
10 FOR x=1 TO 6 :PAUSE 25 :PRINT AT 3, 3;
INK x+1 : BRIGHT 1; FLASH 1; "M. M.
POPOVICI SOFTWARE 1992" : NEXT x :PAU-
SE 100 :BEEP 1, 46
```

Exemplul 11.6 :

```
8 REM scriere simultană pe trei rînduri
10 BORDER 0 :PAPER 0 :INK 7 :CLS
20 LET a$="Programarea în limbajul BASIC _
_ _ " :LET b$="este acum accesibilă tuturor
_ _ _ _ _ " LET c$="celor care au studiat
cartea _ _ _ _ _ " :GOSUB 100
30 STOP
100 FOR n=1 TO 32 :PRINT 0, 0; a$(TO n) ; AT 1,
0; b$(TO n) ; AT 2, 0; c$(TO n) :NEXT n :
RETURN
```

Exemplul 11.7 :

```
8 REM scriere sub formă de "ploaie de litere" pe
o porțiune a ecranului
10 BORDER 7 :PAPER 7 :INK 0 :CLS
```

```
20 PRINT AT 10,0; INK 7; "_ PANA INCLINA-
   TA" ' "este un restrictor" ' "mecanic torsional"
   ' "de contact și de" ' "frecare care" ' "transmite
   Mt prin" ' "frecare"

30 OVER 1 :INK 0 :FOR i=1 TO 128 :PRINT AT
   10+INT i-7*INT(i/7), (5*i)-18*INT(5*i/18);
   "_" :NEXT i
```

Exemplul 11.8 :

```
8 REM "ploaie de litere" sub un dreptunghi crescător

9 BORDER 0 :PAPER 0 :INK 7 :CLS

10 BORDER 0 :INK 0 :PRINT AT 6, 1, "Cercul de
   informatică și-a propus să realizeze mai multe
   sco-_puri și anume: 48 blancuri-invatarea
   limbajului BASIC; 67 blancuri-elaborarea de
   programe de in-_ _ _struire asistată_; 44
   blancuri-inițiere în cod mașină"

20 OVER 1 :INK 7 :FOR i=87 TO 0 STEP -8 :FOR
   j=1 TO 2 :PLOT i, i :DRAW 255-2*i, 0 :
   DRAW 0, 175-2*i :DRAW 2*i-255, 0 :
   DRAW 0, 2*i-175 :NEXT j :NEXT i :OVER 0
```

Exemplul 11.9 :

```
8 REM scriere simultană stînga-dreapta pe același
  rînd

10 CLS :PRINT INK 7; "text 32 caractere" :GOSUB
   9000 :STOP
   9000 OVER 1 :FOR j=0 TO 7 :INK 1 :FOR i=0
   TO 15 :PRINT AT j, i; "_"; AT j, 31-i;
   "_" :NEXT i :NEXT j :RETURN
```

Exemplul 11.10 :

```
8 REM scriere simultană către stînga pe întregul
  ecran

10 INK 7 :PRINT AT 9, 0; "Să ne imaginam că am
   realizat o serie de programe educaționale
   pentru majoritatea discipline-_ _lor de învăță-
   mînt pentru care elevii au dorit să lucreze".

20 INK 1 :FOR i=31 TO 0 STEP -1 :FOR j=0
   TO 21 :PRINT OVER 1; AT j, i; "_" :NEXT
   j :NEXT i
```

Exemplul 11.11 :

```
8 REM scriere cu cursor (varianta 1)

10 RESTORE 10 :PRINT AT 10, 1 :DATA "_ _
   _ _ _ _ _ TEMA DE PROIECTARE", "_Să se
   proiecteze un angrenaj", "_cilindric exterior cu
   dinți", "_drepți cunoscînd :" :GOSUB 100 :
   GOSUB 100 :GOSUB 100 :GOSUB 100 :STOP :
   REM atîtea trimiteri la linia 100 cîte șiruri
   DATA sînt
```

```
100 LET q$=CHR$8 :FOR w=1 TO 4 :LET q$=
    q$+q$ :NEXT w :LET u=1 :OVER u :PRINT :
    READ a$ :FOR i=1 TO LEN a$ :PRINT CHR$8;
    "["; a$(i);"]"; q$(TO 3); :BEEP .01, -20 :
    PRINT" [ _ ]"; CHR$8; :BEEP .01, -10

110 LET tab=31 :IF 33-PEEK 23688>tab THEN
    PRINT

120 NEXT i :OVER 0 :RETURN
```

Exemplul 11.12 :

```
8 REM scriere cu cursor (varianta 2)

10 LET r=660 :PRINT AT 2, 2; :LET a$=" _ _
   M. M POPOVICI SOFTWARE 1992_ "+CHR$
   13+"_Modalitate de scriere cu cursor" :GOSUB
   r :STOP

660 INK 9 :FOR i=1 TO LEN a$

665 IF a$(i)=CHR$13 THEN PRINT a$(i); :NEXT i

670 FOR k=0 TO 4 :PRINT PAPER k; a$(i);
    CHR$8; :NEXT k :PRINT a$(i); :NEXT i :
    RETURN
```

Exemplul 11.13 :

```
8 REM scriere în spirală

10 BORDER 0 :PAPER 0 :INK 7 :CLS

20 FOR x=0 TO 5 STEP .3 :LET t=15+5*SIN
   x :PRINT TAB t; "M. M. POPOVICI" :POKE
   23692, -1 :NEXT x
```

Exemplul 11.14 :

```
8 REM deplasarea unui text (cuvînt) spre stînga

10 OVER 0 :FOR j = 10 TO 0 STEP -1 : PRINT
   AT 0, 0 :PAPER 5; INK 1; "1. CARACTERI-
   ZARE"; AT 1, j; PAPER 7; INK 0; "1. 1_
   Definiții_ _ _ _ _ _"; :PAUSE 1 :NEXT j
```

Exemplul 11.15 :

```
8 REM scriere cu fond care se colorează

9 BORDER 7 :PAPER 7 :INK 0 :CLS

10 DIM d$(128) :PRINT AT 0, 0; d$(TO 86) :INK
   7 :PRINT AT 0, 2; "_IATA UN EFECT INTE-
   RESANT DE_ _ _ SCRIERE CARE POATE
   FI UTILIZAT_ _ IN PROGRAMELE DUM-
   NEAVOASTRA" ';TAB 3; "SPER CA VA VA
   PLACE!!..."

20 PAUSE 50

30 FOR i=3 TO 0 STEP -1 :PAPER 3 :INK
   2*(i<=3) :PRINT AT i, 0; OVER 1; d$(TO
   32) :NEXT i
```

Exemplul 11.16 :

```
8 REM titluri cu litere care se colorează
10 BORDER 2:PAPER 0:INK 7:CLS
20 PRINT AT 0,0; "max. 32 caractere":OVER
1:POKE 23659, 2:LET a$="_":FOR i=0
TO 31:LET a$=a$+CHR$ 16+CHR$ (i-7*
INT(i/7))+"_":NEXT i:FLASH 8:FOR i=0
TO 32:BEEP .01, -10:BEEP .01, -10:
PRINT AT 0,0; a$(3*i+2 TO); a$(TO 3*i):
NEXT i:FLASH 0:OVER 0
```

Exemplul 11.17 :

```
8 REM ecran cu INK în colorare
10 OVER 1:PRINT AT 0,0; INK 7:FOR i=32
TO 255:PRINT CHR$i; :NEXT i
20 DIM d$(320):OVER 1:FOR i=0 TO 7::PRINT
AT 0,0; INK i; d$; INK i+1; d$; d$(TO 32):
PAUSE 10:NEXT i:OVER 0
```

Exemplul 11.18 :

```
8 REM text dispus pe ecran în formă de "dublu W"
10 INK 7:PRINT AT 1,4; "Vom da un exemplu :"
' ""_ _Să se calculeze viteza" ' "_unghiulară,
cea liniară," ' "_accelerația și perioada" ' " de
_rotație a unui punct" ' "_de pe paralela 45"
20 LET z=0:LET u=1:LET x=u:LET y=u:
LET a=u:LET b=u
30 INK z:OVER u:FOR i=u TO 200
40 IF x=24 OR x=z THEN LET a=-a
50 IF y=7 OR y=z THEN LET b=-b
60 LET x=x+a:LET y=y+b:PRINT AT y, x;
"_ _":NEXT i:INK z
```

Exemplul 11.19 :

```
8 REM "defilarea culorilor" pe titluri
10 PRINT AT 0,0; "max. 32 caractere"
20 INK 9:LET a$=" ":FOR j=0 TO 3:FOR i=7
TO 0 STEP -1:LET a$=a$+CHR$ 17+
CHR$ i+CHR$ 32:NEXT i:NEXT j
30 FOR k=0 TO 50:PRINT OVER 1; AT 0,0;
a$:LET a$=a$(94 TO)+a$(TO 93):NEXT k
40 PRINT OVER 0; AT 0,0; INK 9; TAB 31;
"_"
```

Exemplul 11.20 :

```
8 REM schimbarea culorilor caracterelor pe întregul ecran
10 POKE 23560, 255:PAPER 8:OVER 1
```

```
20 FOR i=32 TO 255 :PRINT CHR$ i; :NEXT i
30 FOR w=1 TO 6
40 FOR i=0 TO 5 :PRINT INK i; AT 0, 0; TAB
   31; TAB 30; "_ _"; AT 8, 12; TAB 18;
   AT 10, 7; TAB 24; AT 14, 0; TAB 31; TAB
   30; AT 18, 12; TAB 18; AT 20, 7; TAB 24
   :NEXT i
50 PAUSE 20 :FOR i=0 TO 5 :PRINT INK i;
   AT 0, 0; TAB 31; TAB 30; "_ _"; AT 8,
   12; TAB 30; AT 18, 12; TAB 18; AT 20, 7;
   TAB 24 :NEXT i
60 IF PEEK 23560=255 THEN NEXT w
```

Exemplul 11.21 :

```
8 REM "defilarea culorilor" pe textul "Apasă o
  tastă oarecare"
10 BORDER 2 :PAPER 2 :CLS
20 LET b$=" " :IF b$=" "THEN FOR i=0 TO
   81 : LET b$=b$+CHR$ 17+CHR$(i-8*INT
   (i/8))+"_" :NEXT i :POKE 23659, 1 :PRINT
   # 0; "_ _ _ _ _ APASA O TASTA OARE-
   CARE_ _ _ _ _" :POKE 23659, 2
30 PRINT # 0; AT 1, 0; OVER 1; INK 9; b$ :IF
   IN 254=191 THEN LET b$=b$(4 TO)+b$
   (TO 3) :GO TO 30
40 OVER 0 :PRINT # 0; AT 1, 0; TAB 31; "_"
```

Exemplul 11.22 :

```
8 REM modalitate de afișare a mesajului "APA-
  SATI O TASTA OARECARE"
10 LET z=NOT PI :LET u=NOT z :LET q$="_
   _ _ _APASATI O TASTA OARECARE_ _
   _ _ _":POKE 23659, u :FOR i=u TO 32:
   :PRINT AT 21, i-u; q$(i) :BEEP .05, -30-
   i/2 :NEXT i :POKE 23659, 2
```

11.1.2. Cortine și modalități de ștergere a ecranului

Exemplul 11.23 :

```
8 REM cortină stînga-dreapta
10 FOR f=32 TO 255 :PRINT CHR$ i; :NEXT f
   :PAUSE 40
20 INK 7 :FOR i=0 TO 255 STEP 8 :PLOT i, 0
   :DRAW 0, 175 :NEXT i :INK 0
```

Exemplul 11.24 :

```
8 REM ștergerea ecranului în diagonală
10 FOR i=32 TO 255 :PRINT CHR$ i; :NEXT i
   :PAUSE 50
```

```
20 INK 7:FOR i=0 TO 255 STEP 8:PLOT i,
0:DRAW -i, i*175/255:NEXT i:FOR i=0
TO 255 STEP 8:PLOT i, 175:DRAW 255-i,
(i-255)*175/255:NEXT i:INK 9
```

Exemplul 11.25 :

```
8 REM cortină de jos în sus
10 FOR f=32 TO 255:PRINT CHR$ i; :NEXT i
:PAUSE 50
20 FOR i=21 TO 0 STEP -1:PRINT AT i, 0;
TAB 31; "_":NEXT i
```

Dacă se dorește o cortină de sus în jos, linia *20* se modifică astfel :

```
20 FOR i=0 TO 21:PRINT AT i, 0; TAB 31;
"_":NEXT i
```

Exemplul 11.26 :

```
8 REM cortină stînga-dreapta cu întîlnire la
mijlocul ecranului
10 FOR i=32 TO 255:PRINT CHR$ i; :NEXT i
:PAUSE 50
20 FOR i=0 TO 127 STEP 8:PLOT i, 0:DRAW
INK 7; PAPER 7; 0, 175:PLOT 255-i, 0:
DRAW INK 7; PAPER 7; 0, 175:NEXT i
```

Exemplul 11.27 :

```
8 REM ștergerea ecranului în etape succesive
10 FOR f=32 TO 255:PRINT CHR$ i; :NEXT i
:PAUSE 50
20 LET r=RND:IF r<.8 THEN FOR f=64 TO
87:BEEP .006, 80-f:POKE 23681, f:LPRINT
"32 blancuri":NEXT f
```

11.1.3. Modalități de operare cu ecranul

Un programator cu experiență și gust în efectuarea de programe în *BASIC* va căuta să dea o înfățișare cît mai atractivă ecranului care va afișa conținutul programului. De asemenea, se va îngriji ca mirele desenate și salvate ca *SCREEN*-uri să fie aduse instantaneu pe ecran în momentele în care ele sînt necesare. Programele care urmează oferă cîteva soluții.

Exemplul 11.28 :

```
8 REM "pagină" de ecran cu schimbarea paperului
10 BORDER 7:PAPER 7:INK 0:CLS
20 LET zz=22:LET zl=32
30 FOR x=0 TO zz-1:PRINT AT x, 0; "#";
AT x, zl-1; "#":NEXT x:FOR y=0 TO
zl-1:PRINT AT 0, y; "#"; AT zz-1, y;
"#": NEXT y:REM contur
40 PRINT AT 0, 0:FOR i=1 TO 20:PRINT PAPER
5; AT i, 1; OVER 1-(i=10); TAB 31:PAUSE
5:NEXT i
```

```
50 PRINT AT 10, 9; "CURS DE BASIC"

60 DIM a$(704) :INK 7 :OVER 1 :FOR i=1 TO 7
   :POKE 23624, i*8 :INPUT " " :BORDER i :
   PAPER i :PRINT AT 0, 0; a$ :NEXT i
```

Exemplul 11.29 :

```
8 REM model de "pagina de ecran"

10 BORDER 5 :PAPER 6 :INK 2 :CLS :PLOT 15,
   0 :DRAW 255, 0 :DRAW 15, 15 :DRAW 0,
   145 :DRAW -15, 15 :DRAW -225, 0 :DRAW
   -15, -15 :DRAW 0,-145 :DRAW 15,-15 :
   DRAW 0, 15 :DRAW -15, 0

20 PLOT 0, -145 :DRAW 15, -15 :DRAW 0,
   15 :DRAW -15, 0 :PLOT 255, 15 :DRAW -15,
   0 :DRAW 0, -15 :PLOT 255, 160 :DRAW -15,
   0 :DRAW 0, 15 :PLOT 0, 160 :DRAW 15, 0 :
   DRAW 0, 15

30 PRINT AT 0, 0; OVER 1; PAPER 5; "_ _";
   AT 0, 30; "_ _"; AT 1, 0; "_ _"; AT 1,
   30; "_ _"; AT 20, 0; "_ _"; AT 20, 30;
   "_ _"; AT 21, 0; "_ _"; AT 21, 30; "_ _"
```

Exemplul 11.30 : aducerea instantanee a unei mire stocate în memorie nu se poate face decît cu ajutorul unui program în *cod mașină* (în acest scop valorile necesare sînt introduse într-o linie DATA)

```
8 CLEAR 57999 :LET adr=58000 :REM adresa
  locației de memorie la care se încarcă SCREEN-ul

10 RESTORE 20 :FOR i=0 TO 11 :READ a :POKE
   adr+i, a :NEXT i

20 DATA 33, adr-INT(adr/256)*256, INT(adr/
   256), 17, 0, 64, 1, 0, 27, 237, 176, 201

30 RANDOMIZE USR adr
```

În prealabil *mira* este încărcată la adresa *adr* cu comanda

```
LOAD "nume" CODE adr, 6912
```

Rutina este relocabilă (adică se poate folosi o altă adresă în locul valorii *58000*).

11.2. MODALITĂȚI DE PROTECȚIE A PROGRAMELOR

Se oferă cîteva variante de protecție a programelor care pot fi folosite separat sau împreună.

Exemplul 11.31 : scrierea numelui autorului într-o linie fără etichetă.

a) Se tastează instrucțiunea după modelul următor

```
1 REM_ _ _ _ _ _ _ _ M. M. POPOVICI SOFTWARE 1992
```

b) Se tastează comenzile :

POKE 23756, 0 :POKE 23760, 21 :POKE 23761, 0 :POKE 23762, 22

POKE 23763, 0 :POKE 23674, 0

Exemplul 11.32 : modificarea atributelor afișării (linii invizibile)
a) Se editează linia (cu *CS* și *1*), după care se tastează

CS cu **SS** și tasta de culoare (*7* sau *0*), **CR**

b) Se editează linia din nou (*CS* și *1*), după care se tastează *CS* cu *8* și o tastă oarecare (ex. : *w*) ; calculatorul *„bîzîie"* ; **CR**.

La comanda **LIST** apare mesajul *Invalid Coulour*, dar programul rulează.

Exemplul 11.33 : realizarea unui șir succesiv de linii *0*.

De pildă la programul următor toate liniile sale vor fi etichetate cu eticheta *0 :*

```
10 PRINT AT 13, 11 ; "Bună ziua !"
20 LAT a$="Mă numesc" :LET b$="_M. M. POPOVICI"
30 LET c$=a$+b$
40 PRINT AT 14, 4 ; c$
50 GO TO 100
60 STOP
100 DEF FN p( )=PEEK 23636+256*PEEK 23636 :DEF FN
    l(x)=256*PEEK x+PEEK (x+1) :LET p=FN p( )
110 LET l=FN l(p)
120 IF l>=60 THEN STOP :REM cifra 60 este numărul de linie de
    la care eticheta nu se mai modifică în valoarea 0
130 POKE p, 0 :POKE (p+1), 0
140 LET p=p+4+PEEK (p+2)+256*(PEEK (p+3))
150 GO TO 110
```

Exemplul 11.34 : listing invizibil.

Se introduce în prima linie a programului *BASIC* una din instrucțiunile :

POKE 23743, 80 (se revine cu POKE **23743, 83**) sau

POKE 23755, 100 (se revine cu POKE **23755, 0**) sau

POKE PEEK 23635+256*PEEK 23636, 100

Exemplul 11.35 : scrierea unei linii cu eticheta *0* către sfîrșitul programului *BASIC* care afișează numele autorului. Instrucțiunea care urmează se scrie *înaintea liniei la care se dorește eticheta 0 :*

```
7499 LET A=PEEK 23637+256*PEEK 23638 :POKE A, 0 :POKE
    A+1, 0 :STOP
```

7500 PRINT # 0; AT 0, 5; "M. M. POPOVICI SOFTWARE!" :PAUSE 0

Se dă comanda **RUN 7499** și se constată că linia *7500* are acum eticheta *0*; în continuare se *șterge linia 7499*.

Exemplul 11.36 : mascarea datelor.

Se presupune instrucțiunea

10 PRINT 12345

pentru care se dorește ca la comanda **LIST** să apară

10 PRINT 54321

În acest scop se procedează astfel :

a) Se tastează : **CLS :FOR n=23755 TO 25000 :PRINT n; "="; PEEK n, CHR$ PEEK n AND PEEK>31 :NEXT n**

b) Calculatorul afișează :

23755 = *0*
23756 = *10* (numărul liniei)
23757 = *13* (lungimea în octeți a liniei)
23758 = *0*
23759 = *245* (adică **PRINT** — v. tab. 3.1)
23760 = *49* (adică *1* — idem)
23761 = *50* (adică *2*)
23762 = *51* (adică *3*)
23763 = *52* (adică *4*)
23764 = *53* (adică *5*)
23765 = *14*
23766 = *0*
23767 = *0*
23768 = *58* (adică octetul *L* al nr. *12345*)
23769 = *47* (adică octetul *H* al numărului)
23770 = *0*
23771 = *13* (adică *ENTER*) — sfîrșitul liniei *10*

c) Se tastează :

POKE 23760, 53 (adică *5*)

POKE 23761, 52 (adică *4*)

POKE 23762, 51 (rămîne reprezentînd cifra *3*)

POKE 23763, 50 (adică *2*)

POKE 23764, 49 (adică *1*)

d) Se tastează **RUN** și calculatorul afișează corect *12345*, dar la comanda **LIST** linia *10* apare modificată : **10 PRINT 54321.**

Exemplul 11.37 : protecție anti MERGE.

Procedeul de lucru este următorul :

a) SE scrie linia : **9999 REM ***

b) Se dă comanda: **PRINT PEEK 23627 + 256 * PEEK 23628.** Calculatorul va afișa adresa *„adr"* pentru ultimul octet al programului *BASIC*.

c) Se dă comanda: **FOR f=(adr−10) TO adr; PRINT f; "="; PEEK f; INVERSE 1; CHR$ PEEK f AND PEEK>31 :NEXT f**

Calculatorul afișează adresele și valorile din locațiile de memorie; se va nota adresa *„x"* unde se găsește **REM**.

d) Se dă comanda: **POKE (x−2), 255 :POKE (x−1), 255.**
e) Se salvează programul (care va fi protejat la **MERGE**) cu comanda

SAVE "nume" LINE 0 :VERIFY " "

11.3. ALTE MODALITĂȚI DE PERFECȚIONARE A PROGRAMELOR FOLOSIND RUTINE ÎN COD MAȘINĂ

Exemplul 11.38 : introducerea mascată a unei rutine în cod mașină în linia *0* (sau *1*) a programului BASIC.
a) Se scrie instrucțiunea

1 REM șir de litere a egale numeric cu șirul valorilor DATA

(ex: **1 REM 25 litere a**); eventual se tastează **POKE 23756, 0** pentru a face schimbarea etichetei din cifra *1* în cifra *0*.
b) Se scrie linia pentru introducerea celor *x* valori **DATA**:

10 FOR f=23760 TO (23760+x−1) :INPUT (f), i; POKE f, i; PRINT f, i :NEXT f :REM x=25

c) Se dă comanda **RUN 10** și se tastează valorile **DATA**. De exemplu:

6, 12, 197, 33, 200, 0, 17, 20, 0, 205, 181, 3, 33, 200, 0, 17, 40, 0, 205, 181, 3, 193, 16, 234, 201

d) Se tastează *10* pentru a se șterge linia *10* și se înlocuiește cu

10 RANDOMIZE USR 23760

În exemplul dat se va auzi sunetul telefonului. Se precizează că literele *a* din instrucțiunea **REM** vor fi înlocuite cu o serie de comenzi *BASIC*.

Exemplul 11.39 : reintroducerea caracterelor grafice definite de utilizator în linia *0*.
a) Se scrie linia

1 REM șir de litere a egal numeric cu 8*nr. caracterelor

b) Se scriu liniile:

2 FOR f=23760 TO (23760+8*nr. caractere−1) :READ a :POKE f, a :NEXT f

3 DATA șir de 8 valori numerice pentru fiecare caracter

c) Se dă comanda **RUN** și se șterg liniile *2* și *3*. Se apelează caracterele grafice cu instrucțiunile:

```
10 FOR f=0 TO (8*nr. caractere-1) :POKE USR "a"+f,
   PEEK (23760+f) :NEXT f
```

Caracterele nou definite se *tastează în modul grafic.*

Exemplul 11.40 : realizarea de blocuri cod mașină fără header. Se știe că organizarea programului pe bandă magnetică este formată dintr-o porțiune de *17* octeți numită *header* (care conține informații privind titlul, tipul, adresa și lungimea programului) și programul propriu zis. Pentru rutinele în cod mașină este posibilă o măsură de protecție care constă în *suprimarea header-ului* așa cum se indică prin programul următor.

```
5 LET adr=adresa unde se va amplasa rutina "fără header"

10 FOR f=0 TO 15 :READ a :POKE adr+f, a :NEXT f

20 DATA 243, 17, 240, 254, 221, 33, 0, 1, 62, 255, 55, 205, 86, 5,
   243, 201 :REM rutina "fără header"

30 LET c=lungimea în octeți a codului mașină ce se dorește încărcat
   fără nume (header) :LET b=adresa unde este salvat acest cod
   mașină

40 POKE (adr+2), c-256*INT(c/256) :POKE (adr+3), INT(c/256)
   :REM în locațiile (adr+2) și (adr+3) se introduce lungimea c

50 POKE (adr+6), b-256*INT(b/256) :POKE (adr+7), INT(b/256)
   :REM în locațiile (adr+6) și (adr+7) se introduce adresa b

60 POKE (adr+14), 201 :REM revenire în BASIC

70 RANDOMIZE USR adr
```

Evident că după acest program, rutina în cod mașină se va salva pe bandă *fără header.* (de exemplu : dacă rutina în cod mașină este la adresa *50000* și are lungimea de *1000* octeți, atunci în programul anterior *b = 50000* și *c = 1000*; rutina „fără header" se va amplasa la o adresă diferită, de pildă adr = *55000*).

Capitolul 12

ALCĂTUIREA ȘI COPIEREA PROGRAMELOR COMPLEXE

12.1. STRUCTURA UNUI PROGRAM PE CALCULATOR

12.1.1. Programe loader

Un program pe calculator poate avea una din următoarele două structuri principale:

a) *Programe scrise numai în BASIC*; acestea se salvează:

— fie direct cu comenzile

SAVE "nume" [LINE x] : VERIFY" " unde *x* — linia de autolansare a programului în execuție;

— fie introducînd în program linia

99) SAVE "nume" [LINE x] : VERIFY" ".

b) *Programe complexe* formate din:

— un program ce apelează restul programelor numit *loader*;
— screen-ul de prezentare al programului (titlul, autorul, un desen care sugerează tematica programului, etc.);
— unul sau mai multe blocuri în cod mașină;
— programul *BASIC* propriu zis.

Aceste programe se salvează rînd pe rînd și anume:

— pentru *loader* și *programul propriu zis în BASIC* cu comenzile

SAVE "loader" LINE x: VERIFY" "

SAVE "nume" LINE x: VERIFY" "

— pentru *screen*:

SAVE „nume" SCREEN$ sau **SAVE "nume" CODE adr, 6912** (dacă se dorește aducerea lui instantanee pe ecran în timpul rulării programului BASIC);

— pentru blocurile în *cod mașină*:

SAVE "nume" CODE adresă, lungime

adresă — adresa codului mașină; *lungime* — lungimea în octeți a codului respectiv)

Se prezintă cîteva modele de *programe loader*.

Exemplul 12.1 :

```
10 PAPER 1 :CLS :BRIGHT 1 :BORDER 5 :INK 7 :
   PLOT 0, 0 :DRAW 255, 0 :DRAW 0, 175 :DRAW
   -255, 0 :DRAW 0, -175

20 INK 6 :PLOT 29, 6 :DRAW 197, 0 :DRAW 0,
   27 :DRAW -197, 0 :DRAW 0, -27

30 PRINT PAPER 2 ; INK 7 ; AT 18, 5 ; "PROGRA-
   ME PE CALCULATOR" ; INK 2 ; PAPER 5 ;
   FLASH 1 ; AT 19, 5 ; "M. M. POPOVICI SOFT-
   WARE ©" ; INK 1 :PRINT PAPER 7 ; AT 20, 7 ;
   "INITIERE IN BASIC"

40 POKE 23739, 111 :LOAD" "
```

Exemplul 12.2 :

```
1 BORDER 2 :PAPER 0 : NK 7 :CLS

2 FOR f=40 TO 30 STEP -2 :PLOT f-12, f-15 :
  DRAW 278-2 * f, 0 :DRAW 0, 123 :DRAW -278
  +2 * f, 0 :DRAW 0, 132 :DRAW -278+2 * f,
  0 :DRAW 0, -132 :NEXT f

3 BRIGHT 1 :PRINT AT 16, 4 ; PAPER 5 ; INK 1 ;
  "M. M. POPOVICI SOFTWARE ©"

4 PRINT AT 11, 7 ; FLASH 1 ; PAPER 6 ; INK 4 ;
  "FORTA LUI LØRENTZ"

5 POKE 23739, 111 :LOAD " "
```

Exemplul 12.3 :

```
1 PAPER 2 :BORDER 2 :CLS :INK 6 :PLOT 19,
  132 :DRAW 216, 0 :DRAW 0, -88 :DRAW -216,
  0 :DRAW 0, 88 :PLOT 20, 131 :DRAW 214, 0 :
  DRAW 0, -86 :DRAW -214, 0 :DRAW 0, 86

2 PRINT PAPER 7 ; INK 1 ; AT 5, 6 ; "21 blancuri" ;
  AT 16, 5 ; "M. M. POPOVICI SOFTWARE" :
  PRINT INK 6 ; AT 9, 9 ; FLASH 1 ; "M_E_C_
  A_N_I_C_A" ; FLASH 0 ; AT 11, 9 ; "PRO-
  GRAME PENTRU" ; AT 12, 6 ; "CINEMATICA
  RIGIDULUI" :PRINT FLASH 1 ; AT 20, 5 ; "_
  ASTEPTATI_" ; AT 20, 16 ; INVERSE 1 ; "_
  INCARCAREA_"

3 PRINT AT 0, 0 ; :LOAD" "
```

Exemplul 12.4 :

```
3 BORDER 1 :PAPER 1 :INK 6 :CLS

4 FOR f=64 TO 71 :POKE 23681, f :LPRINT TAB
  5 ; "M. M. POPOVICI SOFTWARE" :NEXT f
  :FOR f=80 TO 87 :POKE 23681, f :LPRINT
  TAB 5 ; "M. M. POPOVICI SOFTWARE"
  :NEXT f

5 FOR j=1 TO 5 :FOR k=0 TO 255 STEP 32 :
  FOR i=72 TO 79 :POKE 23681, i :POKE 23680,
```

```
k :LPRINT TAB 6; "CURSA DE FORMULA 1 !"
:NEXT i :NEXT k :NEXT j
6 POKE 23739, 111 :LOAD" "SCREEN$ :LOAD" "
```

Exemplul 12.5 :

```
10 BORDER 1 :PAPER 0 :INK 0 :CLS
20 FOR n=7 TO 38 STEP 8
30 FOR m=7 TO 0 STEP -1
40 PAPER m :PRINT AT 0, (n-m); "_"; AT 21,
   31-(n-m); "_"; AT 1, (n-m); "_"; AT 20,
   31-(n-m); "_"; AT 2, (n-m); "_"; AT 19,
   31-(n-m); "_"
50 NEXT m
60 NEXT n
70 BRIGHT 1 :PAPER 7 :INK 7 :PRINT AT 3, 0;
   "32 blancuri"; AT 18, 0; "32 blancuri"
80 PRINT AT 9, 3; PAPER 5; INK 1; "M. M.
   POPOVICI SOFTWARE_1992"; AT 14, 10;
   PAPER 6; INK 4; INVERSE 1; FLASH 1;
   "SE INCARCA_ !" :FLASH 0
90 POKE 23739, 111 :LOAD " "SCREEN$ :LOAD
   " UDG " CODE 65368, 168 : LOAD" "
```

12.1.2. Modalităţi de economisire a memoriei

Pentru a afla cîţi octeţi mai sînt disponibili se recomandă ca în programul *BASIC* să se introducă linia

```
9999 PRINT "Inca_"; PEEK 23730+256*PEEK 23731-PEEK
23653-256*PEEK 23654; "octeţi disponibili".
```

La scrierea programelor lungi, existînd pericolul lipsei de memorie disponibilă, este necesar să se ia măsuri de economisire cum sînt cele prezentate în continuare. Se precizează că fiecare linie de program are nevoie de *5* octeţi în afară de conţinutul liniei şi anume : *2* octeţi pentru *numărul liniei*, *2* octeţi pentru *lungimea liniei* şi *1* octet pentru *sfîrşitul liniei*. Este evident că dacă se reunesc două linii de program într-una singură se face o economie de *5* octeţi. De exemplu, în locul liniilor

```
10 PRINT "Numărul încercărilor nereuşite :" ; g
20 PRINT AT 5, 0; "Numărul avioanelor : "; a
```

este recomandabilă scrierea unei linii comune :

```
10 PRINT "Numărul încercărilor nereuşite :" ; g ; AT 5, 0 ; "Numărul
avioanelor :" ; a
```

Pe lîngă utilizarea *liniilor multiple de program* se pot folosi modalităţile prezentate în cele ce urmează.

a) *Folosirea instrucțiuni AND*

Știind că instrucțiunile din linia următoare sînt tratate numai dacă $t = 0$

```
10 PRINT "Bomba explodează !" AND t=0
```

se sugerează utilizarea instrucțiunii **AND** așa cum se indică în exemplificarea de mai jos : În locul programului

```
10 PRINT "Aruncarea ta a fost";
20 IF w=1 THEN PRINT "prima";
30 IF w=2 THEN PRINT "a doua";
40 IF w=3 THEN PRINT "a treia";
50 IF w=4 THEN PRINT "a patra";
60 IF w=5 THEN PRINT "a cincea";
70 IF w=6 THEN PRINT "a șasea";
80 PRINT "."
90 IF w<4 THEN PRINT "FOARTE BINE !"
```

se poate folosi forma :

```
10 PRINT "Aruncarea ta a fost_";"prima" AND w=1;"a doua"
   AND w=2;"a treia" AND w=3;"a patra" AND w=4;"a
   cincea" AND w=5;"a șasea" AND w=6;"." ' "FOARTE
   BINE !" AND w<4
```

care aduce o economie de *50* octeți .

b) *Înlocuirea grupului de instrucțiuni IF—THEN cu GO TO sau GOSUB*

Se bazează pe faptul că o condiție îndeplinită are valoare de adevăr. Astfel, în locul liniei

```
IF a=5 THEN GO TO 1000
```

se poate scrie

```
5 GO TO 1000*(a=5)
```

Combinată cu linii multiple de instrucțiuni, economia de memorie crește. De pildă, în locul programului

```
20 IF x=1 THEN GO TO 1000
30 IF x=2 THEN GO TO 2000
40 IF x=3 THEN GO TO 3000
```

se poate folosi forma mai avantajoasă

```
20 GO TO 1000*(x=1)+2000*(x=2)+3000*(x=3)
```

Această tehnică funcționează nu numai la instrucțiunile **GO TO** sau **GOSUB** ci și la instrucțiunea **LET**. De exemplu programul:

```
10 IF a=0 THEN LET x=25 * X+2 ↑ X
20 IF a>0 THEN LET x=X-X/2
30 IF a<0 THEN LET x=Y
```

se scrie

```
10 LET x=(25*X+2 ↑ X)*(a=0)+(X-X/2)*(a>0)+
   +Y*(a<0)
```

c) *Evitarea numerelor arabe*

Fiecare număr arab are nevoie, în afara cifrelor vizibile, de încă *6* octeți care nu se văd: unul arată calculatorului că urmează un număr iar ceilalți sînt cei *5* octeți ai numărului. O posibilitate de a evita acești octeți „invizibili" este de a înlocui cifrele arabe prin variabile (pentru numerele care apar des în program). Iată cîteva exemple:

1) `10 LET dd=2000 :GOSUB dd`

2) `10 LET z=NOT PI :LET u=NOT z :PRINT AT z, u; "Economisirea memoriei"`

3) `10 LET n0=NOT PI :LET n1=SGN PI :LET n2=n1+n1 :LET n3=n2+n1 :LET n4=n3+n1 :LET n5=n4+n1 :LET n6=n5+n1 :LET n7=n6+n1 :LET n8=n7+n1 : LET n9=n8+n1 :LET n10=VAL "10" : BORDER n4 :PAPER n4 :INK n7 :CLS`
`20 FOR a=-n2*n10 TO n2*n10 STEP n4 :BEEP .05, a :NEXT a`
`30 PRINT AT n0, n0; PAPER n6; INK n0; "32 blancuri"; AT n1, n0; "_ _ _ _ _ _ _ _ _ _ SOFTWARE 1992 _ _ _ _ _ _ _ _ _ _ _ _ _ _"; AT n2, n0; "32 blancuri"`

Presupunînd că variabila *q* este definită cu o valoare numerică în program, atunci *q = q* are valoarea *1* și *q > q* are valoarea *0*; prin urmare se pot scrie instrucțiunile:

```
20 LET p=q=q          (în loc de LET p=1)
30 LET x=q>q          (în loc de LET x=0)
40 PRINT TAB q>q      (în loc de PRINT TAB 0)
50 FOR j=q=q TO 10    (în loc de FOR j=1 TO 10)
```

O altă modalitate este utilizarea instrucțiunii **VAL**. Astfel se scrie:

```
10 LET z=VAL "200"        (în loc de LET z=200)
20 GO TO VAL "1000"       (în loc de GO TO 1000)
```

ceea ce aduce o economie de *3* octeți pentru fiecare număr (dispar cei *6* octeți invizibili dar apar *3* octeți pentru instrucțiunea **VAL**)

În mod analog:

```
10 IF x=20 OR y=10 THEN GO TO 300
10 IF VAL "x=20 OR y=10" THEN GO TO VAL "300"
```

12.2. COPIEREA PROGRAMELOR COMPLEXE

Programele complexe se copiază cu ajutorul programelor de copiere realizate de firme specializate. Cele mai folosite programe de copiere sînt

Zotyocopy+, *Monster Copy2*, *MACROCOPY* și *COPY/86 M*.

Modul de lucru :

a) Se încarcă programul de copiere cu comanda

LOAD "[nume]"

b) După încărcarea sa în memoria calculatorului, fiecare program de copiere afișează :

— capacitatea de memorare (încărcare) în octeți ;
— comenzile de lucru.

Capacitatea de memorare poate fi majorată la unele programe de copiere așa cum se indică în tabelul 12.1 :

TABELUL 12.1

Programul	*Zotyocopy +*	*MonsterCopy2*
Comanda de majorare	**CS** și **M**	**M**
Comanda de încărcare	**ENTER**	**L**
Comanda de salvare	**ENTER**	**S**

Programele la care s-a majorat capacitatea de încărcare nu mai pot fi utilizate după salvarea blocului cu o capacitate mai mare decît capacitatea inițială de memorare și trebuie reîncărcate. Din acest motiv este recomandabil ca să se copieze programele complexe prin divizarea lor în grupuri de cîte 2—3 blocuri de program.

Dacă vreunul dintre blocuri este greșit încărcat, programul de copiere semnalează eroarea (fie vizual ca la *Zotyocopy+* sau *COPY/86M*, fie sonor ca la *Monster Copy 2*). Programul *Monster Copy 2* prezintă particularitatea că blocurile fără *header* pot fi copiate tastînd tasta *D*.

Un program deosebit de eficient este *Copy/86M* care are avantajul de a comprima blocurile ce se copiază, reușind să salveze programe cu o lungime a tuturor blocurilor de circa *70 ko*, fără să fie necesară divizarea programului respectiv în loturi de blocuri componente (proprietatea de comprimare o deține și programul *TFCOPY*). De asemenea *COPY/86M* afișează conținutul programelor scrise în *BASIC*, ușurînd înțelegerea structurii programului care se copiază dacă se analizează loader-ul acestuia.

c) Se încarcă în memoria calculatorului programul care se copiază. Se recomandă să se scrie denumitile blocurilor componente, adresele de încărcare și lungimea în octeți a fiecărui bloc în parte.

d) Se salvează pe casetă programul utilizînd comenzile de salvare:

S, A și *ENTER* la **Zotyocopy+**

S la **MonsterCopy2**

A, C și ENTER la **COPY/86M.**

e) Se verifică dacă înregistrările pe bandă au fost corect efectuate, folosind comenzile de verificare, în prealabil banda fiind derulată la începutul înregistrării.

Observație : programul *COPY/86M* are în plus posibilitatea dezactivării autolansării programelor *BASIC*, apăsînd tasta *R* (după ce programul respectiv a fost încărcat de pe banda magnetică în memoria calculatorului sub *COPY/86M*).

BIBLIOGRAFIE SELECTIVĂ

1. DUMITRESCU LIVIU, *Microelectronica interactivă*, Editura tehnică, București, 1989
2. GEE S. M., *The Spectrum Programmer*, Granada, London, Toronto, Sidney, New York, 1982
3. NICULESCU STELIAN, DUMITRESCU M, *Algoritmi și metode de reprezentare*, E.D.P., București, 1978
4. NICULESCU STELIAN, *Algoritmi*, Colecția Știință și tehnică pentru toți, Editura tehnică, 1981
5. PETRESCU ADRIAN, FRANCISC IACOB, *Microcalculatorul personal* HC-80, AMC vol. 49, Editura tehnică 1985
6. PETRESCU ADRIAN și colectiv, *ABC de calculatoare personale și... nu doar atît...*, Editura tehnică, București 1990
7. VICKERS STEVEN, *ZX SPECTRUM — Sinclair, Basic Programming*, Sinclair Research Ldt, London, UK, 1983
8. *Colecția de reviste* SINCLAIR USER anul 1986
9. ZX SPECTRUM, manual de utilizare
10. HC-85, manual de utilizare

CUPRINS

Pag.

Capitolul 1. NOȚIUNI INTRODUCTIVE

1.1. Preliminarii 3
1.2. Configurația hard a calculatoarelor personale . . . 4
1.3. Punerea în funcțiune a calculatorului 4
1.4. Tastatura 6
1.4.1. Prezentarea tastaturii 6
1.4.2. Modurile de lucru cu tastatura 8
1.5. Probleme 13

Capitolul 2. LIMBAJUL BASIC

2.1. Noțiuni introductive 14
2.2. Caracterizarea limbajului BASIC 15
2.3. Structura limbajului BASIC 15
2.3.1. Alfabetul 16
2.3.2. Vocabularul 16
2.3-3- Gramatica 17
2.3.4. Semantica 17
2.4. Forma generală a unui program BASIC 17
2.5. Organizarea memoriei 18
2.6. Organizarea ecranului 22
2.7. Codurile caracterelor 23
2.8. Mesajele de eroare 25
2.9. Probleme 26

Capitolul 3. INSTRUCȚIUNI/COMENZI PENTRU EXECUTAREA, EDITAREA, ȘTERGEREA, LANSAREA, OPRIREA, CONTINUAREA, ÎNCĂRCAREA, SALVAREA PROGRAMELOR ȘI ȘTERGEREA ECRANULUI

3.1. Noțiunile „instrucțiune" și „comandă" 28
3.2. Instrucțiunea REM pentru comentarii 29
3.3. Comenzi pentru editarea și ștergerea unei linii de program 29
3.3.1. Cursorul de program (prompterul) 30
3.3.2. Comanda DELETE 30
3.3.3. Comanda EDIT 31
3.4. Instrucțiunea/comanda NEW pentru ștergerea programelor 31
3.5. Instrucțiunea comanda RUN pentru lansarea programelor 32
3.6. Instrucțiunile STOP, BREAK și PAUSE pentru oprirea programului 32
3.7. Comanda CONTINUE 33
3.8. Instrucțiunile/comenzile LOAD și MERGE pentru încărcarea programelor 34
3.9. Instrucțiunile/comenzile SAVE și VERIFY pentru salvarea programului și verificarea înregistrării sale . 36

3.10. Instrucțiunea comanda CLS pentru ștergerea ecranului 38
3.11. Probleme 38

Capitolul 4. INSTRUCȚIUNI PENTRU INTRODUCEREA DATELOR ȘI AFIȘAREA REZULTATELOR

4.1. Instrucțiuni pentru introducerea datelor 40
4.1.1. Instrucțiunile READ și DATA 40
4.1.2. Instrucțiunea RESTORE 41
4.1.3. Instrucțiunea INPUT 42
4.2. Instrucțiunea/comanda PRINT pentru afișarea rezultatelor 44
4.3. Probleme 48

Capitolul 5. INSTRUCȚIUNI DE ATRIBUIRE, SALT ȘI PENTRU REZERVAREA MEMORIEI. INSTRUCȚIUNI LOGICE ȘI PENTRU TESTAREA CLAVIATURII

5.1. Instrucțiunea de atribuire LET 50
5.2. Instrucțiuni comenzi de salt 53
5.2.1. Instrucțiunea/comanda GO TO 53
5.2.2. Instrucțiunea IF—THEN 54
5.2.3. Instrucțiuni de ciclare (FOR—TO—NEXT—STEP) . . . 55
5.3. Instrucțiunea DIM pentru rezervarea memoriei . . 60
5.4. Instrucțiuni logice și pentru testarea claviaturii (NOT, AND, OR, INKEY$) 66
5.5. Probleme 69

Capitolul 6. FUNCȚII STANDARD, FUNCȚII UTILIZATOR ȘI SUBRUTINE

6.1. Funcții standard 74
6.2. Funcții utilizator (DEF, FN, FN) 79
6.3. Subrutine (GOSUB-RETURN) 81
6.4. Probleme 83

Capitolul 7. INSTRUCȚIUNI PENTRU PRODUCEREA SUNETELOR ȘI FOLOSIREA CULORILOR

7.1. Producerea sunetelor (BEEP) 87
7.2. Instrucțiuni pentru culori (BORDER, PAPER, INK, FLASH, BRIGHT) 90
7.3. Probleme 94

Capitolul 8. INSTRUCȚIUNI GRAFICE

8.1. Instrucțiuni grafice propriu zise (PLOT, DRAW, CIRCLE) 98
8.2. Instrucțiuni ajutătoare (OVER, ATTR, POINT, INVERSE, SCREEN$, TRUE, VIDEO, INVERSE VIDEO) 100
8.3. Probleme 116

Capitolul 9. INSTRUCȚIUNILE DE LUCRU CU MEMORIA

9.1. Variabilele de sistem 120
9.2. Instrucțiunile USR 123
9.3. Instrucțiunile PEEK și POKE 123
9.4. Unele efecte realizate cu instrucțiunea POKE . . 124
9.5. Rutine în cod mașină apelabile cu instrucțiunea USR 125
9.6. Definirea de noi caractere grafice 130

9.6.1. Caractere semigrafice predefinite 130
9.6.2. Definirea de noi caractere grafice folosind o matrice de 8×8 pixeli 130
9.6.3. Definirea de noi caractere grafice folosind mai multe matrice 8×8 pixeli 133
9.6.4. Definirea de noi caractere grafice pe fiecare tastă cu un singur simbol. 136
9.6.5. Definirea de noi caractere grafice folosind variabila de sistem UDG 136

9.7. Folosirea unui set de caractere realizat de firme creatoare de soft 137
9.8. Folosirea mai multor seturi de caractere grafice definite pe literele din șirul a..u. 137
9.9. Probleme 140

Capitolul 10. INSTRUCȚIUNI PENTRU PORTURI, CĂI, CANALE, IMPRIMANTA, MICRODRIVE ȘI DISCHETE

10.1. Instrucțiuni pentru porturi (IN, OUT) 143
10.2. Instrucțiuni pentru căi și canale (OPEN#, CLOSE#) 145
10.3. Instrucțiuni comenzi pentru lucrul cu imprimanta (LLIST, LPRINT, COPY) 146
10.4. Comenzile pentru microdrive și dischete (FORMAT, CAT, ERASE, MOVE) 147

Capitolul 11. ARTIFICII PENTRU PERFECȚIONAREA ȘI PROTEJAREA PROGRAMELOR

11.1. Artificii pentru perfecționarea programelor . . . 148

11.1.1. Modalități de scriere 148
11.1.2. Cortine și modalități de ștergere a ecranelor 153
11.1.3. Modalități de operare cu ecranul. 154
11.2. Modalități de protecție a programelor 155
11.3. Alte modalități de perfecționare a programelor folosind rutine în cod mașină 158

Capitolul 12. ALCĂTUIREA ȘI COPIEREA PROGRAMELOR COMPLEXE

12.1. Structura unui program pe calculator 160

12.1.1. Programe loader. 160
12.1.2. Modalități de economisire a memoriei 162

12.2. Copierea programelor complexe 165

Bibliografie selectivă 167

Cuprins 169

Lei 396

În curs de apariție de același autor :

● **BASIC pentru calculatoarele ZX SPECTRUM, HC, TIM-S, COBRA, CIP, JET...**

(*COLECȚIE DE PROGRAME*)

Lucrarea conține programe tehnico-științifice, de matematică, de interes general, programe de divertisment (jocuri), precum și prezentarea programelor *BETA BASIC și HISOFT BASIC* însoțite de aplicații.

● **LIMBAJUL MAȘINĂ al calculatoarelor ZX SPECTRUM, HC, TIM-S, COBRA, CIP, JET...**

Este prima lucrare care tratează în mod unitar folosirea limbajului de asamblare Z80 și este ilustrată cu peste 150 rutine care realizează spectaculoase efecte vizuale sonore, de scriere, de animație, etc.

Au apărut :

R. M. Hristev — **Introducere în PROLOG**

Limbajul inteligenței artificiale pentru calculatoarele compatibile cu ZX Spectrum.

GHIDUL utilizatorului SPECTRUM

Scheme hard, harta memoriei ROM, modurile de utilizare ale tuturor compilatoarelor disponibile : BASIC, BETA-BASIC, FIFTH, Asamblor, Dezasamblor, PASCAL, C, FORTH, PROLOG.

ISBN 973 — 95175 — 4 — 4