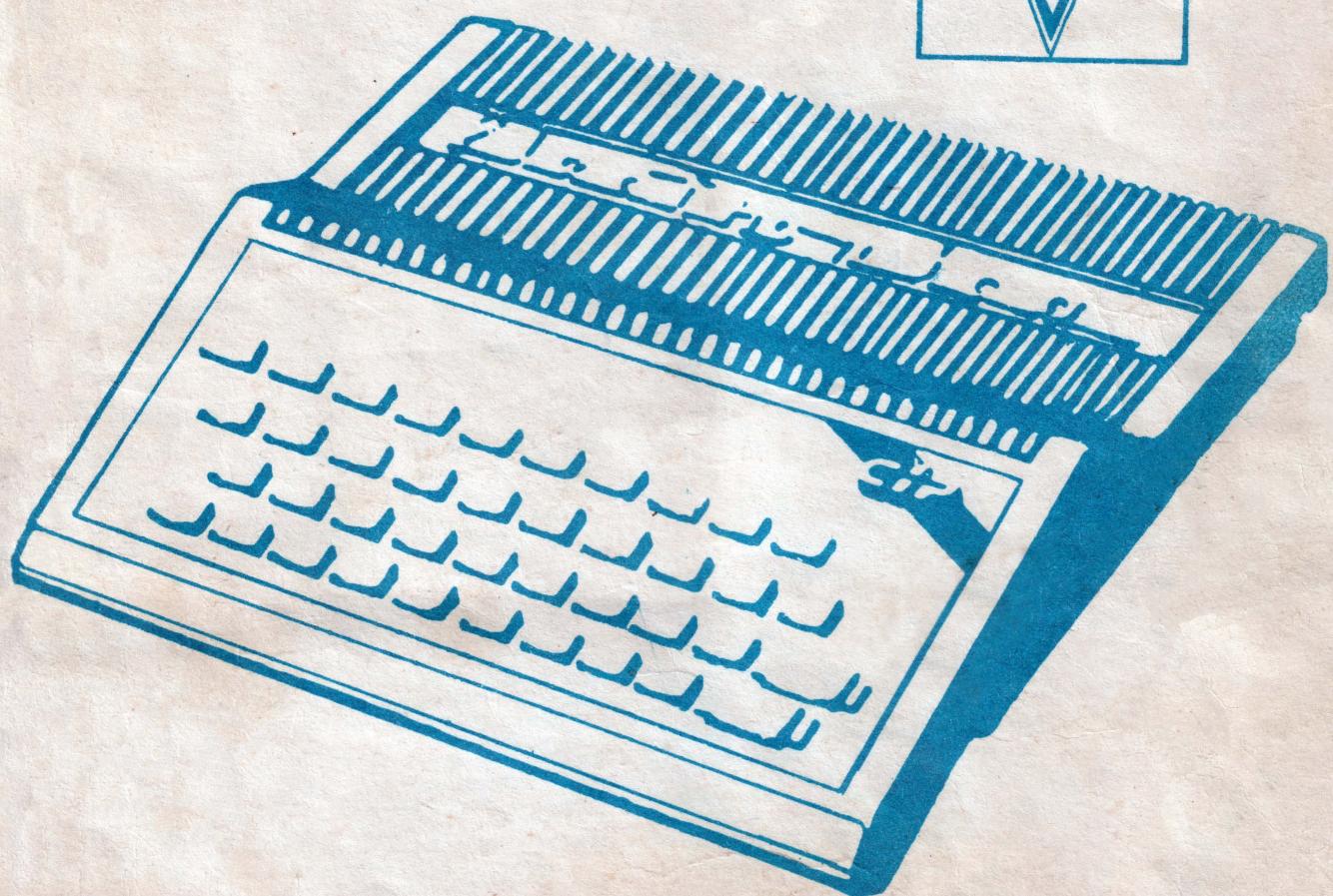
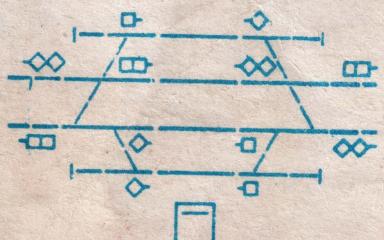
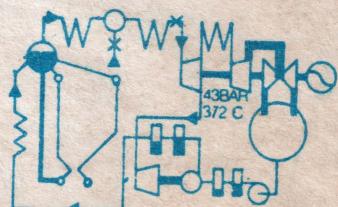


# CALCULATOR PENTRU INSTRUIRE 03



$$\begin{array}{r} \text{Pears} \\ \text{Pears} - \text{Pear} = ? \\ \text{Pears} + \text{Pears} = ? \end{array}$$



**manual de utilizare**



## INTRODUCERE

STIMATE CUMPĂRĂTOR,

Felicitori pentru că v-ați hotărât să folosiți această jucărie care, vă va oferi clipe de plăcere și satisfacție.

CIP-ul a fost conceput și vă este oferit ca un sprijin de nădejde în activitatea de instruire școlară, în aplicații practice din activitatea profesională, în industrie, în medicină, în cercetare, în calcule economice, etc..

### DATE TEHNICE

Microporcesor . . . . .	Z 80 A (MMN 80 CPU)
Memorie EPROM . . . . .	16 Ko
Memorie RAM . . . . .	64 Ko
Tastatură . . . . .	QWERTY cu 40 taste
Afișare . . . . .	TV. alb-negru/color PAL (canalul 8 OIRT) . . . . monitor alb-negru/color
Rezoluție . . . . .	8 culori cu două trepte de luminositate . . . . . 192 x 256 în regim grafic
Limbaj. . . . .	24 x 32 în regim alfanumeric . . . . . BASIC-S
Alimentarea . . . . .	220 V ca / 5 V cc
Puterea consumată . . . . .	max. 20 VA
Gabarit aprox. . . . .	330 x 280 x 60 mm
Greutatea aprox. . . . .	3,5 Kg. cu sursa de alimentare

### INVENTAR DE LIVRARE

- Unitatea de bază CIP
- Manual de utilizare
- Sursă de alimentare
- Rețea unităților service
- Cablu de conectare la TV
- Certificat de garanție

### CONDIȚII DE GARANȚIE

CIP-ul este garantat de ELECTRONICA S.A. 12 luni de la data cumpărării. Ies din garanție aparatelor la care a fost violat sigiliul. Înțeînerea aparatelor în perioada de garanție se asigură de către S.A. ELECTRONICA - SERVICE prin unitățile de lucru din întreaga țară nominalizate în "Rețea unităților service" - AX 1443.

### DE CE SE NUMEȘTE CIP ?

Aveți în față un Calculator pentru Instruire Personală programabil denumit mai pe scurt și mai nostrim "CIP", conceput pentru a fi folosit în procesul de învățămînt elementar, mediu și superior.

În învățămîntul preșcolar pe CIP se pot prezenta noțiuni elementare de grafică, muzică, desene animate etc. În învățămîntul liceal poate fi folosit eficient în instruire asistată, pentru însușirea materiilor.

În învățămîntul superior CIP-ul poate reduce timpul necesar muncii de rutină în realizarea proiectelor.

CIP-ul îl poate utiliza pentru însușirea unor limbiage evolute de programare a calculatoarelor.

CIP-ul este capabil de multe alte utilizări. Cu el puteți juca pe ecranul unui televizor: șah, GO, SCRABBLE, fotbal, baschet, puteți învăța să conduceți mașina, puteți desena în culori pe ecranul televizorului color, puteți compune muzică etc., CIP-ul dovedindu-se un partener intelligent, răbdător și perspicace.

Dar pentru unii dintre dumneavoastră, poate mai interesant decât folosirea programelor făcute de alții va fi realizarea unor aplicații proprii, în care imaginația și creativitatea de care veți da dovadă vă vor face să fiți foarte incintați, iar CIP-ul se va dovedi colaboratorul intelligent, bun la toate, cu condiția să fie învățat ce are de făcut. Deși poate uneori veți fi uimit de ce este capabil, rețineți că tot ce știe el să între în creații umane, CIP-ul fiind numai un calculator personal familial.

### CE FEL DE CALCULATOR PERSONALE EXISTĂ ?

Calculatoroarele personale sunt considerate ca fiind de două mari categorii:

- calculatoare personale familiale (HOME COMPUTER), cu un preț accesibil pentru a putea fi cumpărat "pentru acasă" și având destule posibilități de utilizare pentru calcule, desene, instruire, jocuri, etc.;
- calculatoare personale profesionale (PERSONAL COMPUTER) cu posibilități foarte mari de utilizare în instituții sau întreprinderi pentru aplicații de cercetare-proiectare, activități de birou, instruire asistată de calculator etc.

### CE AVEȚI ÎN FAȚĂ ?

Obiectul minunat cu care doriți să lucrați, face parte din categoria calculatoarelor familiale și veți constata cu plăcere că de mari fi sunt posibilitățile de utilizare.

### CE PROGRAME POT FI FOLOSITE ?

CIP-ul a fost conceput să înțeleagă și să execute programe făcute de firme renomate pentru calculatoarele de largă răspândire SINCLAIR - SPECTRUM. De asemenea, CIP-ul acceptă programe elaborate pe calculatoare românești HC-85 și TIM-S. Reciproc, programele realizate pe CIP vor putea fi executate pe calculatoare SINCLAIR ZX SPECTRUM sau pe calculatoarele românești HC-85 și TIM-S.

### CUM SE FOLOSEȘTE ACEST MANUAL ?

Acest manual vă prezintă, simplu și practic, cunoștințele necesare pentru a înțelege ce cuprinde un calculator, cum funcționează și cum puteți lucra curent cu el. Pentru a înțelege fără greutăți deosebite ce vi se prezintă în continuare, este necesar ca parcurgerea materialului să o faceți simultan cu folosirea CIP-ului. Veți avea foarte multe exemple, nu ezitați să le verificați pe calculator. Veți constata că limbajul utilizat este foarte exigent și prietenos în același timp, CIP-ul semnalându-vă orice eroare, neexecutind decit comenziile corecte.

### CIP-UL VĂ AJUTĂ SĂ VĂ CUNOAȘTEȚI !

De fapt calculatorul este o oglindă a intelectului dumneavoastră, executind cele ce-i veți da corect și clar, dar sănctorinând și orice lipsă de atenție, de logică sau de cunoștere.

Manualul se adresează tuturor, fără a cere o pregătire prealabilă în electronică sau informatică. El nu vă arată cum se construiește un calculator ci din ce este format, cum lucrează și cum îl puteți utiliza pentru diverse aplicații.

Fiecare capitol este structurat astfel:

- aplicație practică;
- noțiuni de bază;
- exemple de programe;
- întrebări recapitulative și exerciții;
- probleme propuse pentru rezolvare;
- răspunsuri.

Acolo unde în text aveți lăsat spațiu liber subliniat, vă solicităm să introduceți un răspuns al dumneavoastră bazat pe noțiuni explicate anterior.

Fiind un manual de autoinstruire, nimic nu vă forțează să-l studiați contra cronometru. Apreciați posibilitățile dumneavoastră de asimilare, care pot varia de la o zi la alta. Vă indicăm să nu studiați mai puțin de o oră, dar nici mai mult de trei-patru ore pe zi.

Materialul este astfel redactat încit să asimilați cunoștințele noi pas cu pas, cu minimum de efort, încit veți fi plăcut impresionat de posibilitățile dumneavoastră.

<<< S U C C E S >>>

## INSTALARE ȘI PUNERE ÎN FUNCȚIUNE

- În cazul în care aparatul a fost depozitat în condiții de umiditate excesivă, sau diferența dintre temperatura aerului în care a fost depozitat și cea a camerei în care este instalat este mare, trebuie să treacă minimum 3 ore pînă la prima punere în funcțiune;
- Nu introduceți sursa de alimentare în spații închise în timpul funcționării;
- Nu astupăți orificiile de aerisire ale sursei de alimentare și ale CIP-ului în timpul funcționării;
- Nu instalați CIP-ul și sursa de alimentare lîngă vase cu lichid încrucișat prin răsturnarea lor accidentală, lichidul poate pătrunde în interior, provocînd avarii sau incendii;
- În cazul în care a pătruns accidental lichid în interiorul aparatului, este recomandabil să nu-l porniți decît după cel puțin 12 ore, eventual chemăți service-ul;
- Înainte de curățarea exterioară, aparatul va fi deconectat de la rețea. Aparatul se va curăța cu o țesătură moale ușor umedă într-o soluție de săpun. Nu se vor utiliza diluații puternici sau benzina, deoarece atacă finisajul.

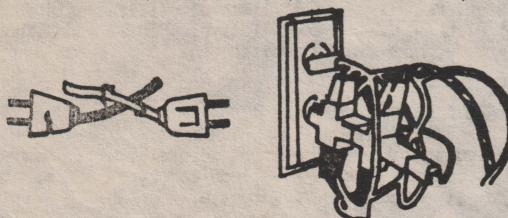
**ATENȚIE!** ► NU LĂSAJ CIP-UL SĂ FUNCȚIONEZE ÎN UMIIDITATE EXCESIVĂ ► NU FOLOȘI CIP-UL ÎN ÎNCĂPERI CU UMIDITATE EXCESIVĂ ► NU PERMITEȚI COPIILOR SĂ INTRODUCĂ MONEZI SAU ALTE OBIECTE METALICE ÎN APARAT, PRIN FANTELE DE VENTILAȚIE ► NEGLIGAREA ACESTOR REGULI POATE DUCE LA SUPRAINCĂLZIREA APARATULUI SAU CHIAR LA INCENDIEREA LUI.

### SURSA DE ALIMENTARE

CIP-ul se alimentează prin sursa de alimentare la tensiunea de 5 V cc. Sursa de alimentare a CIP-ului se alimentează de la rețea de curent alternativ de 220 Vef. În momentul conectării la rețea, asigurați-vă că priza este bine fixată în doză, iar ștecherul intră fix în priză.

### STRICIT INTERZIS !

- Să folosiți cordoane de rețea cu izolația compromisă.
- Să introduceți două tripluștecherie în aceeași priză.



- Să înlocuiți siguranțele cu alte valori decît cele pre-văzute.
- Să folosiți sursa de alimentare cu capacul desfăcut, intrucît riscăți să vă electrocutați.

Alegeți un loc de lucru care să vă permită un spațiu confortabil de așezare pentru CIP, televizor (la o distanță corespunzătoare) și casetofon.

Instalarea este extrem de simplă. Vă trebuie o priză triplă și ... puțină atenție. Prinț din spate CIP-ul și veți observa următoarele mufe/conectori:

conector pentru extensie (impri-mantă etc.)

mufă pentru videomonitor

buton pentru inițializare



Mufă CAS  
1,4 - ieșire  
3,5 - intrare  
2 - masă

Mufă AV  
5 - ieșire video  
4 - ieșire audio  
2 - masă  
3 - ieșire +5 Vcc.  
1 - neconectat

### EFFECTUAȚI URMĂTOARELE OPERAȚII:

- introduceți fișa alimentatorului în mufa DC-IN a CIP-ului;

**ATENȚIE !** FACEȚI ÎNTOȚDEAUNA ACEASTĂ OPERAȚIE CU ȘTECHERUL ALIMENTATORULUI SCOS DIN PRIZĂ, ALTfel RISCĂȚI SĂ SE ARDĂ SIGURANȚA ALIMENTATORULUI.

- introduceți cablul de casetofon în mufa notată CAS a CIP-ului;
- introduceți cablul TV în bornă de antenă a televizorului și în mufa RF a CIP-ului;
- porniți televizorul;
- introduceți ștecherul alimentatorului în priză (220 V);
- acordați televizorul pe canalele 6-12 pînă obțineți o imagine stabilă cu următorul text: BASIC-S.

BASIC-S

### DACĂ NU APARE IMAGINEA DESCRISSĂ MAI SUS VERIFICAȚI:

- dacă ați montat corect cablurile și dacă mufele sunt bine impinsene în contacte;
- încercați acordarea televizorului pe canalul necesar;
- acionați butonul RESET din spate-stînga. Acest buton provoacă inițializarea tuturor circuitelor fără a fi opri-tă alimentarea electrică.
- scoateți din priză și reintroduceți ștecherul alimentatorului.

Dacă nici una din acțiunile de mai sus nu au avut ca rezultat obținerea mesajului corect, există ceva defect în configurația CIP-ului și veți proceda conform condițiilor de garanție ale produsului.

### LISTA SEMNALELOR PREZENTE LA CONECTORUL DE EXTENSIE

Față A	A15	A13	D7	NC	SLOT	DO	D1	D2	D6	D5
Nr.	1	2	3	4	5	6	7	8	9	10

Față B A14 A12 VCC(+5V) NC SLOT GND GND CLK(3,5MHz) AO A1

Față A	D3	D4	INT	NMI	HALT	MREQ	IORQ	RD	WR	NC	WAIT
Nr.	11	12	13	14	15	16	17	18	19	20	21

Față B A2 A3 IORGE GND VIDEO NC NC NC BUSRQ RESET A7

Față A	NC	NC	M1	RFSH	A8	A10	VINH
Nr.	22	23	24	25	26	27	28

Față B A6 A5 A4 ROMINH BUSACK A9 A11

### OBSERVAȚII:

1. Față A este față plană.
2. Semnalele sunt numerotate privind dinspre față A, de la stînga la dreapta, ținînd calculatorul în poziție normală de lucru.
3. Atribuirea semnalilor la conectorul de extensie este identică cu cea de la calculatorul SINCLAIR SPECTRUM, exceptînd tensiunile de -5 V, +12 V, -12 V și semnalele U,V,Y (semnale de culoare) care nu se regăsesc la calculatorul CIP.

## STRUCTURA CIP-ului

CUM REZOLVĂ CIP-ul O PROBLEMĂ ?

Dacă, de exemplu, trebuie făcut următorul calcul :

$$D = A - (B - C)$$

Prin operații de "intrare" se introduc:

- programul de prelucrare, prin care CIP-ul este instruit cum să rezolve problema;
- datele problemei, adică valori pentru A, B și C.

Ce este programul de prelucrare ?

Programul cuprinde o mulțime de instrucțiuni (comenzi) scrisă într-un limbaj pe care-l înțelege calculatorul și ordinea în care vor fi executate.

Instrucțiunile programului corespund unor pași elementari în care se rezolvă problema.

În exemplul dat, instrucțiunile pot fi comentate astfel:

- 1 - cere / citește valori pentru B și C
- 2 - calculează  $(B - C)$
- 3 - cere / citește valoarea lui A
- 4 - calculează  $D = A - (B - C)$
- 5 - afișează / scrie (écran / hârtie) valoarea lui D
- 6 - mergi la punctul 1 (pentru a repeta secvența cu alte valori pentru A, B și C)

CIP-ul rezolvă problema conform programului introdus și rezultatele prelucrării făcute în circuitele electronice, sunt comunicate prin operații de ieșire.

Cum se soluționează o problemă pe CIP ?

Pentru realizarea unei aplicații pe calculator, se parcurg următoarele etape:

- formularea problemei
- stabilirea intrărilor (date furnizate CIP-ului) și ieșirilor (ce trebuie să furnizeze calculatorul)
- stabilirea modului de rezolvare
- verificarea programului pe calculator
- executarea curentă a programului.

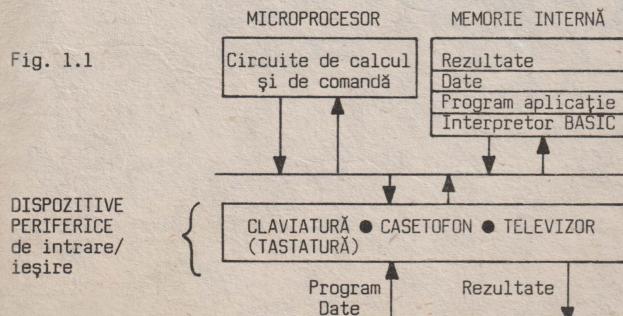
## CE SE ÎNTELEGE PRIN : HARDWARE și SOFTWARE

Specialiștii denumesc echipamentele ce alcătuiesc calculatorul printre-un singur cuvînt: HARDWARE, iar programele sint denumite: SOFTWARE.

## Configurația HARDWARE a CIP-ului

Cîteva noțiuni generale despre fiecare componentă și despre funcționarea lor în ansamblu sint prezentate în cele ce urmează.

Schema de funcționare a componentelor hardware ale unui calculator este următoarea (fig. 1.1):



Schema funcțională descrisă se regăsește concretizată în configurația hardware a calculatorului. Configurația minimă a CIP-ului cuprinde:

- microprocesor (tip Z80)

- memorie internă
- circuite de control a afișării
- circuite de interfață
- difuzor
- tastatură (claviatură)

Se adaugă următoarele "periferice" de intrare / ieșire :

- casetofon
- televizor sau monitor video.

Ce ar fi bine dacă ați avea ?

O configurație mai puternică poate include ca dispozitive periferice, imprimantă, joystick și unitate cu disc flexibil. Schema hardware de detaliu, incluzând și alte echipamente ce pot fi legate la CIP, este cea din fig. 1.2.

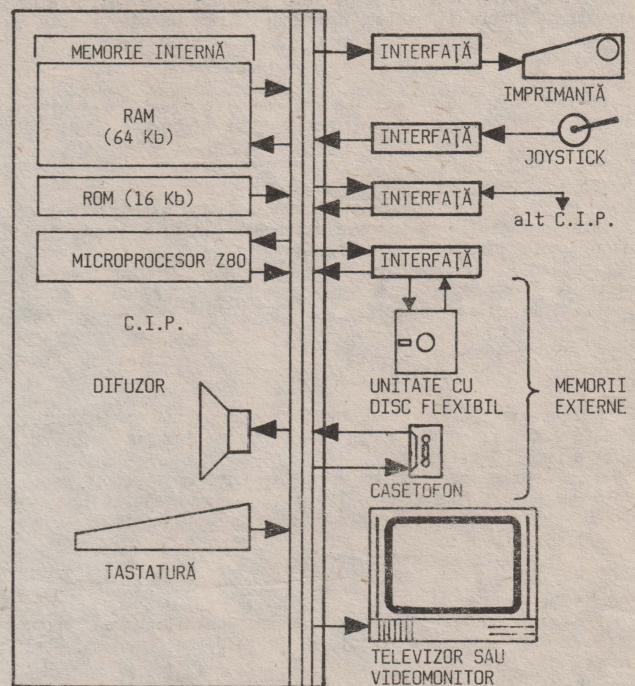


Fig. 1.2.

## Micropresorul

Micropresorul cuprinde pe o mică pastilă de siliciu (în engleză CHIP), piese electronice în miniatură care descrefăză, verifică și execută comenzi și calculele prevăzute în program.

## Memoria internă

În memoria internă se află un interpretor care transformă programul aplicației scris într-un limbaj ușor de învățat (limbajul BASIC). Într-un limbaj intern mult mai complicat, specific fiecărui tip de calculator (limbaj în "COD MAȘINĂ") care va putea fi executat de micropresor. Interpretorul permite deci introducerea în memorie (fig. 1.1) a diverselor programe de aplicații (SOFTWARE aplicativ) și executarea lor.

Programele solicită datele necesare și acestea sint memorate într-o altă zonă a memoriei interne. În acest fel, CIP-ul știe ce are de făcut și cu ce să lucreze. Micropresorul decodifică comenzi și instrucțiunile și le execută folosind datele disponibile.

Rezultatele prelucrării sint (fig. 1.1) expediate la dispozitivele periferice de ieșire.

Așa după cum probabil ați reținut, în memoria internă se introduc informații prin operații de "încărcare" sau de "scriere", iar din memorie se "citesc" sau se "extrag" informațiile introduse.

CIP-ul dvs. utilizează două tipuri de memorii:

## CAPITOLUL 1

### Memoria ROM

ROM (Read - Only - Memory) este o memorie al cărei conținut este stabilit din fabricație și care nu poate fi schimbat, ci poate fi numai "citit". Veți constata că, după întreruperea alimentării electrice a CIP-ului, la repunerea în funcție, programul inscris în ROM vă va sta la dispozitie fără a fi nevoie să-l introduceți dvs. O astfel de memorie se spune că este "nevolutilă" pentru că nu își pierde conținutul la întreruperea tensiunii de alimentare.

### Memoria RAM

RAM (Random - Acces - Memory) este memoria în care se păstrează programele și datele introduse de utilizator precum și informațiile interne necesare funcționării corecte a CIP-ului.

### Atenție !!!

Este o memorie "volatilă", la întreruperea alimentării electrice programele și datele introduse se pierd (se "volatilizează").

Apare o întrebare:

Cum facem să nu pierdem ce am muncit poate cîteva ore ? Există o soluție - memoria externă - caseta magnetică - despre care vom vorbi mai departe. Capacitatea de memorare este limitată de dimensiunea memoriei. Cum se exprimă ea și cum se lucrează cu memoria puteți afla în capitolul 12.

### Dispozitive periferice

Dispozitivele periferice sunt cele care permit introducerea programelor și datelor sau comunicarea informațiilor de ieșire pentru utilizator.

Sunt denumite "periferice" spre deosebire de microprocesor, care mai este denumit și "unitate centrală" a aparatului.

CIP-ul necesită ca periferice în configurația minimă:

- ▶ tastatură
- ▶ televizor sau monitor video
- ▶ difuzor
- ▶ casetofon

Într-o configurație extinsă, prin achiziționarea unor circuite de legătură specială (interfață), se mai pot atașa:

- ▶ imprimantă
- ▶ unitate cu disc flexibil
- ▶ joystick

### Tastatura

Claviatura CIP-ului (denumită TASTATURĂ) este asemănătoare cu a mașinii de scris (formată din TASTE), iar operația de utilizare se numește "TASTARE". Veți afla la capitolul 3 că ea are un mod deosebit de funcționare și de utilizare.

### Televizorul

Pentru vizualizarea dialogului cu calculatorul și a rezultatelor prelucrării puteți folosi televizorul dvs. alb-negru sau color.

### Difuzorul

CIP-ul are încorporat un difuzor care vă va semnaliza încărcarea programelor, sau prin care puteți obține diferite sunete (melodii).

### Casetofonul

Caseta magnetică este folosită ca suport de MEMORIE EXTERNA, deoarece pe ea se pot "salva" (înregistra) programele și datele din memoria internă, pentru a preveni "volatilizarea" lor la întreruperea tensiunii de alimentare.

Pentru a "încărca" programe de pe casetă și pentru a le "salva" pe casetă, folosiți un casetofon obișnuit, dar în bună stare de funcționare.

Un casetofon uzat, cu o mecanică defectă, vă va produce multe clipe neplăcute cind, având o casetă cu niște

jocuri/programe frumoase, nu veți reuși să le încărcați sau veți obține mesajul "Tape loading error" indicind o eroare de bandă.

O mică îndoitoră a benzii, insensibilă în cazul casetei cu muzică nu este acceptată de calculator.

În general, vă recomandăm să utilizați același casetofon la "salvarea" și la "încărcarea" programelor, pentru a evita incompatibilități datorită nealinierii capetelor la două apare deosebite.

Caseta magnetică prezintă avantajul unui preț redus, dar are și dezavantajul dificultății de acces la o anumită informație, deoarece permite numai o căutare secvențială (program după program) de la începutul benzii, fapt care cere timp și... răbdare.

### Discul flexibil

Discul flexibil este un suport de memorie externă care înlătură dezavantajul casetei, el permitând accesul direct și imediat la orice informație stocată.

Unitatea de disc flexibil este însă un dispozitiv mai scump, și deci mai dificil de procurat.

### Imprimanta

CIP-ul permite, prin intermediul unei interfețe specializate, cuplarea unei imprimante care să scrie sau să deseneze pe hârtie rezultatele prelucrării.

### Joystick-ul

Joystick-ul permite mai ușor deplasarea pe ecran în orice direcție, cu ajutorul unei manete, decit prin tastatură cu ajutorul căreia puteți obține numai "sus" / "jos" / "dreapta" / "stinga" / "foc".

### Întrebări recapitulative

I 1.1 Ce informații trebuie să introducem în calculator pentru a rezolva o problemă / a realiza un joc / a desena pe ecran ?

I 1.2 Ce echipamente hardware trebuie să aveți la dispozitie pentru a lucra cu CIP-ul ?

I 1.3 Casetofonul nu este permanent necesar. Cind aveți nevoie de el ?

### Răspunsuri

R 1.1 Programul aplicației și datele necesare.

R 1.2 CIP-ul plus televizor plus casetofon plus o priză multiplă plus cabluri de legătură.

R 1.3 a). Cind dorim să încărcăm programe "memorate" pe casetă.

b). Cind am introdus prin tastatură programe mai mari pe care este bine să le "salvăm" pe casetă înainte de oprirea alimentării CIP-ului.



## CAPITOLUL 2

Ați introdus o comandă de "ștergere" a ecranului pe care interpretorul BASIC a înțeles-o și a executat-o prompt (CLS - Clear Screen).

11. Numerotați liniile ecranului astfel:

```
apostrof
PRINT 1'2'3'4'5'6'7'8'9'10'11'12'13'14'15'16'17'18'19'20'21'
```

12. Priviți mai atent la o tastă, de exemplu la tasta "H". Deasupra ei sunt scrise patru caractere ce pot fi obținute cu acestă tastă:

```
GOSUB CIRCLE SQR ↑
```

Încercați obținerea lor apăsând tastele indicate mai jos, observând litera afișată în cursorul cliptor. Atenție! după afișarea fiecărui caracter, ștergeți cu CS + Ø pentru a obține din nou cursorul K.

caracter	tastări	cursor
GOSUB	H	K ▶ L
CIRCLE	1. CS + SS 2. SS + H	K ▶ E E ▶ L
SQR	1. CS + SS 2. H	K ▶ E E ▶ L
	SS + H	L

Pentru a obține "CIRCLE" este nevoie de două tastări succesive:

- 1 - CS + SS - CS și SS apăsați simultan
- 2 - SS + H - SS și H apăsați simultan

Tastatura se află în orice moment într-un anumit MOD CURENT de lucru, indicat printr-o literă în cursorul cliptor:

K (KEY WORD - cuvînt cheie) pentru comenzi/instrucțiuni  
L (LETTERS - litere) pentru litere mici și cifre  
C (CAPITALS - majuscule) pentru litere mari  
E (EXTENDED - mod extins) pentru caractere/comenzi  
G (GRAPHICS - mod grafic) pentru simboluri grafice

### Cum obțineți litere mari/mici ?

13. Tastați următoarele caractere observînd pe ecran litera din cursor și caracterele afișate:

```
cursor K -----> L
character 1ØPRINT " "
↑ apăsați tasta P ↑ apăsați tasta S + P
```

Apăsați simultan (CS + 2) și cursorul devine C permitînd introducerea literelor mari:

```
C
PRINT "ABC"
↑ (CS + 2)
```

Reapăsați simultan (CS + 2) și cursorul redevine L, putînd introduce litere mici:

```
C L C
10 PRINT "ABC abc DEF" <ENT>
↑ (CS + 2) ↑ (CS + 2)
```

### Cum obțineți VIDEONORMAL/VIDEOINVERS ?

14. Atunci cînd doriți să apară pe ecran informații care să însă în evidență față de altele, puteți folosi modul de afișare videoinvers, adică în loc de caractere negre pe fondul alb al ecranului (videonormal), să apară caracterele albe pe fond negru (videoinvers). Tastați următoarele:

```
20 PRINT "VIDEONORMAL [VIDEOINVERS] VIDEONORMAL" <ENT>
```

```
↑ (CS + 4) ↑ (CS + 3)
(INV. VIDEO) (TR. VIDEO)
```

Ați remarcat că trecerea dintr-un mod de afișare în altul se face tastînd (CS + 4) sau (CS + 3).

Dacă la apăsarea tastei <ENT> după linia 10 sau 20, vă apare un semn de întrebare în linie, BASIC-S vă semnalizează că aveți o greșală. Greșala constă probabil în faptul că nu ati introdus " după PRINT sau la sfîrșitul liniei. Într-un asemenea caz, ștergeți linia cu DEL (CS + Ø) și retastați-o atent.

Oricum retineti că BASIC-S nu acceptă informații neconforme cu regulile lui și vă semnalizează prompt orice eroare prin "?".

Dacă ati introdus corect linia 10 și 20, pe ecran trebuie să aveți:

```
10 PRINT "ABC abc DEF"
20 PRINT "VIDEONORMAL [VIDEOINVERS] VIDEONORMAL"
```

Nu vă impacientați că linia 20 continuă dedesubt și că nu este frumos aliniată cu partea de sus. BASIC-S nu ține neapărat la frumusețe, ci la corectitudine și el se descurcă cu linii nealiniate dar corecte.

15. Introduceți:

```
BORDER 1
↑ apăsați tasta 1
↑ apăsați tasta B
<ENT>
```

Ați introdus o comandă BASIC-S care a fost executată imediat, avînd ca rezultat ștergerea celor două linii de pe ecran și apariția unui chenar (o bordură) de culoare închisă (albastră). Apăsați:

```
<ENT> - reapar liniile 10 și 20
RUN <ENT> - se execută instrucțiunile cuprinse în
↑ liniile 10 și 20
apăsați R
```

Pe ecran apare rezultatul execuției:

```
ABC abc DEF
VIDEONORMAL [VIDEOINVERS] VIDEONORMAL
```

Caracterele sunt scrise în dreptul tastelor, prin trei culori corespunzătoare modului în care pot fi obținute. După o perioadă, lucrînd curent, vă va fi foarte ușor să introduceti orice caracter, numai prin simpla asociere vizuală între caracterul dorit, poziția tastei și culoare.

La început, de mare ajutor vă va fi Anexa B în care sunt indicate în ordine alfabetică, toate caracterele și tastăriile necesare pentru obținerea lor.

Pentru utilizarea tabelului din anexă vă fac cîteva precizări:

- cuvîntele cheie reprezentînd comenzi sau instrucțiuni, sunt precedate de ":" (priviți în Anexa B)

```
: BORDER
: GOSUB
: LOAD
: SAVE
```

## CAPITOLUL 2

Ele se obțin prin apăsarea tastei indicate, în modul K:

- la începutul liniei de program
- după numărul de linie
- după caracterul THEN
- după : (două puncte)

Alte caractere se obțin astfel:

BIN	- (CS+SS) B	- simultan (CS+SS) și apoi B
READ	- (CS+SS) A	
FLASH	- (CS+SS) SS+V	- simultan (CS+SS) și apoi SS+V
MOVE	- (CS+SS) SS+6	
*	- SS+B	- simultan tastele indicate
<=	- SS+Q	- idem
>=	- SS+E	- idem

16. Exersați utilizarea Anexei B completând răndurile următoare:

caracter	tastatură
---	----
ASN	----
BREAK	----
:COPY	--
:GRAPHICS	----
:NEXT	--
:RUN	--
:STOP	----
█	----
+	----
=	----
<>	----
cursor sus	----
cursor stînga	----
cursor dreapta	----

### Întrebări recapitulative

Răspundeți în scris pe o foaie separată și confruntați apoi răspunsurile dvs. cu cele corecte prezentate în continuare:

I.2.1. Ce taste veți folosi pentru:

- a introduce un spațiu
- b) a șterge înapoi
- c) a introduce semnele speciale (! ? + - ;)
- d) a șterge tot ecranul

I.2.2. Care este efectul acțiunării :

- a) butonului RESET
- b) tastei ENTER

I.2.3. Cum opriți CIP-ul?

### Răspunsuri

R.2.1. a) SPACE  
b) (CS+O)  
c) (SS+I) (SS+C) (SS+K) (SS+J) (SS+O)  
d) V <ENT>

R.2.2. a) Provoacă inițializarea tuturor circuitelor fără a fi opriță alimentarea electrică

- b) Expediază textul afișat către interpretorul BASIC și mută cursorul la începutul liniei următoare

R.2.3. Întrerupind alimentarea electrică.

## CAPITOLUL 3

### CUM COMUNICĂM CU CIP-UL?

Din capitolul 1 ati reținut că un calculator electronic știe să vă îndeplinească o dorință, în măsura în care l-ați învățat cum să o facă. Pentru a-l instrui, trebuie să-i introduciți în memorie o succesiune logică de INSTRUCȚIUNI / COMENZI formind un PROGRAM.

Pentru ca să înțeleagă programul și să execute instrucțiunile comenzi, acestea trebuie exprimate într-un limbaj pe care-l "știe" calculatorul.

CIP-ul este astfel construit încit să înțeleagă limbajul BASIC. BASIC-ul este un limbaj de programare cu o utilizare foarte largă datorită ușurinței cu care poate fi învățat și multiplelor lui posibilități de utilizare.

### De ce se cheamă BASIC

Numele limbajului BASIC provine din inițialele cuvintelor în limba engleză:

Beginner's All purpose Symbolic Instruction Code,

care în traducere liberă arată că este destinat tuturor aplicațiilor și este pentru începători.

Într-adevăr, acest limbaj este mai ușor de învățat decât limba engleză, el având un vocabular simplu, cu puține cuvinte, totuși cu reguli foarte stricte.

Chiar dacă nu cunoașteți limba engleză, nu va fi greu să învățați vocabularul BASIC-ului deoarece CIP-ul se va dovedi un "încăpăținat", refuzind să treacă peste greșelile dvs. și nu va executa decât ceea ce fi introduse corect.

Un program BASIC este constituit dintr-o mulțime de instrucțiuni scrise în acest limbaj și introduse în CIP pentru a fi executate.

### Cum execută CIP-ul un program?

Programul conceput și introdus de dvs. este de obicei numit PROGRAM SURSĂ. El este tradus de INTERPRETORUL BASIC și transformat într-un limbaj intern al calculatorului denumit limbaj COD MAȘINĂ.

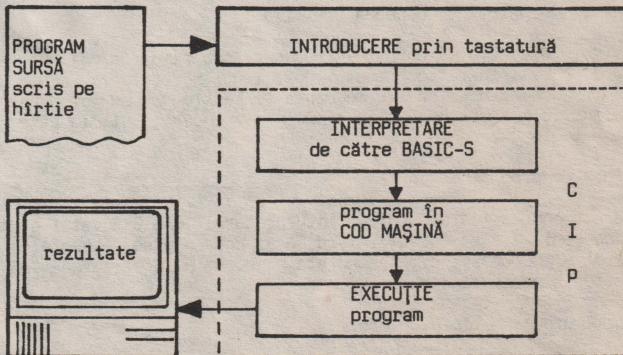


Fig. 3.1

### Cum codifică CIP-ul comenziile și instrucțiunile

Instrucțiunile / comenziile / datele, pentru a putea fi prelucrate în circuitele electronice ale calculatorului, sunt reprezentate sub formă de CARACTERE într-un cod în care sunt numerotate de la 0 la 255.

Prinții în anexa A a manualului și veți găsi în coloana din stînga un număr zecimal reprezentând codul și imediat în coloana din stînga un număr zecimal reprezentând codul și imediat în coloana din dreapta "caracterul" respectiv. De exemplu:



### CAPITOLUL 3

Cum ștergem o linie?

Stergerea unei liniilor complete se obține foarte ușor prin introducerea numărului ei urmat de <ENT>.

```
20 <ENT> - linia 20 dispăre
30 <ENT> - linia 30 dispăre
40 <ENT> - linia 40 dispăre
```

Datează din nou comanda de execuție a programului:

```
RUN <ENT>
a=?10 <ENT>
b=?20 <ENT>
```

Rezultat:

```
S=30
9 STOP statement, 10:5
```

Rezultatul calculului este același, dar diferă mesajul de oprire a programului. La prima execuție, oprirea s-a făcut prin instrucțiunea STOP, a doua instrucțiune din linia 40.

La a doua execuție oprirea s-a făcut prin instrucțiunea STOP, a cincea în linia 10:

Înainte de a trece la aplicația practică, să recapitulăm ce ați învățat din limbajul BASIC pînă acum:

- informațiile sunt considerate în calculator ca fiind formate din caractere;
- codul conform căruia este construit CIP-ul, cuprinde 255 de caractere (vezi anexa A);
- un program se introduce prin liniile numerotate (care sunt executate numai după o comandă specială) și numerotate (care sunt executate imediat);
- pe o linie multiplă se pot introduce mai multe instrucțiuni/comenzi, separate prin ":";
- execuția unui program se obține prin comanda RUN;
- modificarea unei liniilor se obține prin introducerea noului conținut cu același număr de linie;
- ștergerea unei liniilor se obține prin introducerea numai a numărului ei (linie vidă).

#### APLICAȚIE PRACTICĂ AP 3

1. Anulați programul anterior, introducind comanda imediată:

```
NEW <ENT>
tasta "A"
```

2. Tastați:

```
10 LET a=7 <ENT>
20 LET b=13 <ENT>
30 LET c=a+b <ENT>
40 PRINT c <ENT>
50 STOP <ENT>
```

Dacă ați greșit la introducere, linia nu este admisă cînd acționați <ENT> și eroarea vă este semnalată cu "?" clipitor.

Să simulăm acum o astfel de eroare pentru a vedea cum procedați. Tastați linia 30 astfel:

30 LET c ? +a+b L

Stergeți linia cu DEL (CS+Ø) și reintroduceți conținutul corect.

3. Introduceți comanda de ștergere a ecranului (vezi capitolul 2)

```
> ----- <ENT>
```

4. Ecranul fiind sters, să afișăm programul aflat în memorie. Introduceți comanda de listare a lui:

```
LIST <ENT>
```

Pe ecran reapare programul.

5. Instrucțiunile LET pot fi traduse astfel

```
LET a=7 (fie a=7)
LET c=a+b (fie c=a+b)
```

Instrucțiunea PRINT afișează valoarea lui c. Încercați să scrieți ce va fi afișat după execuția programului:

-----

9 STOP statement, -----:

6. Introduceți comanda de execuție a programului și verificați/corectați răspunsul dvs:

----- <ENT>

Rezultat:

-----

9 STOP statement, -----:

7. Tastați:

```
10 LET a=127 <ENT>
```

Observați că vechea linie 10 este înlocuită cu noul conținut introdus.

8. Ce rezultat veți obține după execuția programului?

9. Executați programul și comparați rezultatul obținut cu cel intuit de dvs.

10. Tastați:

```
50 <ENT>
```

11. Comparați programul afișat cu cel anterior. Ce observați?

12. Executați programul și analizați ce se afișează:

Observați că CIP-ul este un aparat "descurcăret". Chiar fără să-i spuneți unde să se opreasă (fără linia 50 - STOP), el a executat programul și v-a dat un mesaj că totul a decurs bine (OK) oprindu-se singur la prima instrucțiune a ultimei linii (40:1).

### CAPITOLUL 3

13. Introduceți comanda:

NEW <ENT>

Transcrieți mesajul obținut:

-----

14. Încercați să afișați programul:

LIST <ENT>

15. Încercați să executați programul:

RUN <ENT>

Nu mai puteți obține nimic pentru că programul a fost șters din memorie ca urmare a comenzi NEW, în vederea introducerii unui nou program.

16. Reintroduceți programul:

```
10 LET a=7 <ENT>
20 LET b=13 <ENT>
30 LET c=a+b <ENT>
40 PRINT c <ENT>
50 STOP <ENT>
```

17. Tastați:

```
35 LET d=b-a <ENT>
45 PRINT d <ENT>
```

18. Priviți programul afișat și veți constata că "ascultătorul" CIP a memorat cuminte noile instrucțiuni la locul potrivit: linia 35 între 30 și 40 și linia 45 între 40 și 50. Vom spune că liniile 35 și 45 au fost "INSERATE" (intercalate) în program.

Mai puteți remarcă ceva în linia 45

45 > PRINT d  
↑  
prompter de linie curentă

Semnul ">" reprezintă un prompter (indicator) de LINIE CURENTĂ și el arată întotdeauna ultima linie introdusă sau corectată.

Linia curentă poate fi modificată prin editarea (afișarea) ei la baza ecranului, printr-o comandă de editare tastind CS+1.

Încercați chiar acum următoarea secvență de operații:

BORDER 5 <ENT>	- apare chenarul, dar dispără programul de pe ecran
(CS+1) <ENT>	- programul este reafisat
	- la bază este "editată" linia 45
<ENT>	- linia 45 este repusă la locul ei în program
(CS+7)	- prompter-ul de linie este deplasat în linia 40
(CS+7)	- prompter-ul de linie este deplasat în linia 35
(CS+1)	- este editată ca linie curentă linia 35
<ENT>	- linia 35 (care poate fi modificată) este repusă la locul ei în program

Prompterul de linie poate fi deplasat în sus (CS+7), sau în jos (CS+6) având astfel posibilitatea să stabiliți ca linie curentă, orice linie a programului.

Verificați deplasarea prompter-ului de linie tăstind (CS+7) sau (CS+6).

19. Analizați programul afișat și încercați să intuiți care vor fi rezultatele execuției lui:

----- valoarea lui c  
----- valoarea lui d

20. Introduceți comanda de execuție și comparați rezultatele.

21. Executați următoarea secvență:

```
<ENT> - programul este reafisat
RUN - programul este șters de pe ecran și apar rezultatele
```

22. Reintroduceți linia 30 cu cîte un spațiu / blank între caractere astfel:

30 LET c = a + b <ENT>  
↑ ↑ ↑  
spații

Executați programul! Observați că totul decurge normal. BASIC-ul ignoră deci spațiile.

23. Stergeți din program liniile 45 și 35 introducind două liniile vide.

24. Modificați liniile 10 și 20 restabilindu-le astfel:

```
10 INPUT "a=?";a <ENT>
20 INPUT "b=?";b <ENT>
```

25. Modificați linia 40 prin editare astfel:

- stabiliți linia curentă dind comanda

LIST 40 <ENT>

Apăsind a doua oară pe <ENT> reapare programul cu prompterul în linia 40  
- coborîți linia curentă în poziția de editare cu (CS+1);  
- deplasați cursorul spre dreapta cu (CS+8), sau spre stînga cu (CS+5) și introduceți ghilimelele astfel încît linia să arate astfel:

40 PRINT "c=a+b=";c <ENT>

26. Noul program instruiește CIP-ul astfel:

10 INPUT "a=?";a	- cere valorile lui a și b și le
20 INPUT "b=?";b	memorează
30 LET c=a+b	- calculează c=a+b
40 PRINT "c=a+b=";c	- afișează c=a+b
50 STOP	- oprește execuția programului

27. Executați programul:

```
RUN <ENT>
a=?10 <ENT>
b=?20 <ENT>
c=a+b=30
9 STOP statement, 50:1
```

28. Repetați:

```
RUN <ENT> - rezultatul anterior dispără
a=?20 <ENT>
b=?10 <ENT>
c=a+b=100
STOP
```

29. Încercați altă formă de execuție a programului:

GOTO 10 <ENT> - rezultatul anterior nu mai dispără de pe ecran

tasta "G"

```
a=?23 <ENT>
b=?16 <ENT>
c=a+b=39 <ENT>
STOP
```

Iar:

```
GOTO 10      <ENT>
a=?2530      <ENT>
b=?39870      <ENT>
c=a+b=42400
STOP
```

Observați că execuția programului se poate face cu RUN și în acest caz valorile unei execuții anterioare sunt anulate, sau cu o instrucție

GOTO n

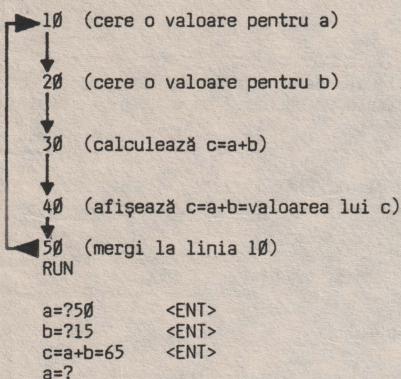
de trimis la o linie "n" a programului de unde doriti să inceapă execuția.

În acest ultim caz, valorile anterioare nu sunt anulate.

30. Înlocuiți conținutul liniei 50 cu următorul:

```
50 GOTO 10 <ENT>
```

Acum programul se va executa astfel:



deoarece ați înlocuit instrucția de oprire STOP cu instrucția GOTO ("mergi la") programul nu s-a mai oprit din execuție și va cere în continuare alte și alte valori pentru a și b. Încercați!

31. În momentul în care doriti să întrerupeți această execuție fără sfîrșit a programului, la solicitarea unei valori pentru a sau b, introduceți:

```
STOP <ENT> (STOP obțineți cu SS+A)
```

Programul va fi oprit cu mesajul:

H STOP in INPUT

32. Anulați programul existent în memorie și introduceți:

```
NEW
10 PRINT "multe salutări, prietene!"
20 PAUSE 0
30 GOTO 10
```

Execuția programul:

```
RUN <ENT>
```

Veți primi mesajul:

multe salutări, prietene!

Apăsați pe orice tastă și veți fi din nou salutat:

multe salutări, prietene!

În continuare, la apăsarea oricărei taste, veți primi la nesfîrșit salutări. Dar pînă cînd? Pînă veți dori să întrerupeți primirea de mesaje, determinată de fapt, de execuția

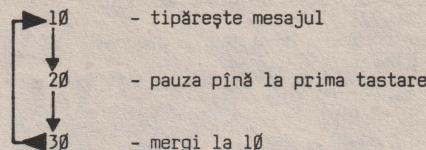
repetată a programului (se spune că programul "ciclează", adică a intrat într-un ciclu fără sfîrșit).

Introduceți comanda de întrerupere BREAK (CS+SPACE) și execuția programului va fi întreruptă, afișindu-se mesajul:

L. BREAK into program, 20:1

Apăsați <ENT> pentru a fi reaflat programul și priviți-l cu atenție. Ce s-a întîmplat?

Ultima instrucție a programului GOTO 10 a determinat de fiecare dată următorul ciclu:



Comanda BREAK va permite întreruperea execuției unui program în orice moment, cu excepția cazului cînd CIP-ul aşteaptă un răspuns la o instrucție INPUT, cînd puteți întrerupe execuția numai prin STOP, așa cum ați văzut în exemplul precedent.

Execuția unui program întreruptă prin BREAK/STOP, poate fi reluată cu instrucția CONTINUE (prescurtat CONT). Încercați!

O ultimă noutate în acest capitol:

Instrucția PAUSE permite oprirea execuției programului, la dorința programatorului, astfel:

PAUSE n	- oprește execuția pe o perioadă de (n/50) secunde
PAUSE 0	- oprește execuția pînă la apăsarea oricărei BREAK <ENT>
-----	- ștergeți ecranul
-----	- reluați programul întrerupt
multe salutări, prietene!	
-----	- întrerupeți programul
-----	- ștergeți programul din memorie

Recapitulare: operații și comenzi pentru elaborarea și execuție program

- Linia de program curentă (în care se pot face modificări) este indicată de prompter-ul ">". Ea poate fi stabilită prin deplasarea prompter-ului cu ↑ sau ↓.
- Modificări într-o linie de program se fac astfel:
  - .se poziționează prompter-ul de linie curentă cu ↓ (CS+6) sau cu ↑ (CS+7).
  - .se deplasează cursorul pe linie cu → (CS+8) sau ← (CS+5), se șterg (CS+0) caracterele nedorite și se introduc cele noi.
  - .se apăsă <ENT>.
- Execuția unui program:
  - RUN sau RUN nr. linie
  - GOTO nr. linie
- Afisarea unui program
  - LIST sau LIST nr. linie
- Se afișează un ecran încheiat cu scroll? (alt ecran?) puteți răspunde:
  - .apăsind orice tastă, pentru continuarea afișării;
  - .tastind "N" pentru a renunța la afișarea programului
- Ștergerea unui program din memorie:
  - NEW
- Întreruperea execuției unui program:
  - BREAK (CS+SPACE)
  - STOP (CS+A)
 execuția se reia cu:
  - CONT
  - PAUSE n -oprește execuția pe o perioadă de n/50 sec.
  - PAUSE 0 -oprește execuția programului pînă se apasă orice tastă
- Ștergerea ecranului
  - CLS

ÎNTREBĂRI RECAPITULATIVE

- Î 3.1 Cum ștergeți o linie dintr-un program BASIC?  
 Î 3.2 Cum inserăti o linie nouă?  
 Î 3.3 Cum schimbați conținutul unei linii deja introdusă?  
 Î 3.4 Care este comanda de afișare a unui program memorat?  
 Î 3.5 Cum ștergeți din memorie un program memorat?  
 Î 3.6 Instrucțiunile fără număr de linie au nevoie de comanda RUN pentru a fi executate?  
 Î 3.7 Cum este indicată linia curentă și cum puteți stabili o linie curentă?  
 Î 3.8 Cum deplasați la stînga și la dreapta cursorul într-o linie de editare?  
 Î 3.9 Cum puteți întrerupe execuția unui program și cum reluați execuția?  
 Î 3.10 Cum opriți execuția unui program pînă la prima tastare?

RĂSPUNSURI

- R 3.1 a) Introducind o linie vidă cu același număr de linie.  
 b) Ștergind înapoi cu DEL (CS+Ø) caracter după caracter.  
 R 3.2 Cu un număr de linie intermediar.  
 R 3.3 Introducind o linie cu același număr, dar cu un nou conținut.  
 R 3.4 LIST.  
 R 3.5 Cu ajutorul comenzi NEW.  
 R 3.6 Nu, ele sunt imediat executate.  
 R 3.7 Linia curentă este indicată de prompter-ul de linie și poate fi stabilită prin deplasarea prompter-ului cu ↓ sau ↑.  
 R 3.8 Cu ← sau →.  
 R 3.9 Cu BREAK/STOP; CONT.  
 R 3.10 Cu PAUSE Ø.

CUM CALCULEAZĂ CIP-UL?Operații; Operatori; Constante; Variabile; Expresii

Deoarece acum ati făcut "primii pași" în BASIC, puteți trece la noțiuni de detaliu ale limbajului.  
 O precizare importantă!

În continuare nu veți mai avea specificată acționarea tastei ENTER. Nu uitați însă ca de fiecare dată să o tastați, pentru că altfel cursorul va clipe uiuit la sfîrșitul liniei de program, și informațiile tastate de dvs. vor rămîne numai pe ecran, fără a fi cunoscute de calculator.

APLICAȚIE PRACTICĂ AP4

1. Introduceți următorul program:

```
10 INPUT "x=";x:PRINT"x=";x
20 INPUT "y=";y:PRINT"y=";y
30 LET z=x+y
40 PRINT "====="
50 PRINT "z";z
60 STOP
```

Așa cum probabil v-ați dat seama, în linia 30 se face o operație de adunare între două numere, care vă sănt cerute în linile 10 și 20.  
 Execuția programul:

(nu uitați ENTER!)  
 $x=14.7$  (în BASIC numerele zecimale se scriu cu punct, nu cu virgulă)  
 $y=15.3$   
 $=====$   
 $z=30$

2. Schimbați linia 30 astfel:

30 LET z=x-y

Înainte de a executa programul, scrieți mai jos care va fi rezultatul dacă veți introduce  $x=20$  și  $y=6$

$z=$  \_\_\_\_\_

Acum execuția programul și comparați rezultatul cu cel presusul de dvs.

3. Introduceți:

30 LET z=x\*y

Afișați programul.  
 Execuția programul cu  $x=8$  și  $y=3$

$z=$  \_\_\_\_\_

Ce operație aritmetică se notează cu "\*"? \_\_\_\_\_

4. Introduceți:

30 LET z = x / y

Afișați programul și execuția-l atribuind variabilelor valorile:  $x=36$  și  $y=4$

$z=$  \_\_\_\_\_

Ce operație aritmetică se notează cu "/"? \_\_\_\_\_

5. Încercați:

30 LET z=x+y-y/4

Afișați programul și studiați-l puțin. Ce rezultat credeți că se va obține dacă introduceți  $x=6$  și  $y=12$

$z=$  \_\_\_\_\_

Execuția programul și comparați rezultatul cu cel anticipat

## 6. Anulați programul și introduceți:

```

10 LET a=5*5
20 LET b=5  $\Delta$  2      (" $\Delta$ " se obține cu SS+H)
30 PRINT "a=";a
40 PRINT "b=";b
50 STOP

```

Execuția programului și veți remarca obținerea aceluiși rezultat. Prin " $\Delta$ " în BASIC se notează ridicarea la putere. sau exponentierea, așa încât în linia 20 se calculează "cinci la puterea a două" ceea ce este echivalent cu linia 10.

## 7. Schimbăți:

```

10 LET a=5*5*5
20 LET b=5  $\Delta$  3

```

Cit credeți că se va obține?

a= \_\_\_\_\_  
b= \_\_\_\_\_

Execuția programului și confruntați rezultatele.

## 8. Completăți operatorii aritmétici folosiți pentru următoarele operații:

scădere \_\_\_\_\_  
exponentiere \_\_\_\_\_  
înmulțire \_\_\_\_\_  
împărțire \_\_\_\_\_  
adunare \_\_\_\_\_

## 9. Anulați vechiul program și introduceți o instrucțiune imediată (fără număr de linie):

```
PRINT "a=3+4*8/2=";3+4*8/2
```

notați rezultatul a= \_\_\_\_\_ = \_\_\_\_\_

Șă analizăm puțin ce s-a întâmplat:

- instrucțiunea a fost executată imediat (fără RUN);
- instrucțiunea PRINT a executat calculul expresiei și apoi a afișat rezultatul;
- operațiile s-au făcut într-o anumită ordine.

Instrucțiunea respectivă este transcrierea în limbajul BASIC a expresiei aritmétice:

$$a = 3 + 4 \times \frac{8}{2}$$

CIP-ul a efectuat mai întii înmulțirea și împărțirea și apoi a făcut adunarea.

## 10. Introduceți, notați, analizați și comparați rezultatele:

```

PRINT "a=(3+4)*8/2=";(3+4)*8/2 rezultat: _____
PRINT "a=3+(4*8)/2=";3+(4*8)/2 _____
PRINT "a=3+4*(8/2)=";3+4*(8/2) _____

```

Calculatorul a efectuat calculele în următoarea ordine:

1. paranteza
2. înmulțirea/împărțirea
3. adunarea

Dacă înțilnește paranteze evaluează prioritățile lor

## 11. Introduceți și notați rezultatele:

```

PRINT 2x100
PRINT 2x100x100
PRINT 2x100x100x100
PRINT 2x100x100x100x100

```

Remarcați că rezultatul ultimei operații a fost afișat sub forma  $2E+8$  în loc de: 200000000.

## 12. Ștergeți ecranul (CLS) și introduceți:

rezultat	-----
PRINT 2/100	.02
PRINT 2/(100x100)	-----
PRINT 2/(100x100x100)	-----

Ultimul rezultat în loc de 0.000002 a fost afișat sub forma:  $2E-6$  ( $2E$  minus 6) unde "E-6" arată că este un număr zecimal în care cifra 2 se află după virgulă în a șasea poziție (cu zerouri în față).

## 13. Scrieți în forma acceptată de interpretorul BASIC următoarele expresii aritmétice. Cum credeți că vor fi afișate rezultatele?

7	-----
10x10	-----
6	-----
10:10	-----

Introduceți instrucțiunile corespunzătoare și verificați forma de afișare a rezultatelor.

## 14. Introduceți:

```

10 PRINT 17
20 LET a=17:PRINT a
30 LET a=39:PRINT a

```

Execuția programului!

În linia 10 cerem calculatorului să afișeze numărul 17. În linia 20 cerem același lucru, dar prin intermediul literei "a" căreia îi vom putea da și alte valori. Limbajul BASIC vă permite să lucrezi atât cu valori CONstanTE (neschimbate) cât și cu VARIABILE (nume care pot lua diferite valori). Constantele și variabilele sunt admise în anumite condiții despre care veți afla multe în continuarea acestui capitol. Dacă nu veți respecta aceste condiții (din necunoaștere sau neatenție) interpretorul BASIC va refuza să le ia în considerare semnând eroarea cu un semn de întrebare în linia de editare.

## 15. Încercați validitatea unor constante numerice introducând valorile date pentru x și notând ce se afișează:

```

NEW
10 INPUT "x=";x:PRINT x
20 GO TO 10
RUN

```

valori pentru x: \_\_\_\_\_ ce se afișează: \_\_\_\_\_

123	123
3.14	-----
12345678	-----
123456789	-----
.123	-----
.0123	-----
+210	-----
-210	-----
12abc	-----

I2 ? abc (eroare nu sunt admise litere)

Ștergeți literele și valoarea va fi admisă.

## 16. Schimbăți linia 10:

```

x=STOP      <ENT> (STOP obțineți cu SS+A)
10 INPUT    "x=";x:$:PRINT x$
RUN

```

valori pentru x: \_\_\_\_\_ ce se afișează: \_\_\_\_\_

abc	-----
12abc	-----
BaSiC	-----

aceasta e o VARIABILĂ SIR

Prin adăugarea caracterului "\$" variabila x este definită ca o VARIABILĂ SIR în care se pot introduce orice caractere alfanumerice.

## CAPITOLUL 4

17. Acționați butonul RESET și introduceți cu atenție:

```
10 INPUT "caracter 1: ";a$;"caracter 2: ";b$
20 IF a$ > b$ THEN PRINT a$;" > ";b$:GO TO 10
30 IF a$ < b$ THEN PRINT a$;" < ";b$:GO TO 10
40 PRINT a$;" = ";b$:GO TO 10
```

Programul compară două caractere introduse la cerere și afișează relația dintre ele, folosind operatorii BASIC

```
> (mai mare decit ...)
< (mai mic decit ...)
= (egal cu ...)
```

Compararea se face pe baza codurilor ASCII ale caracterelor, indicate în anexa A. Priviți în anexa A:

- litera "F" are codul 70, iar litera "E" are codul 69.
- Rezultă că: F > E (pentru că 70 > 69).
- cifra "7" are codul 55, iar litera "d" are codul 100.
- Rezultă că: 7 < d.
- semnul "%" are codul 37, iar semnul "?" are codul 63.
- Rezultă că: % < ?.

Verificați executind programul și confruntați rezultatele cu codurile din anexa A:

caracter 1: 4	caracter 2: 9
	4 < 9
caracter 1: k	caracter 2: A
	k > A
caracter 1: ♠	caracter 2: +
	♠ > +
caracter 1: p	caracter 2: B
	p > B

18. Introduceți:

```
RESET
10 CLS:INPUT "a=(5...10)= ";a:PRINT "a= ";a
20 IF a>5 AND a<=10 THEN CIRCLE 125, 85, 30
:PAUSE 200: GO TO 10
30 IF a<5 OR a>10 THEN PRINT AT 18,1;FLASH 1;
"număr în afara intervalului!":PAUSE 300:GO TO 10
```

Programul vă cere să-i introduceți un număr cuprins între 5 și 10 și în funcție de răspunsul dvs. reacționează astfel:  
1) dacă numărul introdus va fi mai mare decit sau egal cu 5 (a>=5) și (în engleză AND).mai mic sau egal cu 10 (a<=10), adică dacă va fi cuprins în intervalul (5,10], CIP-ul vă va oferi ca premiu un cerc în mijlocul ecranului (cu instrucția CIRCLE);

2) dacă numărul introdus va fi mai mic decit 5, SAU (în engleză OR) mai mare decit 10, CIP-ul vă va atrage atenția clipind că numărul nu se încadreză în valorile premiabile (cu instrucția FLASH 1).

Executați programul și verificați vigilența calculatorului (de ex. cu a=7, apoi a=3).

Pe scurt, programul poate fi comentat astfel:

- sterge ecranul
- linia 10
  - . cere un număr între 5 și 10
  - . afișează numărul introdus
- linia 20
  - . dacă numărul este între 5 și 10 desenează un cerc
  - . așteaptă 4 secunde și apoi întoarce-te la linia 10
- linia 30
  - . dacă numărul este mai mic decit 5 sau mai mare decit 10 afișează clipitor "număr în afara intervalului"
  - . așteaptă 6 secunde și apoi întoarce-te la linia 10.

19. Încercați acum dacă CIP-ul este suficient de "deștept" încât să înțeleagă și informații spuse invers. În programul următor să-i cereți să se descurce în următoarea situație:

- îi memorăți printre variabilelă și numele unei păsări cunoscute: "cuc";
- el să vă ceară un nume de pasare;
- dacă vreți să-i încercați răbdarea nu-i dați la ince-

put ce știe el, ci orice alte nume de păsări, caz în care să vă ceară cu insistență pasarea pe care o ține el minte;  
- cind îi veți satisface, dorința, să afișeze bucuros cîntecul cucului !

Introduceți:

```
RESET
10 CLS:LET p$="cuc"
20 INPUT "o pasare: ";x$
30 IF NOT p$=x$ THEN PRINT "alta pasare!":PAUSE 50:
GO TO 10
40 PRINT AT 10,12;FLASH 1;"cucu!!!":PAUSE 200:
GO TO 10
```

În program este utilizat operatorul logic NOT (în românește "nu") pentru a pune condiția: dacă p\$ NU este egal cu x\$ ATUNCI solicită din nou un sir pentru x\$.

Și acum după ce ați constatat în dialog direct cu CIP-ul cum calculează, dacă nu toate noțiunile vă sint foarte clare, ele vor fi reluate cu explicații suplimentare și în exerciții/intrebări la sfîrșitul fiecăruia capitol.

### NOȚIUNI DE BAZĂ. OPERATORI

Limbajul BASIC folosește următoarele tipuri de operatori:

a) operatori aritmétici:

- + adunare
- scădere
- \* înmulțire
- / împărțire
- $\sqrt[4]{}$  ridicare la putere (exponențiere)

b) operatori relationali

- = egalitate
- $\neq$  neegalitate
- > mai mare decit
- < mai mic decit
- $\geq$  mai mare decit sau egal cu
- $\leq$  mai mic decit sau egal cu

c) operatori logici

- AND "și"
- OR "sau"
- NOT negație

d) operator de alipire (concatenare) siruri

- + plus

### PRIORITĂȚI ÎN EXECUȚIE

Prioritatea (ordinea) în execuție a operațiilor este următoare:

1- expresiile din paranteze

2-  $\sqrt[4]{}$  (ridicarea la putere/exponențierea)

3- \*, / (înmulțirea, împărțirea)

4- +, - (adunarea, scăderea)

5- =,  $\neq$ ,  $>$ ,  $\geq$ ,  $<$ ,  $\leq$

6- NOT

7- AND

8- OR

O expresie cu operatori de aceeași prioritate se execută de la stînga la dreapta.

De exemplu expresia  $3 + \frac{1}{2} - 2$  se scrie în limbajul BASIC

$3+4 \sqrt[4]{2}/8-2$  și se execută astfel:

$$1. 3+16/8-2$$

$$2. 3+2-2$$

$$3. 5-2$$

rezultat:3

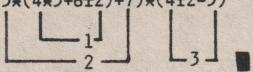
Încercați:

NEW

PRINT  $3+4 \sqrt[4]{2}/8-2$

Exemplu:  $[3(4 \times 5 + 6^2) + 7] (4^2 - 3)$

Expresia în limbajul BASIC se scrie astfel:  
 $(3 \times (4 \times 5 + 6 \times 2) + 7) \times (4 \times 2 - 3)$



Ordinea de execuție este:

- 1-  $6 \times 2$
- 2-  $4 \times 5 + 36$
- 3-  $3 \times 56 + 7$
- 4-  $4 \times 2$
- 5-  $16 - 3$
- 7-  $175 \times 13$

Calculați singuri și notați rezultatul:

Verificați cu o instrucțiune PRINT imediată.  
 Probabil v-ați întrebat de ce: dar radicalul sau extragerea de rădăcină cum se calculează?  
 Pentru exemplificare să scriem linia de program pentru calculul ipotenuzei unui triunghi dreptunghic. Fie variabilele:

$a, b$  - catetele triunghiului       $c$  - ipotenuza

Formula aritmetică de calcul este:

$$c = \sqrt{a^2 + b^2}$$

În BASIC se poate scrie în două feluri:

$(a \uparrow 2 + b \uparrow 2) \times 0.5$  sau  $\text{SQR}(a \uparrow 2 + b \uparrow 2)$

#### CONSTANTE NUMERICE

Constantele numerice pot avea:

- format întreg:  
16 sau +16  
-305
- format decimal (real):  
6.25 (echivalent cu 6,25)  
0.95 sau .95 (pentru 0,95)
- format exponential:  
 $E + n$  (echivalent cu  $10^n$ ) unde  $n$  este un număr întreg cuprins între -38 și +37

Exemple:

-1234=-1.234E+3=-.1234E4  
50000005E6=5E+6  
.0000123=-1.23E-5=-123E-7

Introduceți și verificați afișarea aceluiași rezultat:

```
PRINT -.0000123
PRINT -1.23E-5
PRINT -123E-7
```

#### CONSTANTE ȘIR DE CARACTERE

Oricine șir de caractere alfanumerice introduse între "" constituie o constantă șir.

Exemple:

"abc"  
"NR,\_CRT"  
"Nu opriți banda, se incarcă programul!"

Dacă aveți nevoie să folosiți în șirul de caractere chiar caracterul " (ghilimele), îl introduceți de două ori pentru fiecare apariție în șir.

```
10 PRINT "CIP"      -afișează: CIP
20 PRINT """CIP"""" -afișează: "CIP"
```

#### VARIABILE NUMERICE

Variabilele numerice pot fi formate din oricite caractere alfanumerice, cu condiția să înceapă cu o literă:

Exemple:      a
   
                  p237
   
                  material58

Introduceți:

```
10 LET lungimea=8:LET latimea=6
20 LET suprafata=lungimea*latimea
30 PRINT"suprafata=";suprafata
RUN
```

În acest program aveți:

- constante numerice: 6 și 8
- constante șir: "suprafata"
- variabile numerice: lungimea, latimea, suprafata

#### VARIABILE ȘIR DE CARACTERE

Puteți defini în program variabile cărora să le puteți atribui diferite șiruri de caractere alfanumerice și numerice. Numele unei variabile șir trebuie să conțină o singură literă urmată de semnul \$.

Exemple:

```
A$="DA sau NU"
x$="SALUT!"
```

Un șir de caractere poate fi utilizat și parțial sub formă de SUB\$IR. Manevrarea subșirurilor se face cu:

```
S=(n1 TO n2)
- S este o constantă șir sau o variabilă șir
- n1 primul caracter al subșirului
- n2 ultimul caracter al subșirului.
```

Fie șirul: a\$="abcd"

Verificați manevrarea lui introducind următoarele instrucțiuni cu execuție imediată:

Ce se afișează

CLS	
LET a\$="abcde"	
PRINT 12345	12345
PRINT a\$	abcde
PRINT a\$(3)	c
PRINT a\$(2 TO 4)	bcd
PRINT a\$(1 TO 3)	abc
PRINT a\$(2 TO 5)	bcde
PRINT a\$(2 TO 2)	b
PRINT a\$(4 TO 3)	linie vidă (blank)
PRINT a\$(1 TO 5)	abcde
PRINT a\$(1 TO 6)	mesaj de eroare deoarece șirul are numai 5 caractere

Iată un exemplu mai sugestiv.

Dacă introducem într-un șir lunile anului prescurtate în trei litere astfel: l\$="ian\_feb\_mar\_apr\_mai\_iun"  
 Putem extrage pentru un eventual calendar subșiruri. Introduceți și verificați!

```
NEW
10 PRINT "12345678901234567890123456789012"
20 LET l$="ian_feb_mar_apr_mai_iun"
30 PRINT l$
40 PRINT "luna 2-a:"; l$(4 TO 7)
50 PRINT "trim.1:"; l$(1 TO 11)
60 PRINT "primavara incepe in luna"; l$(8 TO 11)
70 PRINT "semestrul 1:"; l$
RUN
```

#### NOTAȚIA EXPONENȚIALĂ A NUMERELElor

Numerele foarte mari și cele foarte mici pot fi exprimate sub o formă exponentială astfel:

$1234000000 = 1.234 \times 10^9$

mantisa      exponent (se poate nota cu "E" sau "e")

Mantisa reprezintă numărul zecimal (în locul virgulei se folosesc punctul), iar exponentul este format din litera "E" urmată de un număr care arată poziția punctului zecimal. Dacă exponentul este pozitiv, numărul respectiv se obține prin deplasarea spre dreapta a punctului zecimal

3.7e12 = 3700000000000

12 poziții

## CAPITOLUL 4

Dacă exponentul este negativ, numărul respectiv se obține prin deplasarea spre stînga a punctului zecimal.

56e-8=0,00000056  
8 poziții ←

Exemplul următor vă va ajuta să înțelegeți mai bine reprezentarea exponentială:

31400	3.14e4
3140	3.14e3
314	3.14e2
31,4	3.14e1
3,14	3.14
0,314	3.14e-1
0,0314	3.14e-2
0,00314	3.14e-3
0,000314	3.14e-4
0,0000314	3.14e-5

Dacă CIP-ul vă va afișa un rezultat sub formă exponentială, dvs. veți putea obține forma zecimală obișnuită destul de ușor, luând în considerare semnul exponentului. Dacă este + scrieți punctul zecimal la dreapta cu atită poziții cîte arată exponentul.

9.876543E8 este 987654300  
8 poziții ←

Dacă semnul este negativ, scrieți punctul zecimal la stînga cu numărul indicat de poziții.

23e-6 este 0,000023  
6 poziții ←

### OPERATORI RELATIONALI

Un program BASIC se execută "secvențial" linie după linie, dar aveți posibilitatea să-i spuneți calculatorului, ca în anumite condiții să nu mai respecte secvența crescătoare a numerelor de linii. Pentru aceasta veți putea folosi instrucțiunea de testare condiționată IF...THEN... (în traducere: DACĂ...ATUNCI...).

IF condiție THEN acțiune

Cînd condiția este îndeplinită se execută acțiunea indicată după THEN.

Condiția (sau testul) care urmează după IF o puteți pune folosind operatori relationali.

operator	semnificație	exemple (vezi anexa A)	
>	mai mare decît	4>3	b>a
<	mai mic decît	3<4	a<b
>=	mai mare decît sau egal cu	x>=y	n>=8
<=	mai mic decît sau egal cu	p<=q	t<=10
<>	nu este egal cu (diferit de)	5<>7	m<>n
=	egal cu	K\$="ABC"	x=4

### OPERATORI LOGICI

În instrucțiunea IF veți putea utiliza operatori logici:  
AND (în traducere: și)  
OR (SAU)  
NOT (NU)

#### AND

Folosind AND între două condiții, impuneti ca amîndouă să

fie îndeplinite pentru ca să se execute ce urmează după THEN

IF condiția 1 AND condiția 2 THEN...

Pentru a înțelege mai bine, închipuiți-vă o țeavă prin care curge apa și care are două robinete R1 și R2:

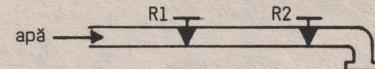


Fig.4.1

Ele pot fi în următoarele condiții:

R1	R2	Efect
închis	închis	nu curge apa
închis	deschis	nu curge apa
deschis	închis	nu curge apa
deschis	deschis	apa curge

Condițiile se pun deci astfel:

DACĂ R1=deschis și R2=deschis ATUNCI apa curge

În BASIC veți scrie:

IF R1=.....AND R2=.....THEN.....

#### OR

Operatorul logic OR îl veți folosi atunci cînd din două condiții este suficient una să fie îndeplinită, pentru ca să se execute ce urmează după THEN.

Închipuiți-vă acum țeava de apă ramificată astfel:

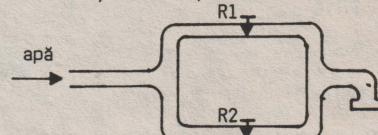


Fig.4.2

Se observă imediat că un singur robinet este suficient să fie deschis pentru ca apa să curgă.

Condițiile se pun astfel:

DACĂ R1=deschis SAU R2=deschis ATUNCI apa curge

În BASIC veți scrie:

IF R1=.... OR R2=....THEN.....

#### NOT

Operatorul NOT se folosește înaintea unei condiții care dacă NU este îndeplinită, se execută ce urmează după THEN.

Exemplu: DACĂ NU este x diferit de y ATUNCI scrie x=y  
IF NOT condiție THEN.....

### ALIPIREA (CONCATENAREA) ȘIRURILOR

Puteți reunii două șiruri de caractere într-unul singur, folosind operatorul de concatenare (alipire) "+".

a\$=b\$+c\$

Presupunind că variabilele și r sint:

LET b\$="Întreprinderea"  
LET c\$=" ELECTRONICA"

după executarea liniei: LET a\$=b\$+c\$:PRINT a\$

se va afișa pe ecran: Întreprinderea ELECTRONICA

## ÎNTREBĂRI RECAPITULATIVE ȘI EXERCIȚII

Î 4.1 În execuția operațiilor aritmetice există o ordine (prioritate). Scrieți în ordine inversă ce operații se execută:

- 4) \_\_\_\_\_ (ultimele)  
 3) \_\_\_\_\_  
 2) \_\_\_\_\_  
 1) \_\_\_\_\_ (primele)

Î 4.2 Scrieți forma BASIC pentru calculul expresiilor:

$$\begin{aligned} 1) \quad d &= \frac{a \times b}{c} + 10 \\ 2) \quad e &= \frac{(a+5) \times b}{2,7(c-d)+1} \\ 3) \quad y &= ax^2+bx+c \end{aligned}$$

Î 4.3 Găsiți erorile în următoarele constante:

constantă	eroare
a) 4.320E2.1	-----
b) 1,05	-----
c) 5.666E+42	-----
d) 0.702-E6	-----

Î 4.4 Ce rezultat va fi afișat după executarea următorului program:

10 LET A=5:LET B=8:LET C=(A\*B+2)/7:PRINT C

Î 4.5 Cum vor fi afișate următoarele numere:

- a) 1200000000  
 b) 0,000000444 \_\_\_\_\_

Î 4.6 Care este forma zecimală obișnuită a următoarelor constante numerice:

- a) 3.001E+10  
 b) 1.989E-6 \_\_\_\_\_

Î 4.7 Date fiind următoarele variabile: M,M\$,Abc,P123,X\$,șir,doi,S\$ precizați care sint:

- a) variabile numerice:  
 b) variabile şir: \_\_\_\_\_

Î 4.8 Ca urmare a instrucțiunii:

```
IF a<=13 AND b>5 THEN PRINT x
se afișează x dacă:
1) a=13;b=13 _____ (răspundeți da/nu)
2) a=20;b=20 _____ (da/nu)
```

Î 4.9 Dacă introduceți:

```
10 LET D$="am CIP"
20 IF b$="știu BASIC" OR D$="am CIP"
THEN PRINT "sint bucuros!"
30 PRINT "regret!"
```

după execuția programului veți primi răspunsul "sint bucuros!?"?

a) \_\_\_\_\_ (da/nu)

b) De ce? \_\_\_\_\_

Î 4.10 Atenție o întrebare puțin mai dificilă!

Judecați cu atenție și încercați să răspundeti, apoi introduceți programul și verificați-vă raționalul initial. Fie următorul program:

```
5 PRINT
10 INPUT "x=";x;"y=";y
20 PRINT "x=";x;"y=";y
30 IF NOT x<>y THEN PRINT "AAAA":GOTO 5
40 PRINT "BBBB"
```

Ce mesaj va fi afișat dacă introducem:

- a) x=4 y=1 \_\_\_\_\_  
 b) x=8 y=8 \_\_\_\_\_  
 c) x=1 y=4 \_\_\_\_\_

Î 4.11 Fie următorul program:

```
10 PRINT "introduceți pentru y valoarea 1,2 sau 3"
20 INPUT "y=";y
30 IF y=1 THEN GOTO 70
40 IF y=2 THEN GOTO 80
50 IF y=3 THEN GOTO 90
60 GOTO 100
70 PRINT "ROȘU":GOTO 20
80 PRINT "ALB":GOTO 20
90 PRINT "VERDE":GOTO 20
100 STOP
```

- a) Ce se va afișa după execuție dacă:  
 y=2 \_\_\_\_\_  
 y=1 \_\_\_\_\_  
 y=3 \_\_\_\_\_

b) Care sint în program constantele numerice?

c) Care sint variabilele numerice?

d) Care sint constantele şir?

e) Ce se întimplă dacă introduceți pentru y o valoare diferită de 1, 2 sau 3 ?

## RĂSPUNSURI

R 4.1 4) adunarea/scăderea  
 3) înmulțirea/împărțirea  
 2) exponențierea  
 1) expresiile din paranteze

R 4.2 1)  $d = (axb)/c+10$   
 2)  $e = ((a+5) \times b) / (2.7 \times (c-d/b)+1)$   
 3)  $y = ax^2+bx+c$

R 4.3 a) exponentul nu este număr întreg  
 b) virgula nu este admisă  
 c) exponent prea mare (>37)  
 d) exponent gresit (trebuie:E-6)

R 4.4 6

R 4.5 a) 1.2E+9  
 b) 4.44E-7

R 4.6 a) 30010000000  
 b) 0,000001989

R 4.7 a) M, Abc, P123, şir, doi  
 b) M\$, X\$, S\$

## CAPITOLUL 4

R 4.8 1) da      2) nu

R 4.9 a) da!  
b) pentru că OR cere să fie îndeplinită o singură condiție

R 4.10 Linia 30 a paragrafului se citește așa:

DACĂ NU este x diferit de y (adică  $x=y$ ) afișează "AAAAA" și întoarce-te la linia 5.  
În cazul în care condiția pusă în linia 30 nu este îndeplinită (adică x este diferit de y) programul trece la linia 40 și afișează mesajul corespunzător.  
Deci răspunsurile corecte sunt:

- a) B888
- b) AAAA
- c) BBBB

R 4.11 a) ALB  
ROȘU  
VERDE  
b) 1, 2 și 3  
c) y  
d) "ROȘU", "ALB", "VERDE"  
e) execuția programului este opriță prin STOP (linia 100)

## CAPITOLUL 5

### INTRĂRI, IEȘIRI, APLICAȚII SIMPLE

LET; INPUT; READ; PRINT; PRINT TAB; PRINT AT; REM; SAVE; LOAD

În acest capitol cunoștințele dvs. de BASIC vor fi îmbunătățite, prin detalii asupra posibilităților de introducere a datelor, de obținere a rezultatelor și prin exemple de programe comentate.

Treptat, vă veți însuși noțiuni despre realizarea unor programe proprii bine gîndite și clare, despre stocarea lor pe casete magnetică și despre o operație pe care o veți executa foarte frecvent și anume încărcarea unor jocuri/programe de pe casetele magnetice.

1. Anulați programul existent în memorie. Aveți două posibilități:

- prin acționarea tastei RESET
- prin instrucțiunea NEW

2. Introduceți și execuția următorul program:

```

10 INPUT "a=";a
20 INPUT "b=";b
30 INPUT "c=";c
40 LET d=a+b+c
50 PRINT d
60 STOP
RUN

```

Programul dispare de pe ecran și cursorul clipește întrebător în colțul ecranului așteptind o valoare pentru variabila numerică "a".

```

a=2
b=4
c=6

```

Scrieți ce se afișează:

-----

Rețineți că cele trei instrucțiuni INPUT au permis introducerea prin interogare, de valori pentru variabilele numerice a, b și c.

3. Anulați liniile 10 și 20 și introduceți din nou comanda de execuție

```

10 <ENT>
20 <ENT>
RUN <ENT>
c=?

```

Notați mesajul afișat: -----

CIP-ul a încercat fără succes să execute adunarea din linia 40 și a semnalat că nu a găsit variabilele "a" și "b".

4. Schimbați linia 30 punând toate variabilele, separate prin virgule:

30 INPUT "a=";a,"b=";b,"c=";c

Aveți programul:

```

LIST
30 INPUT "a=";a,"b=";b,"c=";c
40 LET d=a+b+c
50 PRINT d
60 STOP
RUN

```

De această dată cursorul clipește cerind valoarea lui "a", după care sare la mijlocul liniei și cere pe "b" și apoi sare în linia următoare cerind pe "c".

```

a=10          b=4.3
c=5.7

```

Rezultat: -----

5. Introduceți:

30 READ a,b,c (READ obțineți cu SS+CS și apoi A)  
LIST

Veți avea pe ecran:

```
30 READ a, b, c
40 LET d=a+b+c
50 PRINT d
60 STOP
RUN
```

Notați mesajul afișat: -----

Explicația este următoarea: instrucțiunea READ este o altă posibilitate de a introduce date, direct în program sub formă de constante numerice fixe.  
Deocamdată neavând aceste date, execuția liniei 30 READ... a generat mesajul de mai sus, care vă atrage atenția că nu mai sunt date.

6. Acum introduceți:

```
35 DATA 2, 4, 6
RUN
```

Având datele cunoscute, CIP-ul execută fără ezitare adunarea și va comunica rezultatul..

7. Anulați linia 35 și introduceți linia 10:

```
35 <ENT>
10 DATA 2,4,6
<ENT>
10 DATA 2,4,6
30 READ a,b,c
40 LET d=a+b+c
50 PRINT d
60 STOP
RUN
```

Observați că deși linia DATA a fost aplicată în altă parte în program (chiar și înaintea instrucțiunii READ) BASIC-S acceptă datele și execută programul.

8. Anulați programul anterior și introduceți:

```
10 READ x,y
20 LET z=x/y
30 PRINT z
40 GO TO 10
50 DATA 10,2,9,3,40,4
60 STOP
RUN
```

Notați ce se afișeză: ----  
----  
----  
-----

Instrucțiunea READ a "citit" cîte două valori din DATA, în linia 20 a făcut împărțirea, linia 30 a determinat afișarea rezultatului și totul a fost reluat cu altă pereche de date pînă cînd acestea au fost epuizate.

9. Anulați linia 50 și introduceți:

```
5 DATA 36,6
45 DATA 64,32
70 DATA 35,5
RUN
```

Se afișeză: 6  
2  
7  
Out of DATA,10:1

Puteți pune deci mai multe instrucțiuni DATA într-un program? (DA/NU) --- probabil că ați răspuns corect: da! pot fi mai multe instrucțiuni DATA amplasate oriunde în program.

10. Anulați programul și introduceți:

```
100 LET G=15
110 PRINT G
120 STOP
RUN
```

Notați rezultatul: -----

11. Schimbați linia 110:

```
110 PRINT "G"
```

RUN  
Ce se afișează acum ca rezultat? -----

De reținut:

-PRINT constantă numerică	-afișează valoarea constantei
-PRINT "constantă sir"	-afișează conținutul constantei sir

12. Dacă introduceți:

```
110 PRINT "greutatea";G
```

Ce credeți că se va afișa după execuție? -----  
Execuția programul și verificați răspunsul dvs.

13. Anulați programul anterior și introduceți:

```
10 READ x
20 PRINT x
30 GO TO 10
40 DATA 20,21,22,23
```

Execuția programul și notați rezultatul: -----  
-----  
-----  
-----

14. Schimbați:

```
20 PRINT x,
```

Observați că apare o virgulă după x. Execuția programul și notați din nou rezultatul:  
-----  
-----  
-----

15. Mai schimbați linia 20 astfel:

```
20 PRINT x; ( ; după x)
RUN
```

Notați modul de afișare: -----

Rețineți deci modul de afișare diferit după cum este încheiată instrucțiunea PRINT:

- cu virgulă: se afișează pe două coloane (la marginea și la mijlocul ecranului);
- cu punct și virgulă: se afișează rezultatele unul alături de celălalt;
- fără virgulă/punct și virgulă: se afișează rezultatele unul sub altul

16. Adăugați și schimbați în program:

```
5 PRINT "01234567890123456789012345678901"
20 PRINT TAB 3;x
RUN
```

.Linia 5 a numerotat cele 32 (0-31) poziții ale unei liniilor de afișare;  
.În linia 20 s-a introdus TAB 3, care impune afișarea rezultatelor începînd cu poziția 3.

Observeți pe ecran obținerea celor de mai sus.

17. Să schimbăm poziția de afișare și datele:

```
20 PRINT TAB 15;x
40 DATA 8,100,3210,9,6
```

Rezultatele sănt afișate începînd cu coloana 15.

## CAPITOLUL 5

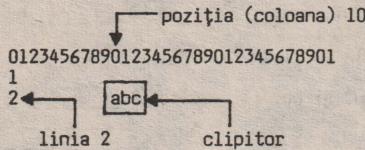
18. Adăugați și schimbați în program:

```
8 PRINT 1:PRINT 2 ← numerotează la margine linia 1  
     și linia 2  
10 READ x$  
20 PRINT AT 2,10;FLASH1;x$  
    ↗ afișează clipitor  
    ↗ afișează începind cu linia 2, poziția (coloana) 10  
30 DATA "abc"  
40 <ENT> - pentru a șterge linia 40.
```

Programul trebuie să arate astfel:

```
5 PRINT "01234567890123456789012345678901"  
8 PRINT 1:PRINT 2  
10 READ x$  
20 PRINT AT 2,10;FLASH1;x$  
30 DATA "abc"
```

Executând programul veți obține următorul rezultat:



### Noțiuni de bază

Datele care sunt primite de un program și prelucrate într-un anumit fel, sunt denumite DATE DE INTRARE.  
Basic-S permite trei moduri diferite de introducere a datelor, folosind instrucțiunile: LET, INPUT, READ, DATA

#### LET

Instrucțiunea LET atribuie unei variabile "v", valoarea unei expresii "e" Sintaxa comenzi: LET v=e

Exemplu: LET a=15 se citește "fie a egal cu 15" și de fapt atribuie variabilei "a" valoarea 15.

Exemple de instrucțiuni LET: LET d=14.3

```
LET L$ = "APRILIE"  
LET x=(p+m)/n  
LET k=k+1
```

Ultimul exemplu pare imposibil dar îl puteți înțelege analizând următorul program:

```
10 LET k=5  
    ↗ 20 PRINT k;  
    ↗ 30 LET k=k+1  
    ↗ 40 GO TO 20
```

După execuție se afișează: 5678910...

Initial k ia valoarea 5 care este afișată. La execuția liniei 30, valoarea anterioară este crescută cu 1, astfel încât a doua oară va fi afișată valoarea 6 etc.

#### INPUT

Instrucțiunea INPUT (în traducere: introduceți) permite introducerea datelor direct de la tastatură.

Sintaxa: INPUT "c";v1;v2  
.c este un sir optional de caractere care explicitează datele;  
.v1,v2... sint nume de variabile numerice/sir de caractere.

Spre deosebire de LET, instrucțiunea INPUT permite deci introducerea conversațională a datelor. Cind CIP-ul o întâlnește, oprește execuția programului și cursorul clipește întrebător pînă cînd dvs. răspundeți prin:

- introducerea datelor solicitate, după care execuția programului continuă
- introducerea instrucțiunii STOP în vederea renunțării la execuție.

Exemple:

```
INPUT x - introduce pentru variabila x, valoarea tastată  
INPUT "a=?";a - afișează a=? și introduce valoarea tastată;  
INPUT a;b;c - solicită pe rînd valori pentru a,b și c
```

Puteți deci cere introducerea datelor pentru mai multe variabile printr-o singură instrucțiune INPUT. Variabilele pot fi separate prin punct și virgulă sau prin virgulă:

```
INPUT m;n - cere date pentru m și cursorul rămîne în aceeași poziție.  
INPUT m,n - cere date pentru m și apoi sare în coloana 15 pentru n
```

#### READ

Instrucțiunea READ (în traducere: citește) atribuie unor variabile v1, v2, ... valoarea unor expresii/date succesive din listele instrucțiunilor DATA. Sintaxa: READ v1, v2...

#### DATA

Instrucțiunea DATA permite introducerea valorilor pentru variabile. Sintaxa: DATA c1, c2,...

Exemple:

```
READ m,n,p$,q$  
DATA 25,1703,"cuvînt","CIP"
```

Pentru a înțelege mai bine ce efect au cele două instrucțiuni, încercați exemplul de mai sus astfel:

```
10 DATA 25,1703,"cuvînt","CIP"  
20 READ m,n,p$,q$  
30 PRINT m,n,p$,q$  
RUN
```

Instrucțiunea READ diferă de INPUT prin faptul că programul nu se oprește să aștepte ca datele să fie introduse prin tastatură, ci le caută automat în instrucțiunea DATA. Instrucțiunea DATA poate fi plasată oriunde în program, chiar și înaintea instrucțiunii READ.

Se obișnuiește ca toate instrucțiunile DATA să fie puse la sfîrșitul programului, înaintea instrucțiunii STOP. Pot fi mai multe instrucțiuni DATA și nu este necesar să fie grupate în program. Instrucțiunea READ citește pe rînd valori pentru variabilele specificate, din DATA începînd cu numărul de linie cel mai mic. Dacă numărul de variabile depășește pe cel al datelor se afișează mesajul: Out of DATA și programul este oprit.

#### RESTORE

Instrucțiunea RESTORE (în traducere: restabilește!) permite folosirea același set de date, de către mai multe instrucțiuni READ. Sintaxa: RESTORE n

Un indicator de citire este restabilă la instrucțiunea DATA cu număr de linie n, astfel încât următoarea instrucțiune READ va începe citirea de la acea linie.

Exemplu:

```
10 READ a,b,c - citește valori din liniile 50 și 60  
20 RESTORE 50 - restabilește ca linie de citire  
     linia 50  
30 READ z,x - va citi din nou linia 50  
40 PRINT a'b'c'z'x' (introduceți apostrof între variabile)  
50 DATA 1,2  
60 DATA 3,4  
70 DATA 5,6 ← de aici nu citește  
80 STOP  
RUN
```



## CAPITOLUL 5

### EXEMPLE DE PROGRAME

Stimate cititor, consider că sunteți suficient de pregătit să începeți să analizați și să scrieți programe din ce în ce mai complicate.

În acest capitol vom analiza cîteva exemple, vom enunța altele pe care să le realizați singuri.

Curaj! veți constata că ați reținut mai multe cunoștințe decât vă închipuiți.

Mai întîi să recapitulăm:

- un program este format din instrucțiuni
- o linie de program poate conține una sau mai multe instrucțiuni separate prin ":"
- fiecare linie trebuie numerotată
- instrucțiunea REM se folosește pentru comentarea (documentarea) programului
- pentru introducerea datelor se folosesc instrucțiunile:  
.LET  
.INPUT  
.READ...DATA (RESTORE)
- pentru calcul se folosesc instrucțiunile:  
.LET  
.PRINT
- pentru afișarea rezultatelor se folosesc instrucțiunile:  
.PRINT  
.PRINT AT  
.PRINT TAB

Iar acum cîteva sugestii, pentru ca să realizați programe bune și... frumoase. Este indicat să urmați următoarele faze:

- analizarea problemei în vederea stabilirii:  
.intrărilor  
.prelucrărilor  
.ieșirilor
- proiectarea programului
- codificarea în libaj BASIC
- testarea și punerea la punct a programului.

Știu că sunteți nerăbdător să faceți cît mai repede programe, dar nu săriți rîndurile următoare prin care vă semnalează cîteva greșeli făcute de începători. Unele greșeli sunt simple, cele făcute chiar la introducere (tastare greșită, erori de sintaxă) și vîrind nevrînd va trebui să le corectați pentru că CIP-ul vigilant nu le acceptă.

Sunt însă erori mai subtile, de logică, și acestea sunt sesizabile doar prin obținerea unor rezultate necorespunzătoare. Pentru ele trebuie să faceți o "depanare" a programului, acțiune despre care vom discuta în alt capitol.

Pentru evitarea greșelilor de introducere vă recomand:

- respectați sintaxa cerută de BASIC
- tastați <ENT> la sfîrșitul fiecărei linii de program
- nu tastați litera 0 în loc de cifra Ø (zero)
- fiți atenți la numărul de linie, să nu îl repetați pe cel al uneia existente, pentru că aceasta va dispărea
- anulați vechiul program prin NEW sau RESET!

### Exemplul 5.1

Să se realizeze un program pentru calculul unei expresii matematice date:

Intrări: a,b,c,m,n - variabile numerice introduse prin tastatură în coloanele 0 și 16.

$$\text{Prelucrări: } x = \frac{[2(a+b)^2 + 3c^2]}{m+n+1}$$

Ieșiri: x = valoare lui x.

Se dorește deci nu numai afișarea valorii rezultatului, ci și unei forme explicite: x = rezultat

Algoritmul problemei:

- 1.solicitarea și memorarea datelor de intrare
- 2.calculul expresiei
- 3.afișarea rezultatului
- 4.stop

### Codificarea programului:

```
10 REM nume program PRG5.1
20 REM program pentru calculul unei expresii matematice
30 REM introducerea datelor
40 INPUT a;b;c;m;n
50 REM calculul expresiei
60 LET x=(2*(a+b)^2+3*c^2)/(m+n+1)
70 REM afișarea rezultatului
80 PRINT "x=";x
90 STOP
```

### Două precizări:

1. - la expresiei cu paranteze trebuie să verificați ca numărul parantezelor deschise "(" să fie egal cu cel al parantezelor închise ")".
2. - dacă vă aflați în execuția unei instrucțiuni INPUT și dintr-un motiv oarecare dorîți să întrerupeți introducerea, singura posibilitate este de a șterge înapoi cu DELETE pînă se oprește cursorul și să dați STOP <ENT>.

Introduceți programul, notați pe o hîrtie datele introduse și rezultatul: x=\_\_\_\_\_

Schimbați linia 60 : 60 LET x=(2\*(a+b)^2+3\*c^2)/(m/(n+1))

Execuțați programul cu aceleași date și notați din nou rezultatul: x=\_\_\_\_\_

Observați că rezultatele diferă deși nu v-a fost semnalată nici o eroare la introducere. De ce diferă rezultatele? Care este cel bun? (nu calculați cu creionul, ci analizați programul)

### Exemplul 5.2.

Să se scrie un program pentru calculul sumei totale a unor cantități date.

- Intrări: c=cantitatea, cerută sub formă explicită;  
cantitate = valoarea ce se introduce
- Ieșiri: total - suma cantităților introduse pînă în acel moment
- Afișarea să se facă în două rubrici denumite CANTITATEA și TOTAL decalate astfel:

	coloana 0	coloana 16
	CANTITATEA	TOTAL
	=====	=====
de ex.	3	.-- subliniere
	2	.-- prima linie cu date
		3 -- pe altă linie
		5 -- (3+2)

#### - Prelucrări:

TOTAL = 0 - valoarea inițială a variabilei TOTAL  
TOTAL = TOTAL + c - se insumează totalul anterior cu cantitatea nou introdusă

#### - Codificarea programului:

```
10 REM nume program PRG5.2
20 REM program pentru însumarea unor date
30 REM afișarea capului de tabel
40 PRINT "CANTITATEA","TOTAL"
50 PRINT "=====","====="
60 REM solicitarea cantității
70 INPUT "cantitatea=";c
80 REM calculul totalului
90 LET total = total + c
100 REM afișarea cantității și a totalului
110 PRINT c';total
120 REM solicitarea altiei cantități
130 GO TO 70
RUN
```

#### Ce s-a întipplat?

A fost afișat capul de tabel, dar sunteți avertizați că nu a fost găsită variabila în linia 90. Deoarece variabila c tocmai ați introdus-o rezultă că nu a fost găsită variabila total.

CIP-ul "v-a prins" cu o greșală de logică. I-ați cerut ceva pe care ați uitat să i-o definiți mai înainte.  
Corectați greșală introducând ceea ce de fapt ați prevăzut la prelucrări: decalarea și inițializarea variabilei "TOTAL".

```
25 REM inițializarea variabilei de totalizare
28 LET total = 0
RUN
```

Acum programul se execută. Introduceți cîteva cantități și verificați:

CANTITATE	TOTAL
4	4
6	10
185	195
1000.5	1195.5
2E+9	2.0000012E+9

La ultimul total, numărul fiind foarte mare, a fost rotunjit pentru a putea fi afișat cu 8 cifre:

```
2.0000011955 -----> 2.0000012
```

### SALVAREA PROGRAMELOR PE CASETĂ

Deoarece în continuare vă voi propune să realizați singuri programe pe "teme" date, veți dori eventual să le păstrați și după oprirea calculatorului ca să le revedeți altădată. Pentru aceasta va fi nevoie să le salvați pe o casetă magnetică (memoria externă a CIP-ului), de unde să le încărcați din nou în memorie oricănd doriti.  
Pregătiți deci casetofonul, o casetă (attenție, ce veți înregistra va șterge ceea ce aveți deja pe casetă).  
Puneți cablu de legătură între CIP și casetofon.

### Înregistrarea programelor pe casetă

Cind doriți să salvați un program procedați astfel:  
a) - fiind în mod comandă (cursor K) introduceți comanda:

```
SAVE "nume program" LINE număr <ENT>
```

Exemplu:

```
SAVE "PRG5.2" LINE 10 <ENT>
```

Obțineți mesajul: Start tape, then press any key  
adică: Porniți banda, apoi apăsați orice tastă

b) - fără să porniți banda apăsați orice tastă.

L-ați "păcălit" pe CIP, el începe transmiterea programului către casetofon și puteți astfel regla nivelul de înregistrare. Acesta este bine să fie cît mai mare, chiar peste limita recomandată pentru muzică.

c) - mai introduceți odată comanda

```
SAVE "nume program" LINE număr <ENT>
```

și acum ascultați-i sfatul, pornind intîi banda și după cîteva secunde (pentru a se crea un spațiu pe bandă între programe) apăsați orice tastă. Dacă totul merge normal obțineți confirmarea prin OK (totul în ordine).

Un program salvat numai cu:

```
SAVE "nume program" <ENT>
```

la încărcarea de pe casetă va fi lansat în execuție prin comanda RUN, spre deosebire de cazul în care introduceți în comanda SAVE numărul primei linii din program (ca în exemplul de mai sus), cind programul va fi lansat automat în execuție.

Alte forme ale comenzi SAVE le găsiți în anexa C.

### VERIFY

O deprindere bună este aceea de a verifica de fiecare dată, dacă salvarea s-a făcut corect. Pentru aceasta:

- a) - derulați banda la începutul înregistrării
  - b) - introduceți comanda: VERIFY "nume program" <ENT>
  - c) - porniți banda
- Dacă programul a fost înregistrat corect, pe ecran apare: Program nume... și la sfîrșitul verificării se afișează Ø OK

Mesajul : " R Tape loading error" indică o eroare de înregistrare

### Încărcarea programelor de pe casetă

Așa cum v-am spus CIP-ul prezintă un mare avantaj, acela de a putea executa orice program elaborat pentru HC 85/TIM-S/SINCLAIR-SPECTRUM.

Programele se difuzează pe casete, aşa încît veți fi frecvent în situația de a avea niște jocuri sau programe formidabile pe o casetă și veți fi nerăbdători să le executați pe CIP-ul dvs.

Pentru încercarea unui program de pe casetă procedați astfel:

- a) - introduceți
- LOAD " " - dacă dorîți să fie încărcat primul program care urmează pe bandă

LOAD "nume program" - pentru a fi căutat și încărcat programul cu numele specificat în comandă

- b) - porniți banda

Dacă totul este în ordine, pe ecran se afișează numele programului și urmează dungile clipitoare însotite de un sunet specific.

- c) - opriți banda la apariția mesajului "OK" sau a celui dat de program.

Dacă după pornirea benzii nu apar numele programului și dungile orizontale, sau dacă apare mesajul: "Tape loading error" opriți banda, derulați-o rapid înapoi și reluați secvența de încărcare. După 2-3 încercări nereușite, puteți renunța la încărcarea acelui program, deoarece poate fi una din următoarele cauze:

- programul a fost salvat cu un nivel de înregistrare necorespunzător;
- casetofonul dvs. nu are capul de citire aliniat la fel cu cel pe care s-a făcut salvarea;
- banda este deformată (întinsă, cutată) sau zgâriată.

O ultimă soluție pe care o puteți încerca este aceea de a utiliza alt casetofon.

### Întrebări recapitulative și exerciții

I 5.1 Ce se va afișa dacă se execută următorul program?

```
10 LET x=1
20 PRINT x,
30 LET x=x+1
40 GO TO 10
```

Scrieți primele 12 valori rezultate și apoi executați programul pentru a vă verifica (întrerupeți cu BREAK).

I 5.2 Care este scopul instrucțiunii REM?

I 5.3 Dacă introduceți într-un program instrucțiunea READ, ce altă instrucțiune trebuie să utilizați obligatoriu?

I 5.4 Ce va fi afișat dacă se execută următorul program?

```
10 LET a=6:LET b=13:PRINT "b=";a
```

I 5.5 Completati valorile lipsă:

În afișarea alfanumerică ecranul este considerat împărțit în --- coloane și --- linii.

I 5.6 Ce instrucțiune folosiți pentru a afișa începînd din diferite coloane?

## CAPITOLUL 5

I 5.7 Diferă afişarea rezultatelor prin liniile 30, 40 și 50 din următorul program?

```
10 LET a=7
20 LET b=4
30 PRINT a
40 PRINT b
50 PRINT a b
```

I 5.8 Ce tastați pentru a afișa programul dacă CIP-ul aşteaptă un răspuns după o instrucțiune INPUT?

I 5.9 Ce instrucțiune permite utilizarea același set de date, de către mai multe instrucțiuni READ?

I 5.10 Scrieți în format BASIC expresiile:

$$a) 2(x + 3x^2)$$

$$b) x = \frac{5,4 + y\sqrt{u+v}}{2/y}$$

I 5.11 Scrieți instrucțiunile BASIC pentru:

- a) a crește c7 cu 0,01
- b) a da valoarea "xeroarexx" variabilelor T\$ și U\$.

I 5.12 Ce erori există în următorul program:

```
10 INPUT l,h
20 a=l*h
30 PRINT l,h,a,p
40 LET p=2(l+h)
```

### Probleme

1) Scrieți un program pentru calculul suprafeței (S) și volumului (V) al unei sfere cu raza r. Rezultatele să fie afișate astfel:

$$r = \underline{\quad} \quad S = \underline{\quad} \quad V = \underline{\quad}$$

2) Scrieți un program pentru calculul volumului unei cutii paralelipipedice având lungimea L, lățimea l și înălțimea h. Datele de intrare să fie cerute astfel:

Lungimea (cm)?

Lățimea (cm)?

Înălțimea (cm)?

Rezultatele să fie afișate astfel:

L = cm

l = cm

h = cm

Volumul este centimetri cubi

### Răspunsuri

R 5.1	1	2
	3	4
	5	6
	7	8
	9	10
	11	12

R 5.2 Instrucțiunea REM permite introducerea unor comentarii care să ajute la înțelegerea programelor.

R 5.3 DATA

R 5.4 b=6

R 5.5 32 coloane (0-31)  
22 linii (0-21)

R 5.6 PRINT TAB

R 5.7 Nu! Apostroful din linia 50 provoacă afișarea rezultatelor la început de rînd, unul sub altul.

R 5.8 1. STOP <ENT>
 2. <ENT>

R 5.9 RESTORE

R 5.10

$$a) 2(x_1 + 3x_2)$$

$$b) x = (5,4 + y(u+v)) / (a+b)^2/y$$

R 5.11

$$a) \text{LET } C7 = C7 + 0,1$$

$$b) \text{LET } T\$ = "xeroarexx"$$

$$\text{LET } U\$ = "xeroarexx"$$

R 5.12

- în linia 20 lipsește LET
- linia 30 trebuie să devină 50 pentru că p nu este cunoscut
- corect se scrie: LET p = 2\*(1+h)

## CAPITOLUL 6

### CUM INTRODUCEM CONDIȚII SAU ALTERNATIVE ÎN PROGRAM ?

Adeseori veți simți nevoie în programe să determinați CIP-ul să nu mai execute instrucțiunea care urmează, ci să execute un salt la o instrucțiune aflată peste un număr de linii înainte sau înapoi.

Un asemenea salt poate fi introdus în program, să fie executat în anumite condiții sau necondiționat.

#### IF...THEN GO TO

Un utilizator necunoscător în programarea calculatoarelor va avea impresia că CIP-ul ia "decizii" în anumite momente, dar de fapt acestea sunt introduse de autorul programului/jocului sub forma unor alternative condiționate sau nu.

1. Introduceți următorul program:

```
NEW  
110 LET a=1  
120 PRINT a  
130 LET a=a+1  
140 IF a<8 THEN GO TO 120  
150 STOP
```

Linia 140 o citiți astfel: "Dacă a este mai mic decât 8 ATUNCI MERGI (sari) la linia 120".

Analizați cu atenție programul și scrieți pe o hârtie ce credeți că va fi afișat.

Executeți programul și verificați-vă.

2. Schimbați linia 110: 110 LET a=6  
Ce se va obține? Executeți programul și verificați!

3. Schimbați și introduceți:

```
130 LET a = a + 10  
140 IF a>=46 THEN GO TO 150  
145 GO TO 120  
RUN
```

Acum linia 140 a comandat CIP-ului: "Dacă a este mai mare decât sau egal cu 46 ATUNCI MERGI la linia 150".

CIP-ul ascultător așa va face, dar cît timp a este mai mic decît 46 va executa linia următoare (145) care îl trimite înapoi la linia 120.

Analizați programul și rezultatele pentru a înțelege ce s-a întîmplat.

4. Completăți semnele prin care veți pune în program condiția pentru (v. capitolul 4):

mai mare decît \_\_\_\_\_  
mai mic decît sau egal cu \_\_\_\_\_  
egal cu \_\_\_\_\_  
mai mare decît sau egal cu \_\_\_\_\_  
neegal cu (diferit de) \_\_\_\_\_

5. Analizați cu atenție următorul program:

```
10 PRINT "introduceți 2, 4 sau 6"  
20 INPUT "n="; n  
30 IF n=2 THEN GO TO 90  
40 IF n=4 THEN GO TO 70  
50 IF n=6 THEN GO TO 80  
60 GO TO 100  
70 PRINT "SCUFITĂ ROȘIE": GO TO 20  
80 PRINT "HARAP ALB": GO TO 20  
90 PRINT "PINOCCHIO": GO TO 20  
100 STOP
```

Ce valoare trebuie să introduceți pentru ca să se afișeze:

HARAP ALB	n= _____
PINOCCHIO	n= _____
SCUFITĂ ROȘIE	n= _____

Anulați vechiul program, introduceți pe cel de mai sus și executeți-l cu valorile anticipate de dvs. verificându-vă!

6. Programul cere să se introducă pentru n valorile 2, 4, 6. Ce credeți că se va întîmpla dacă introduceți altă valoare? Formulați răspunsul și verificați!

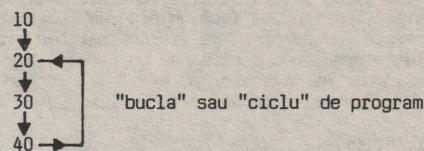
#### NOȚIUNI DE BAZĂ

GO TO Instrucțiunea GO TO produce un salt necondiționat la linia indicată. Sintaxa: GO TO număr linie.

Fie de exemplu următorul program:

```
10 LET x=5  
20 PRINT x  
30 LET x=2*x  
40 GO TO 20  
50 STOP
```

Un program BASIC este executat începînd cu linia avînd cel mai mic număr și continuă linie cu linie. Instrucțiunea GO TO (în traducere: "mergi la") permite întreruperea acestei secvențe printre-un transfer /salt la o linie de program specificată. Programul de mai sus se execută în următoarea ordine a liniilor:



Deoarece nu este pusă nici o condiție în linia 40, GO TO constituie o instrucțiune de salt necondiționat.

Linia 40 produce salutul execuției înapoi la linia 20, astfel că linia 50 de oprire a programului nu ar fi executată niciodată.

Întrucît programul nu se oprește singur, se spune că "a intrat într-o buclă fără sfîrșit" sau "ciclează".

Întreruperea execuției se poate face după cum deja știți prin:

STOP - la întrebarea dată de INPUT  
BREAK

IF...THEN Instrucțiunea IF...THEN (în traducere: "dacă...atunci") permite introducerea în program a unui transfer (salt) condiționat. Sintaxa: IF x THEN s

Dacă expresia x este adeverată (condiția este îndeplinită) se execută s, în caz contrar se execută instrucțiunea care urmează după IF.

Exemple:

```
IF j=3 THEN i=i+1:GO TO...  
IF a>0 AND a<10 THEN GO TO (3+ax100)  
IF ch=z OR z=66 AND ch<>66 THEN...  
IF NOT a OR NOT w THEN...  
IF 4*(x+5) <(a-b) THEN...
```

Să considerăm în acest ultim exemplu

x=1 a=50 b=25

La execuție calculatorul ajunge să verifice dacă:

$4 \times (1 + 5) < (50 - 25)$

24 < 25

În acest caz condiția este îndeplinită și se va executa ceea ce urmează după THEN. Dacă după calculul expresiilor rezultă o condiție neîndeplinită, s-ar executa instrucțiunea cu numărul de linie imediat superior după IF.

Instrucțiunea de transfer condiționat IF THEN este deosebit de utilă, dind posibilitatea calculatorului să sară în program oriunde dorîți.

După cum poate fi remarcat, programele luate de exemplu pînă acum, au avut o mare deficiență: atunci cînd includeau cicluri de repetare a unui număr de linii, nu există altă cale de a fi opriți decît prin STOP/BREAK/RESET introdus de dvs.

Saltul condiționat (cu IF...THEN) permite o soluție mai elegantă de a stopa un program în anumite condiții.

## CAPITOLUL 6

### EXEMPLE DE PROGRAME

#### Exemplul 6.1

Să se realizeze un program care să afișeze începînd din coloanele 0 și 16, numerele:

3	6
9	12
15	18
21	24

După ce ultimul număr (24) a fost afișat, execuția să se opreasă automat.

Analizind puțin problema dată, observăm că fiecare număr este mai mare cu 3 decît cel anterior.

Programul poate fi realizat în mai multe variante. Cea mai puțin elegantă este:

```
10 PRINT "3 (15 spații) 6"
20 PRINT "9 (15 spații) 12"
30 PRINT "15 (14 spații) 18"
40 PRINT "21 (14 spații) 24"
50 STOP
```

Poate unii zîmbiți cu superioritate, dar rețineți că la un începător este important să facă un program care să dea rezultatele cerute, nu neapărat să fie cel mai bun program! Nu este o soluție prea bună pentru că cere multă atenție la introducere și nu permite ușor o eventuală schimbare a numerelor, sau a spațiilor de afișare.

O altă soluție ar fi:

```
10 LET n=3
20 PRINT n,
30 LET n=n+3
40 IF n >= 25 THEN STOP
50 GO TO 20
RUN
```

Virgula din linia 20 a produs afișarea în coloanele 0 și 16, iar linia 40 a condiționat stoparea programului.

O a treia soluție se obține schimbînd:

```
40 IF n <= 24 THEN GO TO 20
50 STOP
```

Introduceți și execuția fiecare variantă analizînd apoi cu atenție programele și rezultatele.

#### Exemplul 6.2

Să se realizeze un program care să facă media aritmetică a unor numere.

- Analizarea problemei:

În tema dată nu se specifică la cîte numere trebuie făcută media. Dacă este vorba de multe numere, cea mai bună formă de introducere va fi cu READ...DATA.

Pentru a putea stopa programul după ultimul număr (fără a ști care este), să introducem un număr - indicator care să semnaleze calculatorului că este ultimul, și să se opreasă programul.

Să alegem ca indicator valoarea "0" pentru că tot nu ar influența media ce se calculează.

Trebuie deci pusă în program condiția ca atunci cînd este citită valoarea 0 programul să se opreasă.

Fie următoarele variabile:

x - un număr pentru calculul mediei  
s - suma numerelor  
n - numărul de numere pentru care s-a calculat s  
m - media aritmetică

S  
m = ---  
n

#### Codificarea programului:

```
5 REM nume program: MEDAR
10 REM inițializare s și n
20 LET s=0: LET n=0
30 REM citire date
40 READ x
50 REM testare sfîrșit de date
60 IF x=0 THEN GO TO 110
65 PRINT x
70 LET s=s+x
80 LET n=n+1
90 GO TO 40
100 REM calculul mediei
110 LET m=s/n
120 REM afișarea mediei
130 PRINT "
140 PRINT "media este: " ; m
150 DATA 6,3,7,5,4,0
160 STOP
RUN
```

6  
3  
7  
5  
4  
-----  
media este: 5

Un amânunt: la scrierea liniei 60 se lasă necompletat numărul liniei la care să se facă saltul (GO TO ...) pînă cînd se știe care este acea linie.

Se pot pune în program atîtea instrucțiuni DATA cîte sunt necesare pentru a cuprinde toate numerelor, avînd grijă să introducem O după ultimul număr.

### ÎNTREBĂRI RECAPITULATIVE ȘI EXERCITII

I 6.1 Ce se va afișa după execuția următorului program:

```
10 LET a=3
20 LET b=2*a
30 PRINT a, b
40 LET a=a+2
50 IF a <= 10 THEN GO TO 20
60 STOP
```

I 6.2 Indicați greșelile din următoarele instrucțiuni:

- a) IF D~~A~~3 >= 199 THEN 30
- b) IF a2 = 5 THEN k=k+1
- c) GO TO 90 IF t=7
- d) IF q > w : GO TO 140

I 6.3 Scrieți instrucțiunile pentru următoarele condiții:

- a) dacă x < 3 afișează "negru", în caz contrar afișează "alb"
- b) dacă sirul A\$ este "da" salt la 90, dacă nu, salt la 200
- c) dacă t=0 salt la 110, dacă nu, crește x cu 1 și salt la 40
- d) dacă x este cuprins între 1 și 4, salt la linia (1000 + x)

#### Probleme

Scrieți programe BASIC:

- 1) Să ceară două numere și să-l afișeze pe cel mai mare;
- 2) Să citească cinci numere în DATA și să-l afișeze pe cel mai mic;
- 3) Să calculeze și să afișeze suma tuturor numerelor de la 1 la 50;
- 4) Să ceară două numere. Dacă amîndouă sunt mai mari decît sau egale cu 10 să se afișeze suma lor, iar dacă amîndouă sunt mai mici ca 10 să se afișeze produsul lor. Dacă un număr este sub 10 și celălalt este mai mare sau egal cu

## CAPITOLUL 6

10, să se afișeze diferența dintre cel mai mare și cel mai mic.

### Răspunsuri

R 6.1      3            6  
              5            10  
              7            14  
              9            18

- R 6.2 a) lipsă GO TO după THEN  
 b) lipsă LET după THEN  
 c) IF și GO TO inversate și lipsă THEN  
 d) lipsă THEN

- R 6.3 a) 50 IF  $x < 3$  THEN PRINT "negru": GO TO 70  
 60 PRINT "alb"  
 70 .....  
 b) IF A\$ = "da" THEN GO TO 90  
 GO TO 200  
 c) IF t=0 THEN GO 110  
 LET x=x+1 : GO TO 40  
 d) IF  $x \geq 1$  AND  $x \leq 4$  THEN GO TO (1000 + x)

## CUM SE POT REPETA PĂRȚI DIN PROGRAM

Veți învăța în continuare o nouă și foarte folositoare posibilitate a limbajului BASIC, de a realiza cicluri automate în cadrul programelor.

FOR  
:  
NEXT

În capitolul anterior v-am arătat cum se pot face repetiții/cicluri, folosind instrucțiuni de transfer condiționat sau necondiționat. BASIC are însă și instrucțiuni speciale care să înceapă, să execute și să termine automat cicluri în program.

### APLICAȚIE PRACTICĂ AP 7

#### 1. Introduceți:

```
NEW
10 LET a=2
20 PRINT a,: LET a=a+3
30 IF a <= 17 THEN GO TO 20
40 STOP
```

Notați rezultatele execuției:

-----  
-----  
-----

#### 2. Tastați:

```
NEW
10 FOR a=2 TO 17 STEP 3
20 PRINT a,
30 NEXT a
40 STOP
RUN
```

Remarcați că se obțin aceleași rezultate.

#### 3. Schimbați linia 10:

```
10 FOR a=2 TO 20 STEP 2
RUN
```

Analizați rezultatele! Observați că sunt afișate din 2 în 2, adică cifra pusă la cuvântul STEP (în traducere: PAS)

#### 4. Schimbați din nou:

```
10 FOR a=2 TO 10
RUN
```

Nemaifiind specificat STEP, calculatorul dă valori lui a cu un PAS egal cu 1.

#### 5. Tastați:

```
10 FOR a=20 TO 10 STEP -2
RUN
```

Notați rezultatele:

-----  
-----  
-----

#### 6. Încercați acum :

```
10 FOR a = 10 TO 20 STEP -2
RUN
```

Notați mesajul obținut și încercați să-l explicați:

-----  
7. Să vedem ce se întimplă dacă se folosesc două variabile în cicluri FOR...NEXT.

Introduceți:

## CAPITOLUL 7

```

NEW
10 PRINT "a", "b"
20 PRINT
30 FOR a=1 TO 3
40 FOR b=2 TO 5
50 PRINT a, b
60 NEXT b
65 PRINT
70 NEXT a
80 STOP
RUN

```

Notăți rezultatele și să încercăm acum să interpretăm puțin programul:

- linia 10 afișează numele variabilelor
- linia 20 introduce o linie vidă
- linia 30 se citește: "pentru a egal cu 1 la 3" și deschide un ciclu pentru variabila a
- linia 40 se citește: "pentru b egal cu 2 la 5" și deschide un ciclu pentru variabila b
- linia 60 se traduce: "următoarea valoare pentru b" și comandă reluarea ciclului variabilei b, cu linia 40. Pentru a=1, b ia succesiiv valorile 2, 3, 4, 5 (privită la rezultat)
- linia 70 produce salt la linia 30 și trecerea variabilei a la valoarea următoare: a=2 reluindu-se ciclul variabilei b. Același lucru se întimplă și pentru a=3 după care programul este stopat de linia 80.

### 8. Modificații:

```
40 FOR b=1 TO 2
```

Încercați să scrieți ce se va obține și apoi execuția programul și confruntați rezultatele cu cele anticipate.

### NOTIUNI DE BAZĂ

```

FOR
.
.
.
NEXT

```

Instrucțiunile FOR...NEXT permit execuția instrucțiunilor cuprinse între ele de un număr limitat de ori.

Sintaxa: `FOR x=n TO m`  
            sau `FOR x=n TO m STEP s`  
            :  
            `NEXT x`

- x este numele variabilei (o singură literă)
- m, n, s sunt numere sau expresii aritmétice.

Exemple:

- `30 FOR x=2 TO 5` (se traduce: PENTRU x=2,3,4,5)
  - . . . linii de program care se vor putea repeta
- `170 NEXT x` (se traduce: URMĂTOAREA valoarea a lui x)  
`FOR i=a TO a+b STEP d`
  - .
  - .
  - `NEXT i`
- `FOR x = (abc - 2) TO 7 STEP -1`  
`FOR j = 1 TO i`
  - .
  - `NEXT j : NEXT x`
- `FOR i = 0 TO INT (6.9 * RND)`
  - .
  - `NEXT i`

Să vedem puțin mai în detaliu cum lucrează instrucțiunile FOR...NEXT

### Introduceți:

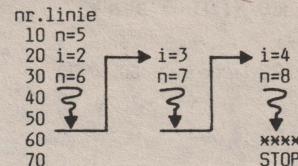
```

NEW
10 LET n=5
20 FOR i = 2 TO 4
30 LET n = n+1
40 PRINT i, n
50 NEXT i
60 PRINT : PRINT "xxxx"
70 STOP
RUN

```

} Se repetă pentru  
i=2 i=3 i=4

Iată rezultatele după execuția fiecărei linii:



La început n primește valoarea 5, apoi calculatorul intră în prima execuție a ciclului, variabila i luând valoarea 2. Se execută liniile 30 și 40 cuprinse în ciclu și linia 50 care produce "saltul" execuției înainte la linia 20 și i devine 3. Bucla se repetă pînă cînd i atinge valoarea limită 4 cu care calculatoruliese din ciclu și se execută instrucțiunea imediat următoare, adică linia 60.

Clauza STEP+s determină creșterea, respectiv scăderea variabilei cu "s" la fiecare execuție a ciclului.

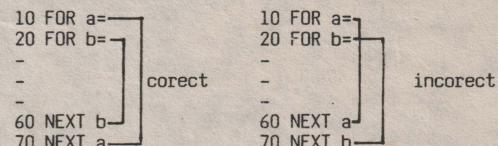
```

FOR x=0 TO 9 STEP 3
x va fi 0,3,6,9
FOR x=16 TO 0 STEP -4
x va fi 16,12,8,4,0

```

În program puteți introduce oricîte cicluri FOR...NEXT, respectînd următoarele condiții:

- un ciclu deschis cu FOR trebuie închis neapărat cu NEXT, deoarece în caz contrar execuția are loc o singură dată, pentru prima valoare a lui x
- buclele să nu se interseceze



Iată alt exemplu de bucle multiple organizate corect:

```

10 FOR x=10 TO 50
20 FOR y=6 TO 8
-
-
-
100 NEXT y
-
-
-
150 FOR z=3 TO 7
-
-
-
200 FOR w=26 TO 12 STEP -2
-
-
-
250 NEXT w
-
-
-
320 NEXT z
-
-
-
360 NEXT x

```

## EXEMPLE DE PROGRAME

## Exemplul 7.1

Program pentru afişarea numerelor cuprinse între 10 şi 50, din 5 în 5.

- a) NEW  
 10 LET a=10  
 20 PRINT a,  
 30 LET a=a+5  
 40 IF a<=50 THEN GO TO 20  
 50 STOP
- b) 10 FOR a=10 TO 50 STEP 5  
 20 PRINT a,  
 30 NEXT a  
 40 STOP

Introduceți programele (aveti grijă! NEW înainte de al doilea) și executați-le verificând identitatea rezultatelor.

## Exemplul 7.2

Se reluăm programul realizat la 6.2, pentru calculul mediei aritmetice a unor numere introduse de la tastatură.

## - Analiza problemei.

Se cere să introducem numere la cerere (interactiv) și programul să calculeze media lor.

Afișarea numerelor să se facă la marginea ecranului, unul sub altul, și la sfîrșit sub o linie de total să se afișeze media.

## - Variabile:

x - un număr pentru calculul mediei

s - suma numerelor

n - numărul de numere pentru care să se calculeze s

m - media aritmetică: m=s/n

## - Codificarea problemei:

```
5 REM nume program: MEDARI
10 INPUT "cîte numere? ";n
20 LET s = 0
30 REM calculul sumei
40 FOR i = 1 TO n
50 INPUT "numărul? ";x
60 LET s = s + x
70 PRINT x
80 NEXT i
90 REM calculul mediei aritmetice
100 LET m=s/n
110 PRINT ""
120 PRINT "media este: ";m
130 STOP
```

Executați programul cu setul de numere din exemplul 6.2:

n=5 numere  
 x=6,3,7,5,4

## Exemplul 7.3

Un program care să vă arate că sinteți atât de bine pregătit, încît puteți decifra conținutul memoriei calculatorului. Despre lucrul intim cu memoria principală veți afla în capitolul 12. Acum anticipăm însă puțin cu cîteva oțuni.

Memoria principală o putem imagina ca un dulap cu multe sertare numerotate unul după altul, începînd cu sertarul avînd numărul 0 și continuînd din 1 în 1 pînă la sfîrșitul memoriei.

Într-un asemenea "sertar" calculatorul memorează un caracter și numărul sertarului reprezentă o ADRESĂ.

Caracterele, vă amintiți, sunt reprezentate prin codurile cuprinse în anexa A.

Dacă introduceți o linie program:

PRINT "x+y=";5

ea este memorată astfel (verificați codurile din anexa A):

adresa	conținut	caracter
23759	245	PRINT
23760	34	"
23761	120	x
23762	43	+
23763	121	y
23764	61	=
23765	34	"
23766	59	;
23767	53	5

Există o instrucțiune care citește conținutul memoriei la orice adresă dorim și îl afișează pe ecran:

PEEK a unde "a" reprezintă adresa

Iată un program care vă afișează adresele de mai sus și conținutul lor:

```
10 PRINT "x+y=";5
100 PRINT "adresa";TAB 7; "conținut"
110 PRINT "-----";TAB 7; "-----"
120 REM ciclu pentru citirea și afișarea adreselor
130 FOR a = 23759 TO 23767
140 PRINT a; TAB 7; PEEK a
150 NEXT a
```

Executați programul începînd cu linia 100:

160 GO TO 100

Analizați rezultatele, așa-i că este interesant!

## ÎNTRERĂI RECAPITULATIVE ȘI EXERCIȚII

Î 7.1 Ce se va afișa după execuția următoarelor programe:

- a) 10 FOR a=20 TO 0 STEP -4  
 20 PRINT a,  
 30 NEXT a  
 40 STOP
- b) 10 FOR a=2 TO 6 STEP 2  
 20 FOR b=2 TO 4  
 30 PRINT a;b;  
 40 PRINT a+b  
 50 NEXT b  
 60 NEXT a  
 70 STOP

Î 7.2 Indicați greșelile din următoarele linii:

- a) FOR a  $\uparrow$  3 = n TO 120 STEP 5  
 b) FOR k\$ = 1 TO 25  
 c) FOR x = y TO n STEP x  
 d) FOR a = 25 TO 70 STEP i  
 NEXT i

Î 7.3 Scrieți buclele FOR...NEXT pentru:

- a) o repetiție de 30 de ori, cu pas 2  
 b) x cuprins între 1 și 10 și o repetiție cu pas 1  
 pînă cînd  $x > 6$  și atunci să se afișeze x și "abc"  
 c) o repetiție cu n cuprins între 0,8 și  $(a^2+3)$  și cu  
 pasul de  $(a+b)$

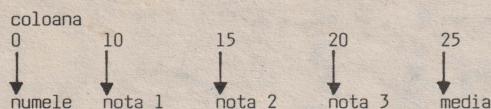
## CAPITOLUL 7

### Probleme

1. Scrieți un program pentru calculul și afișarea următoarelor numere ridicate la cub:

n	$n^3$
2,0	8
2,1	
2,2	
.	
2,9	
3,0	27

2. Scrieți un program care să citească în DATA numele a 5 elevi, notele obținute de fiecare la 3 materii și să se calculeze media. Afișarea să se facă în următorul format:



Indicație: utilizati 5 enunțuri DATA sub forma: DATA "ANDREI"; 7,9,10 numele fiind de maximum 10 caractere.

3. Scrieți un program pentru numerotarea coloanelor și liniilor ecranului, sub forma

```
01234567890123456789012345678901  
1  
2  
3  
. .  
21
```

### Răspunsuri

R 7.1 a) 20 16  
12 8  
4 0  
STOP  
b) 224  
235  
246  
426  
437  
448  
628  
639  
6410

R 7.2 a) variabila nu poate fi o expresie  
b) nu se pot folosi variabile și r  
c) variabila de la FOR nu se poate pune la STEP  
d) variabile diferite la FOR și NEXT

R 7.3 a) FOR n = 2 TO 60 STEP 2  
b) 10 FOR x=1 TO 10  
20 IF x > 6 THEN GO TO 40  
30 NEXT x  
40 PRINT x: PRINT "abc"  
c) FOR n = 0.8 TO (a + 2 + 3) STEP (a+b)

## CAPITOLUL 8

### CUM FOLOSIM COLECȚII DE DATE?

Pe măsură ce dvs. deveniți un programator tot mai experimentat, începeți să aveți pretenția justificată să cunoașteți și posibilitățile mai deosebite ale calculatorului la care lucrăți.

### Tablouri de numere/săruri

Una din aceste posibilități este aceea de a folosi, într-un mod relativ simplu, colecții mari de date. Pentru o mai ușoară înțelegere a noțiunilor, să le comentăm puțin înainte de a trece la aplicația practică.

### DIM. GOSUB. RETURN

Dacă într-un program se utilizează cîteva numere, cinci de exemplu, ele pot fi identificate ușor prin tot atîtea variabile, să zicem a, b, c, d, e. Cum vom proceda însă dacă vrem să lucrăm cu 100 de numere, sau săruri de caractere, diferite?

### Tablou de date

Într-o asemenea situație se lucrează cu un TABLOU DE DATE. Un TABLOU reprezintă o colecție de date identificată printr-un singur nume de variabilă.

Un tablou conține ELEMENTE și poate fi cu o singură dimensiune sau poate avea două dimensiuni.  
Un tablou unidimensional se mai numește LISTA de elemente.

Fie următorul sir de numere: 14, 8, 9, 11, 16, 20, 5, 3

Pentru a putea localiza oricare din aceste numere, le considerăm într-o listă cu opt elemente, căreia îi dăm un nume, să zicem "a". Elementul 1 al listei conține valoarea 14, elementul 2 conține 8 și.m.d.

Număr element	1	2	3	4	5	6	7	8
Conținut	14	8	9	11	16	20	5	3

a (se citește "a indice 1") conține "14",  
1 a conține "11", iar a conține "3".  
4

Într-un tablou bidimensional, fiecare element poate fi localizat prin două coordonate.

	1	2	3	4
1	3	-1	10	9
2	2	4	-2	6

← b  
1,4  
2,4

Dacă vom da tabloului numele "b", valorile cuprinse în el pot fi identificate astfel:

$$\begin{array}{llll} b_1 = 10 & \text{element din rîndul 1, coloana 3} \\ b_{1,3} = 10 & \\ b_{2,3} = -2 & b_{1,4} = 9 & b_{2,1} = - \\ b_{1,2} = - & b_{1,4} = - & b_{2,1} = - \\ & b_{2,4} = - & b_{2,4} = - \end{array}$$

În BASIC indicii se scriu în paranteză:

$$\begin{array}{ll} a_1 & \text{se scrie } a(1) = 14 \\ a_8 & \text{se scrie } a(8) = 3 \\ a_{1,3} & \text{se scrie } a(3) = - \\ a_{1,3} & \text{se scrie } b(1,3) = 10 \\ b_{2,3} & \text{se scrie } b(2,3) = - \\ b_{2,3} & \text{se scrie } b(2,4) = - \\ & \uparrow \uparrow \\ & \text{rîndul coloana} \end{array}$$

## APLICAȚIE PRACTICĂ AP8

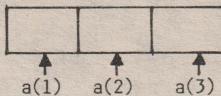
Introduceți:

```

10 DIM a(3)
20 LET a(1) = 6
30 LET a(2) = 33
40 LET a(3) = 999
50 PRINT a(1), a(2), a(3)
RUN

```

Notați rezultatele în: lista "a"



2. Adăugați linia:

```

60 PRINT : PRINT a(1) + a(2)
RUN

```

Ați obținut afișarea conținutului celor trei elemente ale listei și suma primelor două.

3. Tastați:

```

50 FOR i=1 TO 3
55 PRINT a(i),
60 NEXT i
RUN

```

4. Modificați linia 50 pentru a fi afișate numai primele 2 elemente.

```

50 -----
RUN

```

5. Introduceți:

```

50 FOR i=1 TO 4
RUN

```

Notați mesajul obținut.  
Vi s-a semnalat că ați cerut un al 4-lea element din unei liste declarată doar cu 3 elemente prin instrucțiunea:

```
DIM a(3)
```

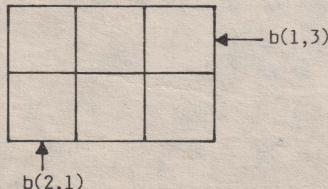
6. Anulați programul existent și introduceți:

```

5 REM tablou cu 6 elemente
10 DIM b(2,3)
20 LET b(1,1) = 3
30 LET b(1,2) = -1
40 LET b(1,3) = 10
50 LET b(2,1) = 2
60 LET b(2,2) = 4.25
70 LET b(2,3) = -2
80 PRINT b(1,1),b(1,2),b(1,3),b(2,1),b(2,2),b(2,3)
RUN

```

Completați rezultatele în tabloul "b"



7. Schimbați programul astfel:

```

80 FOR i=1 TO 2           (rânduri)
90 FOR j=1 TO 3           (coloane)
100 PRINT b(i,j)
110 NEXT j
120 NEXT i
RUN

```

Notați rezultatele:

b(1,1)=	---	b(1,2)=	---	b(1,3)=	---
b(2,1)=	---	b(2,2)=	---	b(2,3)=	---

8. Schimbați programul:

```

NEW
10 DIM a$(12)
20 LET a$ = "ABCDEFGHIJKLM"
25 PRINT "123456789012"
27 PRINT "ABCDEFGHIJKLM"
30 INPUT "număr element="; ne
40 PRINT a$(ne)
50 GO TO 30
RUN

```

Notați ce se obține: număr element = 1	1	----
	= 4	----
	= 9	----
	= 10	----
	= 11	----

9. Anulați liniile 25, 27 și schimbați:

```

30 FOR n=1 TO 10
40 PRINT a$(n);
50 NEXT n
RUN

```

Obțineți conținutul listei de caractere.

10. Introduceți:

```

NEW
10 DIM a$(5)
20 FOR n=1 TO 5
30 READ a$(n)
40 PRINT a$(n)
50 NEXT n
60 DATA "A", "B", "C", "D", "E"
RUN

```

11. Tastați:

```

5 REM tablou cu 5 rânduri și 2 coloane
10 DIM t(5,2)
15 REM r=rind, c=coloana
20 FOR r=1 TO 5
30 FOR c=1 TO 2
40 READ t(r,c)
50 NEXT c: NEXT r
60 REM afișarea tabloului
70 FOR r=1 TO 5
80 FOR c=1 TO 2
90 PRINT t(r,c),
100 NEXT c: NEXT r
110 DATA 1,2
120 DATA 3,4
130 DATA 5,6
140 DATA 7,8
150 DATA 9,10
RUN

```

Comparați ce s-a afișat, cu ce conțin enunțurile DATA.

12. Introduceți alt program:

```

NEW
10 REM program cheltuieli rechizite
15 REM =====
20 REM afișare cap tabel
30 GO SUB 100
35 LET total = 0
40 REM =====
45 REM introducere rechizite
50 GO SUB 150
60 REM =====
65 REM calcul cheltuieli
70 GOSUB 210
80 GO TO 40

```

## CAPITOLUL 8

```
90 REM =====
91 REM calcul total
95 GO SUB 300
99 STOP
100 PRINT "obiect";TAB9;"bucăți";TAB18;"preț";TAB25;"cost"
110 PRINT "*****"
120 PRINT
130 RETURN
150 INPUT "obiect?";o$
152 IF o$="gata" THEN GO TO 90
155 INPUT "bucăți? ";b;"preț unitar? ";p
160 RETURN
210 LET cost = b*p
220 PRINT o$;TAB9;b;TAB18;p;TAB25;cost
230 LET total = total + cost
240 RETURN
300 PRINT "*****"
305 PRINT
310 PRINT "TOTAL: ";TAB25;total
330 RETURN
RUN
```

```
obiect? "caiete"
bucăți? 3      preț? 3.5
obiect? "creioane"
bucăți? 12     preț? 0.9
obiect? "linii"
bucăți? 15     preț? 2
obiect? "gata"
.....rezultate.....
STOP
```

În acest program au apărut două instrucțiuni BASIC pe care nu le-ați întâlnit pînă acum:

GO SUB - prescurtare de la GO SUBROUTINE (în traducere: salt la subrutina)  
RETURN - (în traducere: întoarcere)

Aceste instrucțiuni permit o structurare a programelor astfel încît să se evite scrierea repetată a unuitor părți (denumite subrutine sau subprograme).

Analiați programul de mai sus.

El este structurat cu o parte principală cuprinsă între prima linie (10) și linia STOP (99) și alte părți secundare denumite SUBROUTINE (subprograme).

În cadrul programului principal se fac trimiteri la subprogramele (subroutinele) pentru:

- afișarea capului de tabel
- introducerea rechizitelor cumpărate
- calculul costului acelor rechizite (cheltuieli)
- calculel sumei totale cheltuite

O subrutină începe cu linia dată în GOSUB și se termină cu RETURN care produce întoarcere la programul principal. De exemplu, afișarea capului de tabel este făcută de subrutina conținând liniile (priviți în program):

```
100 PRINT...
.
.
130 RETURN
```

Subrutina este apelată prin linia: 30 GO SUB 100

Urmează execuția liniilor subrutinei pînă la linia 130 care produce întoarcerea la linia care urmează sub GOSUB adică:

35 LET...

Încercați să urmăriți singuri lógica programului și completați informațiile lipsă:

Calculul totalului este determinat de linia: GOSUB  
Subrutina începe cu linia și se termină cu linia după care se execută linia 99.

Introducerea rechizitelor cumpărate, a cantității și a prețurilor corespunzătoare se face ca urmare a liniei :

GO SUB  
Subrutina începe cu linia și se termină cu linia RETURN după care se execută linia REM  
Ați remarcat probabil că pentru a termina execuția introduceți cuvîntul "gata" (adică s-au terminat cumpăraturile), Obiectul? "gata". Se afișează rezultatele. STOP.

### NOTIUNI DE BAZĂ

O colecție de date poate fi memorată în programe BASIC, sub formă de TABLOU conținînd ELEMENTE.  
Pentru a localiza orice element într-un tablou, programatorul trebuie să cunoască numele tabloului și poziția elementului în el. Această poziție este dată prin indexarea (numerotarea) elementelor.

### LISTA

Un tablou în care orice element este localizat printr-un singur indice se numește LISTA.  
Fie de exemplu o listă cu numele "l". Un element al listei este localizat prin:

l(i)  
nume listă ↑↑ poziția/numărul elementului în listă

### MATRICE

Un tablou bidimensional mai este denumit și MATRICE și conține elemente ce pot fi localizate prin doi indici.  
De exemplu, într-o matrice numerică cu numele "t", un element este localizat prin rîndul (r) și coloana (c) în care se află:

t(r,c)  
↑↑  
nume tablou/matrice coloana  
 ↓↓ rînd

Elementele unui tablou pot fi:  
>numere reale= numele tabloului este o singură literă;  
>șir de caractere= numele tabloului este o literă urmată de \$

### DIM

Înainte de a fi utilizat un tablou, în program se pune o instrucție de rezervare a spațiului corespunzător în memorie.  
Instrucția DIM (de la DIMENSION - în engleză) declară tablouri de numere sau de siruri de caractere.

Sintaxa: DIM l(i) sau DIM l\$(i) - pentru LISTA  
DIM t(r,c) sau DIM t\$(r,c) - pentru MATRICE

l,t - nume ale tabloului  
i - numărul elementului în listă  
r,c - numărul rîndului, coloanei la intersecția căror se află elementul

Cu o instrucție DIM se definește numai un singur tablou:

10 DIM a(i), b(j) - greșit  
10 DIM a(i) : DIM b(j) - corect

Instrucția DIM are următoarele defecte:

- 1 - rezerva pentru tablou spațiul corespunzător  
DIM a(7) - spațiu pentru 7 elemente  
DIM t(3,2) - spațiu pentru 6 elemente (3x2)
- 2 - inițializează elementele tabloului cu zero  
Încercați: 10 DIM a(3)  
20 PRINT a(1);a(2);a(3)  
RUN
- 3 - sterge orice alt tablou cu același nume

În program puteți da același nume unui tablou și unei variabile simple, fără să apară eroare deoarece numele tabloului este întotdeauna indexat. Este deci permis să aveți:

LET a=... variabila simplă  
LET a(i)=... tablou

Un tablou de şiruri nu poate avea acelaşi nume cu o variabilă şir simplă.

Un tablou de şiruri are un şir în fiecare rind, toate fiind de aceeaşi lungime (completate sau scurte pînă la aceeaşi lungime declarată):

DIM g\$(3,6)  
3 şiruri      6 caractere în fiecare şir

Tastați: 10 DIM g\$(3,6)  
20 LET g\$(1)="abcdef"  
30 LET g\$(2)="şirul 2"  
40 LET g\$(3)="mdgip"  
50 PRINT g\$(1) g\$(2) g\$(3)  
RUN

Obțineți conținutul matricii: abcdef ← rîndul 1  
şirul 2 ← rîndul 2  
mdgip ← rîndul 3  
                6 caractere

Completați: 60 PRINT g\$(1,4) - se va afișa d  
70 PRINT g\$(2,4) - se va afișa i  
80 PRINT g\$(3,3 TO 5) - se va afișa gi  
              şirul 3      caracterele 3 la 5 (3,4 și 5)

Adăugați: 90 PRINT g\$(1,1) + g\$(2) + g\$(3,1 TO 2)

Ce credeți că va afișa linia 90?

Executați programul și verificați!

În liniile 80 și 90 au apărut două lucruri noi:

- 1 - se pot oferi odată mai multe caractere ale unui şir sub forma: m TO n  
de la acest caracter      pînă la acest caracter (inclusiv)
- 2 - se pot alipi caractere din diferite şiruri prin localizarea și însumarea lor.

## SUBRUTINE

Un program BASIC dacă are anumite părți care se repetă, poate fi structurat astfel încît aceste părți să fie scrise o singură dată și să fie apelate ori de câte ori este nevoie. Astfel de părți de program sunt denumite SUBRUTINE.

### GO SUB

Instrucțiunea GO SUB nr.linie produce saltul în program la subrutina care începe cu linia indicată și se termină cu prima instrucțiune RETURN.

### RETURN

Instrucțiunea RETURN este obligatorie pentru încheierea unei subrute și provoacă întoarcerea execuției programului la prima instrucțiune după GO SUB.

10...	- început program principal
.	
80 GO SUB 200	- salt la linia 200
90...	
.	
130 GO SUB 200	- salt la linia 200
140...	
190 STOP	- sfîrșit program principal
200 REM subrutina	- început subrutină
.	
280 RETURN	- sfîrșit subrutină

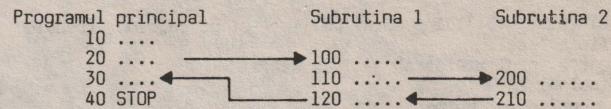
Linia 280 produce întoarcerea la programul principal la liniile 90 și 140.

În general subrutele sunt amplasate spre sfîrșitul programului.

Este permis apelul unei subrute dintr-o altă subrutină. De exemplu:

```
10 PRINT 9
20 GO SUB 100
30 PRINT 9999
40 STOP
100 PRINT 99
110 GO SUB 200
120 RETURN
200 PRINT 999
210 RETURN
RUN
```

```
9 - linia 10
99 - linia 100
999 - linia 200
9999 - linia 30
```



## EXEMPLE DE PROGRAME

### Exemplul 1.

Program pentru declararea unei liste cu 9 elemente care să conțină valorile 1 2 3 4 5 6 7 8 9 și să totalizeze după fiecare element introdus de la tastatură. Rezultatele să fie afișate astfel:

conținut element	suma
1	1
2	3 (2+1)
3	6 (3+3)
.	-
9	-

Variabile: l - numele listei  
n - numărul elementului din listă  
s - suma elementelor

Codificarea programului:

```
10 REM nume program: LISTA
20 DIM l(9)
30 LET s=0
40 PRINT "introduceți în listă valorile:"
50 PRINT "123456789"
60 FOR n = 1 TO 9
70 INPUT "n="; l(n)
80 PRINT n, s+l(n)
90 LET s=s+l(n)
100 NEXT n
110 STOP
```

### Exemplul 2.

Program pentru însumarea valorilor dintr-un tablou bidimensional.

Analizarea problemei:

Fie un tablou cu un număr necarecare de linii și coloane dat la cerere. Programul să cirească datele de calcul într-unul sau mai multe enunțuri DATA, memorîndu-le în tabloul dimensionat corespunzător. Să se facă suma pe linii și coloane, precum și totalul general.

Pentru a înțelege mai bine problema să considerăm, ca un exemplu particular, un tablou cu 4 linii și 5 coloane, care să conțină:

## CĂPITOLUL 8

linii/coloane →	1	2	3	4	5	TOTAL pe linii ↓
↓						
1	1	2	3	4	5	15
2	6	7	8	9	10	40
3	11	12	13	14	15	65
4	16	17	18	19	20	90
TOTAL pe coloane →	34	38	42	46	50	210
						TOTAL GENERAL ↑

Programul propus spre realizare trebuie deci să calculeze:

- suma pe linie; de exemplu pe linia 1:  $1+2+3+4+5=15$
- suma pe coloană, de exemplu pe coloana 1:  $1+6+11+16=34$
- suma totalurilor liniilor:  $15+40+65+90=210$
- suma totalurilor coloanelor:  $34+38+42+46+50=210$

Bineînțeles totalul general pe linii și pe coloane trebuie să fie același.

Afișarea rezultatelor să se facă astfel:

1. tabloul cu date  
conținutul primei linii

.

.

.

.

.

.

.

2. tabloul cu date (vizualizat din nou pentru a verifica ușor totalizarea verticală)

.

.

.

.

.

total general=

.

.

.

coloana 1 total:

.

.

.

coloana c total:

.

.

total general=

.

STOP

Variabile:

nl - număr de linii

nc - număr de coloane

t(nl,nc) - tabloul de calculat

s(nl) - lista unei linii

v(nc) - lista unei coloane

l - număr de linie

c - număr coloană

s - suma totalurilor pe linie (total general pe orizontală)

s(1) - totalul unei linii

v - suma totalurilor pe coloană (total general pe verticală)

v(c) - totalul unei coloane

t(l,c) - tabloul cu date

Codificarea programului:

```

5 REM cerere număr linii și coloane
6 REM
10 INPUT "nl=";nl;"nc=";nc
15 REM declararea dimensiunilor tablourilor
16 REM
20 DIM t(nl,nc):DIM s(nl):DIM v(nc)
30 REM citirea unei linii
35 REM
40 FOR l=1 TO nl
50 FOR c=1 TO nc
60 READ t(l,c)
70 NEXT c
80 NEXT l
90 REM afișarea datelor

```

```

95 REM
100 GO SUB 600
110 REM totalizare orizontală
115 REM
120 PRINT
130 PRINT "totalizare pe linii"
140 PRINT
150 LET s=0
160 FOR l=1 TO nl
170 LET s(l)=0
180 FOR c=1 TO nc
190 LET s(l)=s(l)+t(l,c)
200 NEXT c
210 PRINT "linia " ;l;"total: " ;s(l)
220 LET s=s+s(l)
230 NEXT l
235 PRINT
240 PRINT "total general= " ;s
250 PRINT
260 REM afișarea datelor
265 REM
270 GO SUB 600
280 REM totalizare verticală
285 REM
290 PRINT
300 PRINT "totalizare verticală"
310 PRINT
320 LET v=0
330 FOR c=1 TO nc
340 LET v(c)=0
350 FOR l=1 TO nl
360 LET v(c)=v(c)+t(l,c)
370 NEXT l
380 PRINT "coloana " ;c;"total: " ;v(c)
390 LET v=v+v(c)
400 NEXT c
405 PRINT
410 PRINT "total general= " ;v
500 DATA 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,
19,20
550 STOP
600 REM rutina pentru afișarea datelor
605 REM
610 PRINT "tabloul cu date"
620 PRINT
630 FOR l=1 TO nl
640 FOR c=1 TO nc
650 PRINT t(l,c)
660 NEXT c
670 PRINT
680 NEXT l
690 RETURN

```

Analizați programul și sub aspectul scrierii lui (cu linii decalate) astfel încât să fie cit mai ușor de urmărit.  
Atenție! Dacă vi se afișează mesajul: E Out of DATA, 60:1 nu ati introdus în linia 500 date într-un număr egal cu: număr linii × număr coloane.

### ÎNTREBĂRI RECAPITULATIVE ȘI EXERCȚII

Î 8.1 Ce tipuri de tablouri (listă/matrice) sunt următoarele și cîte elemente conțin?

- DIM A\$(20)
- DIM q(5,10)
- DIM w\$(10,16)
- DIM F(8)

Î 8.2 Ce erori conțin următoarele enunțuri?

- DIM a(6,8,14)
- DIM b(k,-3)
- DIM x(7), y(15)
- DIM l

Î 8.3 Scrieți instrucțiunile necesare pentru executarea următoarelor operații (folosind cicluri FOR NEXT):

- să încarce cu "1" o matrice cu 10 linii și 10 coloane și să se afișeze în format de 10x10

b) să genereze, să încarce și să se tipărească o listă cu următorul conținut:

2	4
8	16
32	64

R 8.4 Fie o listă a(8). Să se afișeze indicele (i) și valoarea (v) pentru toate elementele mai mici decât 15, cuprinse în:

DATA 1,12,17,4,28,20,2,13

Afișarea să se facă începând din coloanele 3 și 12.

coloana 3                    coloana 12  


R 8.5 Ce va afișa următorul program?

```

10 LET n=10
20 GO SUB 100
30 LET n=n/2
40 GO SUB 100
50 LET n=n+i:GO SUB 100
60 STOP
100 LET s=0
110 FOR i=1 TO n:LET s=s+i
120 NEXT i
130 PRINT s: RETURN
    
```

#### Probleme

- Pentru un tablou numeric "a" cu x linii și x coloane, scrieți un program care să calculeze produsul:  $a(1,1)*a(2,2)*a(3,3)*...*a(x,x)$
- Program pentru afișarea elementelor coloanei a șasea a unui tablou T(8,10).
- Program pentru afișarea liniei a treia a unui tablou d(5,8).
- În instrucțiuni DATA puneți notele obținute de 15 elevi la 5 materii.  
Realizați programul pentru calculul mediilor.

Tabloul datelor:

elevi materii	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	10	9	9	7	6	10	5	6	10	10	8	7	9	8	10
2	9	7	8	6	6	8	6	7	9	10	9	7	9	8	9
3	9	5	7	7	5	8	7	8	9	10	10	8	10	8	10
4	8	6	6	8	7	10	6	6	10	9	8	8	10	9	9
5	10	8	10	7	5	9	5	7	9	8	9	8	9	9	10

Rezultatele să fie afișate astfel:

elev	media
1	?
.	.
.	?
15	?

#### Răspunsuri

- R 8.1 a) lista alfanumerică - 20 elemente  
 b) matrice numerică - 50 elemente  
 c) matrice alfanumerică - 160 elemente  
 d) listă numerică - 8 elemente

R 8.2 a) indice zecimal nu este permis  
 b) indice negativ nu este permis  
 c) cu o instrucțiune DIM se definește un singur tablou; corect: DIM x(7):DIM y(15)  
 d) nume de listă fără indice

R 8.3 a) 10 DIM n(10,10)  
 20 FOR l=1 TO 10  
 30 FOR c=1 TO 10  
 40 LET n(l,c)=1  
 50 NEXT c  
 60 NEXT l  
 70 FOR l=1 TO 10  
 80 FOR c=1 TO 10  
 90 PRINT n(l,c);  
 100 NEXT c  
 110 PRINT  
 120 NEXT l

b) 10 DIM a(6)  
 20 LET a(1)=2:PRINT a(1),  
 30 FOR l=2 TO 6  
 40 LET a(l)=2\*a(l-1)  
 50 PRINT a(1),  
 60 NEXT l

R 8.4 10 DIM a(8)  
 20 FOR i=1 TO 8  
 30 READ a(i)  
 40 IF a(i) > 15 THEN GO TO 60  
 50 PRINT TAB 3; "i=";i;TAB 12;"v=";a(i)  
 60 NEXT i  
 70 DATA 1,12,17,4,28,20,2,13

R 8.5 55  
 15  
 66

## CAPITOLUL 9

### SĂ DESENĂM CU CIP-UL

În acest capitol veți cunoaște alte cîteva posibilități ale calculatorului cu care lucrați:

- > moduri de afișare care să pună în evidență numai anumite informații de pe ecran;
- > realizarea de desene pe ecran
- > obținerea desenului de animație

#### Instrucțiuni de utilizare a culorilor. Instrucțiuni grafice. Numere aleatoare. Caractere definite de utilizator

Cip-ul permite colorarea imaginilor, atât alfanumerice cât și grafice astfel încît dacă puteți utiliza un televizor/monitor color, lucrul cu calculatorul devine și mai fascinant.

Puteți folosi opt culori prin acționarea în anumite instrucțiuni a cifrelor de la 0 la 7 astfel:

0 - negru	2 - roșu	4 - verde	6 - galben
1 - albastru	3 - mov	5 - albastru deschis	7 - alb

Pe ecranul unui televizor/monitor alb-negru aceste culori apar ca nuanțe de la negru la gri deschis.

Cele opt culori pot fi utilizate în următoarele instrucțiuni:

BORD'R 0...7 - colorarea conturului (bordurii) ecranului  
PAPER 0...7 - colorarea fondului (hîrtiei)  
INK 0...7 - colorarea caracterului/punctului afișat

Stabiliti legătura cu televizorul color și execuția aplicația practică, observind cu atenție ce se întimplă.

### APLICAȚIE PRACTICĂ AP9

1. Introduceți următoarele instrucțiuni (nenumerotate) și notați ce culori se obțin pe contur.

BORDER 1	-----
BORDER 2	-----
BORDER 3	-----
BORDER 4	-----
BORDER 5	-----
BORDER 6	-----
BORDER 7	-----
BORDER 0	-----

2. Tastați:

```
10 FOR b=0 TO 7 : BORDER b: PAUSE 50: NEXT b
RUN
```

3. Introduceți următoarea instrucțiune și veți obține afișarea în mijlocul ecranului a unui mesaj clipitor, cu litere albe pe fond albastru:

```
NEW
PRINT AT 11,10;PAPER 1;INK 7;FLASH 1;"OPRIȚI BANDA!"
```

S-a afișat în linia        începînd cu coloana       . Afișarea clipitoare se obține cu instrucțiunea FLASH 1, iar efectul intermitent se anulează cu FLASH 0.

4. Stergeți ecranul cu CLS și introduceți instrucțiunile necesare pentru a se afișa clipitor, în ultimele două linii ale ecranului un mesaj în roșu și verde.

```
CLS
PRINT AT 20,1;FLASH 1;INK 2;"Pentru continuare";
INK 4;"   apăsați orice tastă!";INK 0
```

"Cernea" (INK) roșie și verde a fost dată de instrucțiunile        și       .

5. Tastați următoarea linie care va afișa în linia 3, începînd din coloana 3, pe fond (hîrtie) galben și verde:

CLS

PRINT AT 3,3;"Ecranul are";PAPER 6;"   22 lini";  
PAPER 7;"   și "  ";PAPER 4;"32 coloane"

6. Următoarea linie va afișa cu litere albe pe fond negru:

```
CLS
PRINT INVERSE 1; AT 4,10;"ALB"
RUN
```

Comanda INVERSE 1 înversează culorile pentru PAPER și INK. Reveniți la afișarea curentă (fără CLS):

```
PRINT INVERSE 0;AT 4,10; "NEGRU"
RUN
```

Am lăsat intenționat litera "B" afișată invers pentru a sesiza diferența.

7. Introduceți:

```
CLS
PRINT AT 4,10 ;FLASH 1;"clipitor:";FLASH 0;"/NORMAL"
```

8. În exemplul următor realizați și afișarea color a unor linii, dar rețineți și semnificația textului care are coordonatele extreme ale ecranului în afișarea grafică;

```
CLS
PRINT PAPER 6;AT 3,3;"punctele extreme sunt:";PAPER
4;"   ;"(0,0)    - colțul stînga jos"; "   ;
"(0,175)    - colțul stînga sus"; "   ;"(255,0)
   - colțul dreapta jos"; "   ;"(255,175)   
- colțul dreapta sus"
```

Din această linie lungă de program rețineți două aspecte:  
- culoarea verde (PAPER 4) a fost scrisă o singură dată, în continuare ea rămîne stabilită și în liniile care au urmat;  
- un blank între apostrofuri ('    ') introduce o linie vidă.

9. Introduceți următorul program, urmărind cu atenție comentariile:

```
CLS
5 REM marcarea punctelor extreme
10 PLOT 0,0           - colțul stînga jos
20 PLOT 255,0         - colțul dreapta jos (x max)
30 PLOT 0,175          - colțul stînga sus (y max)
40 PLOT 255,175        - colțul dreapta sus
RUN
```

Priviți cu atenție ecranul și veți observa cele patru puncte.

10. Adăugați liniile:

```
45 REM trasarea axei x
50 FOR x=0 TO 255: PLOT x,0 : NEXT x
55 REM trasarea axei y
60 FOR y=0 TO 175 : PLOT 0,y : NEXT y
RUN
```

11. Continuați introducerea:

```
65 REM punct de coordonate x=150, y=100
70 PLOT 150, 100
75 REM trasarea abscisei punctului
80 PLOT 0,100
90 FOR x=0 TO 150 : PLOT x,100 : NEXT x
95 REM trasarea ordonatei punctului
100 FOR y=0 TO 100 : PLOT 150,y : NEXT y
RUN
```

12. Anulați liniile 65,70,75,80,90,95,100 și modificați liniile:

```
50 PLOT 0,0 : DRAW 255,0
RUN
```

Se trasează axa x dintr-o dată și mai rapid decât prin puncte.

```
60 PLOT 0,0 : DRAW 0,175
RUN
```

Se trasează rapid axa y.

```
65 REM desenarea diagonalei ecranului
70 PLOT 255,175
80 DRAW -255, -175
```

13. Desenați un pătrat cu latura de 30.

```
NEW
5 BORDER 2
10 PLOT 50,50 - punctul de plecare
20 DRAW 30,0
25 PAUSE 50
30 DRAW 0,30
35 PAUSE 50
40 DRAW -30,0
45 PAUSE 50
50 DRAW 0, -30
RUN <ENT> <ENT> (tastați de două ori)
```

Instrucțiunea PAUSE am introdus-o pentru a observa treptat trasarea laturilor pătratelor.  
Anulați linile 25, 35 și 45 și executați din nou programul. Completați linia 5

```
5 BORDER 1: PAPER 5 : INK 7
RUN <ENT>
```

Ce s-a schimbat în privința colorilor?

14. Descrieți un cerc cu raza de 30 și cu centrul în punctul de coordonate x=170 și y=120.

```
55 PLOT 170,120
60 CIRCLE 170,120,30
RUN
```

15. Desenați acum un cerc prin două arce de cerc (semicercuri):

```
70 PLOT 40,120
80 DRAW 20,20,PI:PAUSE 50
90 DRAW -20,-20,PI
RUN
```

16. Să recapitulăm (vezi figura 5.1)  
-punctul de origine la afișarea alfanumerică pentru instrucțiunea: PRINT AT linia, coloana este în colțul \_\_\_\_\_ = \_\_\_\_\_

-punctul de origine la afișarea grafică, pentru instrucțiunile : PLOT x, y  
DRAW x, y  
CIRCLE x, y, r  
este în colțul \_\_\_\_\_ = \_\_\_\_\_.

17. Introduceți un nou program:

```
NEW
5 BORDER 6
10 FOR n=1 TO 10
20 LET x = 100 * RND
30 LET y = 170 * RND
40 PLOT 120, 0
50 DRAW x, y
60 PLOT 120, 0
70 DRAW -x, y
80 NEXT n
RUN
```

Iată o nouă surpriză:

CIP-ul poate obține efecte neașteptate (deosebit de utile la jocuri), generând numere la întâmplare, între anumite limite date prin program. Aceste numere li se mai spune numere ALEATOARE (în engleză RANDOM = aleator, la întâmplare).

Linia 20 va produce la întâmplare numere cuprinse între 0 și 100 pentru x, iar linia 30 va produce numere aleatoare între 0 și 170 pentru y.

18. Schimbați în program:

```
5 BORDER 4: FOR i=6 TO 2 STEP -1
50 DRAW INK i ; x, y
70 DRAW INK i; -x, y
90 NEXT i
RUN
```

19. Încă o schimbare în program:

```
50 DRAW PAPER i-1; INK i; x,y
70 DRAW PAPER i-1; INK i;-x,y
RUN
```

Observați că instrucțiunea DRAW permite utilizarea opțiunilor de culoare pentru PAPER și INK, în mod asemănător cu PRINT. Execuția de cîteva ori programul cu RUN.

20. Să analizăm acum puțin numerele aleatoare generate prin instrucțiunea RND, reluind primele linii din programul anterior:

```
NEW
10 FOR n=1 TO 4
20 LET x=100*RND
30 LET y=170*RND
40 PRINT x,y
50 NEXT n
RUN
```

Notați valorile afișate: -----  
-----  
-----  
-----

Execuția programul din nou RUN --

Comparați cele afișate acum cu cele inscrise mai sus și observați valorile diferite pentru x și y.  
Dacă dorîți numere întregi aleatoare, schimbați, de exemplu linia 30:

```
30 LET x=INT(170*RND)
RUN
```

Notați: -----  
-----  
-----  
-----

De data aceasta y ia valori întregi.

21. Să acum un program pentru trasarea graficului funcției trigonometrice SIN între 0 și 2 PI cu două culori și două valori de amplitudine:

```
NEW
10 BORDER 6
20 PLOT 0,88:DRAW 255,0
30 FOR y=0 TO 255
40 PLOT INK 4;y,88+40*SIN(y/128*PI)
50 PLOT INK 1;y,88+85*SIN(y/128*PI)
60 NEXT y
RUN
```

22. Introduceți: NEW

```
1 REM folosirea caracterelor grafice standard █ și █
5 BORDER 6
10 PRINT AT 18,6;INK 2; "█████████████████████"
20 LET l=17
30 LET y$="█"
40 FOR c=6 TO 20
50 PRINT AT 1,c;y$
60 LET l=l-1
70 NEXT c
80 FOR l=3 TO 18
90 PRINT AT 1,l;INK 2;" █ "
100 NEXT l
RUN
```

## CAPITOLUL 9

Obțineți o scară solidă. Să-i dăm o utilizare făcând să sară pe ea o... mingă reprezentată prin literă "O". Veți învăța astfel cum se realizează animația pe ecran. Adăugați liniile:

```

110 FOR k=1 TO 5
120 LET lm=1
130 FOR c=21 TO 4 STEP -1
135 REM se desenează mingea
140 PRINT AT lm,c; INK 1;"O"
150 PAUSE 8
155 REM se sterge mingea
160 PRINT AT lm,c; " "
170 LET lm=lm+1
180 NEXT c
190 REM mingea pe orizontală
200 FOR c=5 TO 22
210 PRINT AT 19,c; INK 1;"O"
220 PAUSE 4
230 PRINT AT 19,c; " "
240 NEXT c
250 REM mingea pe verticală
260 FOR l=18 TO 1 STEP -1
270 PRINT AT l,22; INK 1;"O"
280 PAUSE 2
290 PRINT AT l,22; " "
300 NEXT l
310 NEXT k
RUN

```

Animăția se obține într-un ciclu FOR...NEXT, prin afișarea într-o poziție, menținerea imaginii pe o anumită durată și stergerea imaginii, urmată de afișarea în poziția următoare. De exemplu, ridicarea verticală a mingiei este realizată cu liniile:

260 FOR l=18 TO 1 STEP -1	-scade numărul liniei
270 PRINT AT l,22; "O"	-afișază "O"
280 PAUSE 2	-menține imaginea
290 PRINT AT l,22; " "	-sterge pe "O"
300 NEXT l	-poziția următoare

### NOTIUNI DE BAZĂ

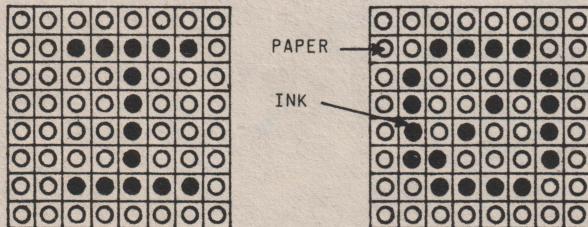
Dacă aveți la dispoziție un televizor color, cu CIP-ul puteți realiza opt culori, astfel:

1. albastru	2. roșu	3. magenta	4. verde
5. bleu (cyan)	6. galben	7. alb	0. negru

Un caracter este afișat într-un pătrat format din 8x8 puncte. De exemplu litera "I" și cifra "0" sunt afișate astfel:

### INK. PAPER

Pentru fiecare caracter se pot folosi două culori:  
 INK c - culoarea punctelor negre din desen  
 PAPER c - culoarea punctelor albe din desen, c putind avea valoarea de la 0 la 7.



### BORDER

Instrucțiunea BORDER c colorează marginea imaginii cu culoarea dată de valoarea lui c.

BORDER 2 - margine roșie

BORDER 6 - margine galbenă

Culorile și afișarea clipitoare (FLASH) sunt considerate ca ATRIBUTE ce pot fi asociate unui caracter.

Să afișăm litera I cu cerneală (INK) albastră pe fond (PAPER) galben:

```

NEW
10 FOR n=1 TO 5
20 PRINT PAPER 6; INK 1; "I";
30 PRINT PAPER 7; INK 7; " ";
40 NEXT n

```

Atributele de culoare pot fi utilizate fie separate, fie în instrucțiunea PRINT.  
 Următorul program va afișa cele opt culori și cifrele corespunzătoare:

```

NEW
10 FOR n=1 TO 80
20 FOR c=0 TO 7
30 PAPER c
40 PRINT c;
50 NEXT c: NEXT n
RUN

```

Dacă dorim ca cifrele să fie mai evidente, se poate obține un contrast mai bun introducind: INK 9 sau PAPER 9.  
 Cifra 9 nu este asociată unei culori, ci are ca efect cerneala mai deschisă pe fond închis (negru, roșu, magenta) și cerneala mai închisă pe fond mai deschis (verde, bleu, galben, alb).

Verificați introducind linia:

```

15 INK 9
RUN

```

Atributele de culoare și clipire le puteți folosi cu mare efect, atunci cind doriți să scoateți în evidență anumite informații de pe ecran.

Dacă, de exemplu aveți ecranul plin și afișarea este uniformă, cel care analizează cele afișate trebuie să parcurgă tot conținutul pentru a găsi o informație. Afișind însă în altă culoare și eventual clipitor, imediat informația respectivă atrage atenția privitorului.

Posiționarea pe ecran se face cu: PRINT AT linie, coloana unde: linie = 0 - 21  
 coloana = 0 - 31

Atributele pot fi plasate în linia de program fie imediat după PRINT, fie după AT linie, coloana:

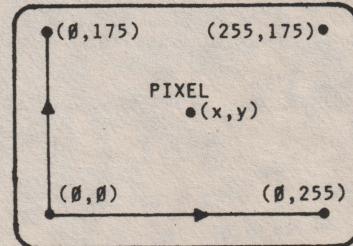
```

PRINT AT 11,10;PAPER 1;INK 7;FLASH 1;"OPRIȚI BANDA"
PRINT PAPER 1;INK 7;FLASH 1;AT 11,10;"OPRIȚI BANDA"

```

Realizarea desenelor pe ecran se obține prin puncte.

Un punct este denumit PIXEL și poate fi pozitionat prin program, cu două coordinate față de colțul din stînga - jos al ecranului (vezi figura 5.1)



### PLOT

Instrucțiunea PLOT x, y stabilește poziția și desenează un punct la coordonatele x, y:

```

x = 0 - 255
y = 0 - 175

```

Desenele pot fi afișate punct cu punct. De exemplu trasarea a două axe la extremitățile ecranului se poate face astfel:

```

axa x: FOR x=0 TO 255 : PLOT x,0: NEXT x
axa y: FOR y=0 TO 175 : PLOT 0,y: NEXT y

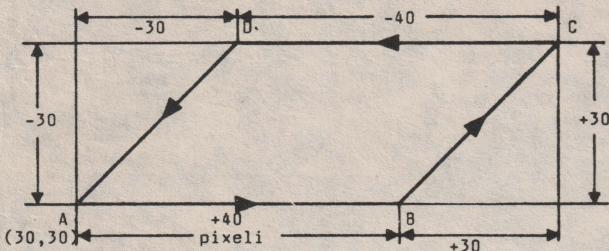
```

DRAW

Instrucțiunea DRAW  $x, y$  desenează o linie de la poziția curentă a cursorului grafic pînă la un punct de coordonate  $x$  și  $y$  pixeli. De exemplu trasarea celor două axe, luînd ca punct de origine (10,10) se poate obține cu:

```
NEW
10 PLOT 10, 10
20 DRAW 245, 0
30 PLOT 10, 10
40 DRAW 0, 165
```

Pentru a înțelege mai bine folosirea instrucțiunii DRAW să analizăm desenarea paralelogramului din figură:



Ne propunem să începem desenul din colțul A de coordonate  $x = 30$ ;  $y = 30$  și să trasăm laturile în sensul arătat.

```
NEW
10 PLOT 30, 30
20 DRAW 40, 0      - latura AB
30 DRAW 30, 30     - latura BC
40 DRAW -40, 0     - latura CD
50 DRAW -30, -30   - latura DA
```

Folosind un asemenea paralelogram în cicluri FOR...NEXT puteți obține construcții spectaculoase ca în programul următor:

```
NEW
10 BORDER 6
20 FOR n=1 TO 100 STEP 1.2
30 PLOT 30, 30 + n
40 DRAW 40, n-n
50 DRAW 30, (n-n)+30
55 BEEP 0.1, n/10
60 NEXT n
70 DRAW -40, n-n
80 DRAW -30, (n-n)-30
90 FOR n = 1 TO 105 STEP 2
100 PLOT 150, 50 + n
110 INK 2
120 DRAW 10, n-n
130 15, (n-n)+15
135 BEEP 0.01, n/5
140 NEXT n
150 DRAW -10, n-n
160 DRAW -15, (n-n)-15
170 STOP
RUN
```

O altă formă a instrucțiunii DRAW este: DRAW  $x, y, z, -z$  este unghiul (în radiani) pentru desenarea unei linii curbe între poziția cursorului grafic și poziția dată de  $x, y$ :

```
NEW
10 FOR i=1 TO 7: READ z
20 PLOT 125, 0
30 DRAW 0, 100, z
35 PAUSE 40: NEXT i
40 DATA PI/2, PI/4, 2*PI
50 DATA -PI/2, -PI/3, PI, -PI
```

Observați că între cele două extremități date de (125, 0) și (0, 100) se obțin curbe diferite în funcție de valoarea lui  $z$  dată în radiani.

CIRCLE

Instrucțiunea CIRCLE  $x, y, r$  desenează un cerc de rază  $r$  cu centru în punctul de coordonate  $x$  și  $y$

```
NEW
10 FOR i = 1 TO 4
20 READ raza
30 CIRCLE 100, 85, raza
40 DATA 20, 30, 50, 80
50 NEXT i
60 PLOT 100, 85
RUN
```

Sau:

```
NEW
10 BORDER 5
20 FOR r = 1 TO 50, STEP 2
30 CIRCLE INK 1; 127, 85, r
40 NEXT r
RUN
```

Generarea unor numere la întâmplare

Adesea în program (în special în jocuri) se folosesc numere diferite generate "la întâmplare". Astfel de numere ALEATOARE (în engleză : RANDOM NUMBERS). Imagineați-vă diferențele combinației de numere date de aruncarea a două zaruri și veți realiza ce se înțelege prin numere aleatoare. Există diverse metode matematice pentru a produce numere aleatoare. CIP-ul generează astfel de numere cu funcția RND.

RND

RND - generează numere zecimale între 0 și 0,99999999

```
NEW
10 FOR i=1 TO 20
20 LET n=RND
30 PRINT n,
40 NEXT i
```

►  $a * RND$  - generează numere zecimale între 0 și  $a$ .  
Schimbați linia 20:

```
20 LET n=15*RND
RUN
```

Execuțați programul de cîteva ori și veți obține numere între 0 și 15.

►  $b + a * RND$  - generează numere zecimale între  $b$  și  $(a+b)$ .  
Schimbați:

```
20 LET n = 2 + 15 * RND
```

Se obțin numere zecimale cuprinse între 2 și 17.

```
20 LET n = 4.4 + 0.6 * RND
```

Se obțin numere între 4,4 și 5

► INT ( $a * RND$ ) - generează numere întregi între 0 și  $a$

```
20 LET n = INT(6*RND)
```

►  $b + INT(a * RND)$  - generează numere întregi între  $b$  și  $(a+b)$

```
20 LET n = 1 + INT(6*RND)
RUN
```

Execuțați de cîteva ori programul și veți obține numere aleatoare cuprinse între 1 și 6, deoarece funcția INT intregește numărul la valoarea întreagă inferioară.

```
INT (3,65) = 3
INT (5,999) = 5
```

## CAPITOLUL 9

### RAND (RANDOMIZE)

Uneori este însă nevoie de numere aleatoare în seturi care să se repete. Această posibilitate o dă enunțul RANDOMIZE (prescurtat: RAND). Pe tastatură obțineți (vezi anexa B):

"RAND" cu tasta T în modul K  
"RND" cu tasta T în modul E

Pentru a vedea diferența între funcția RND și enunțul RAND, introduceți următorul program:

```
NEW
50 FOR r = 1 TO 3
100 FOR i = 1 TO 5
110 LET a = 1 + INT(10 * RND)
120 PRINT a
130 NEXT i: PRINT: .NEXT r
```

La fiecare execuție a programului veți obține trei seturi diferite de numere întregi aleatoare între 1 și 10.  
Completați programul cu liniile:

```
10 FOR x = 1 TO 5
20 INPUT "n= "; n
30 CLS
70 RANDOMIZE n
140 NEXT n
```

Enunțul RAND are sintaxa: RAND n unde n = 0 la 65535.  
Executeți programul dând lui x valorile următoare și notați rezultatele pentru a sesiza diferențele și asemănările între ele:  
n = 1 <ENT>  
n = 70 <ENT>  
n = 999 <ENT>  
n = 12345 <ENT>  
n = 65535 <ENT>

Analizând rezultatele notate, observați că pentru o valoare dată lui n se obțin aceleși seturi de numere, aleatoare, dar diferă numerele din seturi, pentru diferite valori ale lui n.

RAND și RND îl utilizează timpul trecut de la punerea în funcțiune a calculatorului, care dacă nu crește mult între două execuții ale lui RANDOMIZE, determină generarea unor seturi cu numere aproximativ aceleșa.  
Încercați execuția programului repetind n=0!

### CARACTERE GRAFICE STANDARD

Folosind caracterul grafic "█", următorul program vă va da un ecran clipitor:

```
NEW
10 BORDER 3
20 LET n = 1 + INT(7*RND): INK n
30 LET l = 1 + INT(21*RND)
40 LET c = 1 + INT(31*RND)
50 PRINT AT l, c; "█"
60 GO TO 10
```

Întrerupeți cînd dorîți cu BREAK.

Caracterele grafice standard se obțin în modul G (CS+9), cu tastele numerice 1 la 8 (vezi anexa B). Ele sunt utile cînd dorîți să afișați texte sau grafice de dimensiuni mai mari decît cele obișnuite.

Introduceți:

```
10 PRINT AT 8, 10 " ████ ████ ████ "
20 PRINT AT 9, 10 " ███ ███ ███ █ "
30 PRINT AT 10, 10 " ████ ████ ████ █ "
40 PRINT AT 11, 10 " ███ ███ ███ █ "
50 PRINT AT 12, 10 " ███ ███ ███ █ "
60 PRINT AT 13, 10 " ███ ███ ███ █ "
70 PRINT AT 14, 10 " ███ ███ ███ █ "
90 REM: trasare coloane
100 PLOT 0,0: DRAW 0, 175
110 FOR x = 7 TO 255 STEP 8
120 PLOT x,0 : DRAW 0,175
130 BEEP .001,20
```

```
140 NEXT x
190 REM:trasare linii
200 PLOT 0,0: DRAW 255,0
210 FOR y = 7 TO 175 STEP 8
220 PLOT 0, y : DRAW 255,0
230 BEEP .001,20
240 NEXT y
```

### Cum se pot realiza caractere grafice speciale?

Așa cum am arătat, un caracter este reprezentat (afisat) într-un patrat de 8 × 8 puncte. Folosind acest mod de reprezentare utilizatorul poate să-și definească simboluri grafice introducind 0 pentru PAPER și 1 pentru INK.

Dacă priviți în Anexa 1, codurile 144 la 164 sunt rezervate pentru caractere grafice definite de utilizator (în engleză UDG-USER DEFINED GRAPHICS).

Fiecare UDG poate fi asociat prin program unei taste alfabetice, astfel încît să poată fi obținut oricînd se trece în modul grafic și se activează tasta respectivă. Pentru exemplificare să încercăm să definim litera grecească "λ" pe tasta <A>:

1. punctăm noul caracter în 8 × 8 puncte lăsînd însă la margini cîte un sir nefolosit pentru a separa caracterul de altele alăturate:

1	2	3	4	5	6	7	8
1	○	○	○	○	○	○	○
2	○	○	○	○	●	○	○
3	○	○	○	●	○	○	○
4	○	○	○	○	○	○	○
5	○	○	●	●	○	○	○
6	○	●	○	○	○	○	○
7	○	○	●	●	●	○	○
8	○	○	○	○	○	○	○

2. se memorează fiecare din cele 8 rînduri cu enunțul BIN (provenit din BINAR) urmat de 8 cifre binare:

0 - pentru fond hîrtie (PAPER)  
1 - pentru cerneală (INK)

Cele opt numere binare rezultate sunt inscrise în memoria internă a CIP-ului în opt poziții, fiecare poziție avînd o ADRESĂ. Adresa primului rînd este URS"A" (URS de la USER și "A" de la tasta pe care am stabilit-o pentru noul caracter).

Pătratul 8 × 8 puncte devine:

OCTETI	adresele de memorare
BIN 0 0 0 0 0 0 0 0	USR "A"
BIN 0 0 0 0 1 1 0 0	USR "A" + 1
BIN 0 0 0 1 0 0 1 0	USR "A" + 2
BIN 0 0 0 1 0 0 0 0	USR "A" + 3
BIN 0 0 1 1 0 0 0 0	USR "A" + 4
BIN 0 1 0 1 0 0 1 0	USR "A" + 5
BIN 0 0 1 0 1 1 0 0	USR "A" + 6
BIN 0 0 0 0 0 0 0 0	USR "A" + 7

Numărul binar dat de un rînd, fiind format din 8 biți (cifre binare) poartă numele de octet (în engleză BYTE).

Recapitulind puțin:

adresa	octetul memorat
USR "A" + 2	0 0 0 1 0 0 1 0
USR "A" + 5	0 1 0 1 0 0 1 0
USR "A" + 7	0 0 0 0 0 0 0 0

Memorarea directă a unui număr la o anumită adresă se poate face cu: POKE adresă, număr  
Următorul program va memora cei 8 octeti reprezentînd caracterul "λ".

```

NEW
10 FOR r = TO 7
20 READ b
30 POKE USR "a" + r, b
40 NEXT r
110 DATA BIN 0 0 0 0 0 0 0 0
120 DATA BIN 0 0 0 0 1 1 0 0
130 DATA BIN 0 0 1 1 0 0 1 0
140 DATA BIN 0 0 1 0 0 0 0 0
150 DATA BIN 0 0 1 1 0 0 0 0
160 DATA BIN 0 1 0 1 0 0 1 0
170 DATA BIN 0 0 1 0 1 1 0 0
180 DATA BIN 0 0 0 0 0 0 0 0
RUN

```

După execuția programului vi se afișează mesajul: OK, 180:1

### 3. Introduceți linia:

```
200 PRINT "aaaa      (fără <ENT>)
```

4. Întrați în mod grafic și apăsați din nou "A". Acum se afișează "L". Completați linia 200:

```
200 PRINT "aaaa LLLL"
```

Atenție! Pentru a introduce ghilimelele de la sfîrșitul săirului, reveniți din modul G, în modul L reapăsind (CS + 9)

```
RUN 200
```

Se afișează aaaa LLLL

De cîte ori stabiliți modul G, prin apăsarea tastei <A> veți obține "L".

### Animatia (mișcarea) pe ecran

Efectul de mișcare se obține prin afișarea unui desen într-o poziție, menținerea imaginii pe o durată scurtă, stergerea desenului și reafișarea lui în poziția următoare.

Menținerea limitată a imaginii se realizează cu comanda:

```
PAUSE n
```

Oprește execuția programului pe o durată dată de valoarea lui n = 0 - 65535

Pauza maximă este de aproximativ 22 minute: PAUSE 65535  
O pauză de cca 1 secundă va fi dată de PAUSE 50. Puteți opri programul pînă la apăsarea oricărei taste cu: PAUSE 0

Programul următor simulează săritura unei mingi lansată de o paletă. Vom folosi pentru minge litera "0" iar pentru paletă, semnul "/":

```

NEW
5 BORDER 1
10 REM mingea va fi lansată de 5 ori
20 FOR K = 1 TO 5
30 REM ridicarea mingii
40 FOR n = 21 TO 10 STEP -1
50 REM afișarea paletei
60 PRINT AT 21, 10; "/"
70 PRINT AT n, 10; "0".
80 PAUSE 5.
90 PRINT AT n, 10; " ".
100 NEXT n-
110 REM căderea mingii
120 FOR n = 10 TO 21
130 PRINT AT n, 10; "0"
140 PAUSE 5
150 PRINT AT n, 10; " "
160 NEXT n
170 NEXT k .

```

### Exemple de programe

Să realizăm un program pentru a contoriza numărul de aparitii ale cifrelor 1 la 7 (utilizate pentru culori) generate de funcția RND, dintr-un număr mare de execuții.

Dacă de exemplu cerem prin program generarea a 700 numere întregi aleatoare, între 1 și 7, ne-am aștepta ca să obținem fiecare cifra de 100 de ori. Este un mod de a vedea cum lucrează generatorul de numere aleatoare cu care este dotat calculatorul dvs.

#### - Analiza problemei

Se va genera un număr mare de numere aleatoare întregi (de ex. 700) între 1 și 7 și vor fi contorizate (însumate) aparitîile fiecărei dintre ele.

Rezultatele să fie afișate sub forma:

număr	de cîte ori
1	140
2	96
.	.
7	102

Pentru ca exemplul să cuprindă cît mai multe din noțiunile acestui capitol și pentru ca afișarea rezultatelor să fie cît mai expresivă, să se asocieze cifrelor 1 la 7 culorile corespunzătoare și să se obțină sub liniile cu rezultate, un grafic vertical cu șapte coloane a căror înălțime să fie proporțională cu numărul de aparitii pentru fiecare cifră (culoare).

Pentru a însuma numărul de aparitii să se folosească un tabel unidimensional cu 7 elemente declarat astfel:

```
10 DIM c(7)
```

#### - Codificarea programului:

```

10 DIM c(7)
15 PRINT FLASH 1; AT 10, 0; "AȘTEPTAȚI cca 20 sec.- calculez!";FLASH 0
18 REM generarea numerelor
20 FOR k = 1 TO 700
30 LET a = 1 + INT(7*RND)
35 REM contorizarea aparitîilor
40 LET c(a) = c(a) + 1
50 NEXT k
55 REM afișare rezultat
60 CLS: PRINT "număr", "de cîte ori"
70 PRINT
80 FOR p = 1 TO 7
90 PRINT PAPER p; INK 9; p,c(p)
100 NEXT p
110 REM graficul rezultatelor
120 BORDER 0
130 PLOT 155,0:DRAW PAPER 1; INK 9;0,c(1)
135 BEEP c(1)/100,c(1)
140 PLOT 163,0:DRAW PAPER 2; INK 9;0,c(2)
145 BEEP c(2)/100,c(2)
150 PLOT 171,0:DRAW PAPER 3; INK 9; 0,c(3)
155 BEEP c(3)/100,c(3)
160 PLOT 179,0: DRAW PAPER 4; INK 9; 0, c(4)
165 BEEP c(4)/100, c(4)
170 PLOT 187, 0:DRAW PAPER 5; INK 9; 0,c(5)
175 BEEP c(5)/100, c(5)
180 PLOT 195,0:DRAW PAPER 6; INK 9; 0,c(6)
185 BEEP c(6)/100, c(6)
190 PLOT 203,0 : DRAW PAPER 7; INK 9; 0,c(7)
195 BEEP c(7)/100,c(7)
200 STOP

```

Vă propunem să încercați înțelegerea programului răspunzind la întrebările următoare:

### INTREBĂRI RECAPITULATIVE

I 9.1 Ce instrucțiune a determinat:

- a) colorarea conturului ecranului?
- b) afișarea clipitoare a mesajului "AȘTEPTAȚI..." ?
- c) colorarea rezultatelor orizontale?

I 9.2 Care a fost efectul instrucțiunii INK 9?

I 9.3 Ce linie de program a trasat graficul vertical pentru aparitîile cifrei 3?

## CAPITOLUL 9

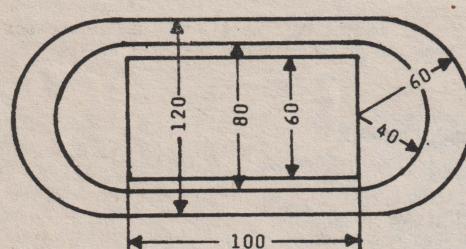
I 9.4 În ce linii de program trebuie să faceți modificări pentru a genera 10000 de numere întregi aleatoare?

I 9.5 Ce instrucțiune bănuiti că generează sunete proporcionale ca durată și înălțime cu numărul de apariții ale cifrelor? (căutați singura instrucțiune neexplcată pînă la acest capitol).

### Probleme

Realizați probleme pentru:

- generarea și afișarea a 20 de numere aleatoare între 0 și 100
- trasarea axelor x și y de lungime maximă, începînd cu colțul din dreapta sus
- umplerea ecranului cu puncte aleatoare, pe fond închis (ca un cer instelat)
- săritura amortizată a unei mingi (înălțimea să descrească pînă la zero)
- desenarea unui "stadion" de forma:



- obținerea literei grecești β prin apăsarea tastei "B"

### RĂSPUNSURI

R 9.1 a) 120 BORDER 0  
b) 15 PRINT FLASH 1; AT 10, 0;  
"AȘTEPTAȚI..."; FLASH 0  
c) 90 PRINT PAPER p...

R 9.2 Afișarea contrastă între INK și PAPER

R 9.3 Linia 150

R 9.4 În linia 15 pentru a modifica timpul de așteptare și în linia 20 care devine: 20 FOR k = 1 TO 10000

R 9.5 Instrucțiunea BEEP (vezi anexa C) care va fi explicată în capitolul 10.

## CAPITOLUL 10

### SĂ INTRODUCEM SUNETE

După cum vă amintiți din capitolul 1, CIP-ul este înzestrat cu un difuzor și este capabil să producă multe melodii pe placul dvs.

#### APLICAȚIE PRACTICĂ AP 10

1. Introduceți următorul program:

```
NEW
2 PRINT AT 10,1;FLASH;"ASCULTAȚI!"
3 PRINT:PRINT FLASH 0;"Cunoașteți melodia?"
5 FOR n=1 TO 3
10 BEEP .2,11:BEEP .2,12:BEEP .2,14:BEEP .2,11
20 BEEP .2,12:BEEP .2,9:BEEP .2,11:BEEP .2,7
30 BEEP .2,9:BEEP .2,12:BEEP .2,11:BEEP .2,9
40 BEEP .4,14:BEEP .4,14
50 BEEP .2,11:BEEP .2,12:BEEP .2,14:BEEP .2,11
60 BEEP .2,12:BEEP .2,9:BEEP .2,11:BEEP .2,7
70 BEEP .2,9:BEEP .2,12:BEEP .2,11:BEEP .2,9
80 BEEP .4,7:BEEP .4,7
90 NEXT n:CLS
RUN
```

Observați ce se întimplă... Fără a comenta prea mult, ați programat calculatorul să facă muzică, respectiv să interpreteze ceva foarte cunoscut, nu?

2. Tastați: NEW  
BEEP 1,0

Ce se întimplă? Pentru circa 1 secundă, pe ecran nu apare nici o schimbare, în schimb, în microdifuzorul incorporat se aude un sunet în care veți recunoaște nota DO din octava 1 (cei cu ureche muzicală, bineînțeles).

Mai încercați: NEW  
10 PRINT "BEEP 2,-20":BEEP 2,-20  
20 PRINT "BEEP 2,30":BEEP 2,30  
30 PRINT "BEEP 4,30":BEEP 4,30:CLS  
RUN

Sesiizați că înălțimea diferă, în funcție de valoarea celui de al doilea argument din BEEP, iar durata sunetului este dată de primul argument.

### NOTIUNI DE BAZĂ DESPRE PRODUCEREA SUNETELOR

Producerea sunetelor se face cu instrucțiunea BEEP, care are forma generală: BEEP d,i  
> d (primul argument) reprezintă durata sunetului. Durata în secunde are valori permise între 0 și 10.  
> i (al doilea argument) indică înălțimea sunetului (frecvența), și are valori permise între -60 și +69.  
Folosirea unor valori în afara limitelor produce o eroare care se soldează cu întreruperea executiei și un mesaj de eroare.

#### Codificarea notelor

Corespondența între valorile lui i și gama muzicală (temperată) este dată de următoarea regulă:  
Notei DO din octava 1 (DO CENTRAL de pe claviatura unui pian) îi corespunde valoarea 0 mergînd în sus (DO DIEZ, RE, RE DIEZ ...) se creste valoarea lui i cu cîte o unitate, mergînd în jos (SI, SI BEMOL, LA ...) se scade cîte o unitate pînă la limitele date mai sus. Precizăm că în gama temperată DO DIEZ este aceiași notă cu RE BEMOL etc..., respectiv diezii și bemolii au ca efect adunarea sau scăderea unei unități la valoarea corespunzătoare notei (similar ca la pian).

FA#	SOL	LA#	DO#	RE#	MI	FA	SOL	LA#	DO
-6	-4	-2	1	-3	4	5	7	9	11

## CAPITOLUL 10

### Exemple de programare

Să încercăm pentru început să punem calculatorul să cînte arpegiul gamei DO MAJOR. Avem nevoie de notele DO, MI, SOL, DO și să convenim o durată de .2 secunde pentru toate notele. Programul ar trebui să fie (sau să semene cu):

```
NEW
5 REM arpegiul gamei DO MAJOR
10 BEEP .2,0
20 BEEP .2,4
30 BEEP .2,7
40 BEEP .2,12
50 BEEP .2,12
60 BEEP .2,7
70 BEEP .2,4
80 BEEP .2,0
RUN
```

### Schimbarea octavei

Pentru arpegiul aceleiași game, dar în altă octavă, ar trebui să adunăm sau să scădem multiplii de 12 (sînt siguri că atîi observat că între doi de DO sînt 12 unități diferență). Respectiv, pentru octava 2, ar trebui să folosim valorile 12, 16, 19, 24.

Mai elegant însă ar fi să scriem programul astfel:  
(editați programul anterior, linie cu linie, pentru a-l obține pe acesta)

```
5 REM arpegiul gamelor DO MAJOR
10 BEEP .2,0+n
20 BEEP .2,4+n
30 BEEP .2,7+n
40 BEEP .2,12+n
50 BEEP .2,12+n
60 BEEP .2,7+n
70 BEEP .2,4+n
80 BEEP .2,0
```

și să-i dăm lui n valori de forma + 12\*k avînd grijă să nu ieșim din limitele permise. Ca rafinament suplimentar, punem următorul program (procedează la fel, prin editarea programului anterior):

### Schimbarea duratei

```
5 REM arpegiul gamelor DO MAJOR
10 BEEP .2xm,0+n
20 BEEP .2xm,4+n
30 BEEP .2xm,7+n
40 BEEP .2xm,12+n
50 BEEP .2xm,12+n
60 BEEP .2xm,7+n
70 BEEP .2xm,4+n
80 BEEP .2xm,0+n
```

și să-i dăm lui m diverse valori avînd deasemeni grijă să nu ieșim din gama valorilor permise.

### Schimbarea tempou-lui

Am realizat "parametrizarea" programului pentru diverse durate ale notelor și diverse octave. Dacă am conveni, de exemplu că .2 secunde este durata unei optimi, dînd diverse valori (intregi sau nu) lui m, am putea interpreta efectul prin schimbarea tempo-ului, sau înlocuirea optimilor cu doimi, note întregi, șaisprezecimi etc...

O utilizare a acestui mod de programare a muzicii ar fi posibilitatea "acordării" calculatorului cu alte instrumente.

Ca soluție de "codificare" a unei partituri, sugerăm să completați portativul cu o linie dedesubt și una deasupra, și să scrieți cu creionul în dreptul liniilor și spațiilor (ținînd cont de armura cheii) valorile corespunzătoare notelor.

Încă o treabă foarte importantă: cum codificăm pauzele? o soluție ar fi folosirea instrucțiunii PAUSE n, ținînd cont că n produce o pauză de 20\*n milisecunde. (Deci PAUSE 50 este echivalent cu BEEP 1...).

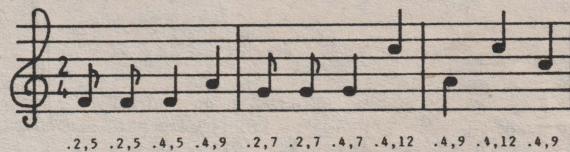
Sau altfel, se poate folosi un BEEP cu durata

corespunzătoare și o valoare pentru înlătîme care să iasă din spectrul practic audibil (ex.69). În acest mod uniformizăm regula de durată și pentru pauze.

### Exemplul 10.2

Să urmărim programul de mai jos care codifică partitura "Cucule pasăre sură" după Ioan D.Chirescu.  
Iată începutul:

```
10 BEEP .2,5:BEEP .2,5:BEEP .4,5:BEEP .4,9
20 BEEP .2,7:BEEP .2,7:BEEP .4,7:BEEP .4,12
30 BEEP .4,9:BEEP .4,12:BEEP .4,9
40 BEEP .2,5:BEEP .2,5:BEEP .4,9:BEEP .4,9
50 BEEP .2,7:BEEP .2,7:BEEP .4,7:BEEP .4,12
60 BEEP .4,9:BEEP .4,12:BEEP .4,9
70 BEEP .2,12:BEEP .2,12:BEEP .4,12:BEEP .4,14
80 BEEP .2,10:BEEP .2,10:BEEP .4,10:BEEP .4,12
85 BEEP .2,9:BEEP .2,9:BEEP .4,9:BEEP .4,5
90 BEEP .2,7:BEEP .2,7:BEEP .4,7:BEEP .4,12
100 BEEP .4,9:BEEP .4,12:BEEP .4,9
110 BEEP .2,12:BEEP .2,12:BEEP .4,12:BEEP .4,14
120 BEEP .2,10:BEEP .2,10:BEEP .4,10:BEEP .4,12
130 BEEP .2,9:BEEP .2,9:BEEP .4,9:BEEP .4,5
140 BEEP .2,7:BEEP .2,7:BEEP .4,7:BEEP .4,12
145 BEEP .4,5:BEEP .4,12:BEEP .4,9
```



Introduceți programul și cu RUN însărcinați calculatorul cu deschiderea partiturii. Melodia care se aude nu se poate să nu o recunoașteți. Dacă dorîți, vă puteți folosi la introducere de "parametrizarea" tempo-ului și octavei. Putem folosi ca nume de variabile numele notelor.

```
5 READ o,p,fa,sol,la,si b,do,re
8 DATA .2,.4,5,7,9,10,12,14
10 BEEP o,fa:BEEP o,fa:BEEP p,fa:BEEP p,la
20 BEEP o,sol:BEEP o,sol:BEEP p,sol:BEEP p,la
30 BEEP p,la:BEEP p,la:BEEP p,la
40 BEEP o,fa:BEEP o,fa:BEEP p,fa:BEEP p,la
```

Continuați...

Mai facem o precizare pentru cei care doresc să facă din acest domeniu un cîmp de aplicații mai serioase: există programe utilitare specializate, aşa numite editoare de muzică cu care se poate ajunge la performanțe mai ridicate în componistica asistată de calculator.

Deasemeni mai precizăm că la frecvențe joase, sunetele ies din clasa celor "muzicale" putînd fi folosite pentru efecte sonore interesante.

## INTREBĂRI RECAPITULATIVE ȘI EXERCȚII

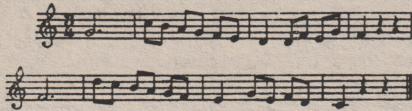
- Î 10.1 Care este forma generală a instrucțiunii BASIC de prelucrare a sunetelor?
- Î 10.2 Care sunt limitele permise pentru cele două argumente ale instrucțiunii BEEP?
- Î 10.3 Cărei note îi corespunde valoarea 0 a celui de al doilea argument al instrucțiunii BEEP?
- Î 10.4 Care este regula de construire a codificării notelor, plecînd de la corespondența Do central=0?

### Probleme

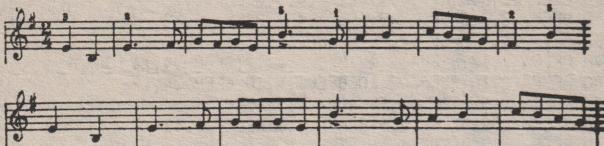
1. Scrieți un program care să cînte gama DO MAJOR în ambele sensuri.

## CAPITOLUL 10

2. Combinăți BEEP cu RND pentru a genera sunete de durate și înălțimi aleatoare, pe un timp nedefinit (întrerupeți cu (BREAK!).  
 3. Codificați următoarea partitură:



4. Codificați următoarea partitură:



### RĂSPUNSURI

R 10.1 BEEP d,i

R 10.2 0 < d < 10      -60 < i < +69

R 10.3 DO central

R 10.4 Față de DO central se adună 1 spre dreapta și se scade 1 spre stânga, pentru fiecare notă din gama temperată (clapa de pian).

### Problema 1

```
10 BEEP .2,0 :BEEP .2,2:BEEP .2,4:BEEP.2,5
20 BEEP .2,7 :BEEP .2,9:BEEP .2,11:BEEP.2,12
30 BEEP .2,12 :BEEP .2,11:BEEP .2,9:BEEP.2,7
40 BEEP .2,5 :BEEP .2,4:BEEP .2,2:BEEP.2,0
```

### Problema 2

```
10 BEEP RND*.8, -60 + RND*129
20 GO TO 10
```

### Problema 3

```
5 LET d=...
10 BEEP 3xd,7 : BEEP d,12 :BEEP d,11
20 BEEP d,9 :BEEP d,7 :BEEP d,5:BEEP d,4
30 BEEP 2xd,2 :BEEP d,2:BEEP d,5 :BEEP d,4
40 BEEP d,7:BEEP 2xd,5:BEEP 2xd,69:BEEP 2xd,69
50 BEEP 3xd,5:BEEP d,14:BEEP d,12:BEEP d,11
60 BEEP d,9 :BEEP d,7 :BEEP d,5
70 BEEP 2xd,4:BEEP d,7:BEEP d,4:BEEP d,5:BEEP d,2
80 BEEP 2xd,0 :BEEP 2xd,69 :BEEP 2xd,69
```

Se va alege pentru d o valoare convenabilă pentru durata unei optimi (ex: .3).

### Problema 4

```
5 LET d=...
10 BEEP 2xd,4 : BEEP 2xd,-1
20 BEEP 3xd,4:BEEP d,6:BEEP d,7:BEEP d,6:BEEP d,7
30 BEEP d,4:BEEP 3xd,11:BEEP d,7:BEEP 2xd,9
40 BEEP 2xd,11:BEEP d,12:BEEP d,11:BEEP d,9
45 BEEP d,7:BEEP 2xd,6:BEEP 2xd,11:BEEP d,9
50 BEEP d,7 :BEEP d,6 :BEEP d,7
60 BEEP 2xd,4 :BEEP 2xd,-1:BEEP 3xd,4 :BEEP d,6
65 BEEP d,7 :BEEP d,6 :BEEP d,7 :BEEP d,4
70 BEEP 3xd,11 :BEEP d,7
75 BEEP 2xd,9 : BEEP 2xd,11 :BEEP d,12 :BEEP d,11
80 BEEP d,9 :BEEP d,7
90 BEEP 2xd,7 :BEEP 2xd,6 :BEEP 4xd,4
```

Se va alege pentru d ca și în exemplul precedent, o valoare convenabilă (ex: .3).

## CAPITOLUL 11

### FUNCTII

BASIC-S vă pune la dispoziție anumite funcții pe care le puteți folosi în expresii introducând numele funcției urmat de o valoare numit argument.

Efectul este că în calcularea expresiei respective numele de funcție va fi înlocuit cu valoarea funcției corespunzătoare valorii argumentului.

### APLICAȚIE PRACTICĂ AP 11

Să introducem următorul program:

```
10 FOR x = 0 TO 2* PI STEP .1
20 PRINT "PENTRU x=";x;" RADIANI"
30 PRINT TAB 4;"SIN(x)=";SIN(x)
40 PRINT TAB 4;"COS(x)=";COS(x)
50 NEXT x
RUN
```

Se va observa că pe ecran apar valorile lui SIN(x) și COS(x). Deasemeni, se observă că în linia 10 apare PI, semnificând funcția constantei cu acest număr, respectiv: 3,1415927, deasemeni SIN și COS, apelul lor provocând furnizarea unor valori numerice pe care le-am folosit în liniile 10, 30 și 40.

### NOTIUNI DE BAZĂ

Funcțiile trigonometrice SIN, COS, TAN, ASN, ACS, ATN:  
 Forma generală a acestor funcții este:

[ n ] SIN x      sau      [ n ] SIN(x)

variabilă pentru toate cele trei funcții: SIN, COS, TAN și inversele lor.

Semnificația lor este cea cunoscută din trigonometrie, argumentul funcțiilor directe este o expresie numerică în radiani. Toate aceste funcții se pot utiliza oriunde este acceptată o expresie numerică.

Exemplificare:

PRINT SIN(0)	va afișa 0
PRINT SIN(PI/2)	va afișa 1
PRINT SIN PI/2	va afișa 0
PRINT ASN (1)	va afișa 1.5707963
PRINT ASN 1	va afișa 1.5707963
PRINT TAN(ATN(3))	va afișa 3

Se observă că folosirea parantezelor poate schimba valorile obținute. Vă recomandăm ca atunci când argumentul este o expresie, aceasta să fie inclusă între paranteze. Altfel rezultatul poate să nu fie cel mai corect.

### Funcția PI

Forma generală este: PI și are ca efect obținerea valorii numărului (3.1415927 ...). Se poate folosi oriunde avem nevoie de o valoare mai exactă a numărului fără a-i scrie noi efectiv atâtea zecimale cîte avem nevoie. (Calculatorul îl generează cu 7 zecimale).

Funcțiile ABS, SGN, INT, SQR, EXP, LN, BIN:

Forma generală pentru ABS:

[ n ] ABS x      sau      [ n ] ABS (x)

valabilă pentru toate funcțiile enumerate, argumentul x fiind un număr sau o expresie numerică.

ABS - returnează valoarea absolută a expresiei argument calculată.

Exemple:	PRINT ABS (-10)	va afișa 10
	PRINT ABS (10)	va afișa 10
	PRINT ABS (10-25)	va afișa 15

**SGN** - returnează: 1 dacă argumentul > 0  
0 dacă argumentul = 0  
-1 dacă argumentul < 0

Exemple: PRINT SGN (-10) va afișa -1  
LET x=15  
PRINT SGN x va afișa 1  
PRINT SGN (-x) va afișa -1  
PRINT SGN (x-x) va afișa 0  
PRINT SGN x-x va afișa -14 (de ce?)

**INT** - returnează partea întreagă a expresiei argument.  
Exemple: INT (3.1415927) va returna 3  
INT (-3.1415927) va returna -3

**SQR** - extrage rădăcina pătrată.

Exemple: SQR 9 va returna 3  
SQR -9 va produce un mesaj de eroare

**EXP** - calculează valoarea funcției exponentiale e<sup>x</sup>.

Exemple: PRINT EXP 10 va returna 22026,466, care reprezintă puterea a-10-a a numărului e.  
PRINT EXP 3.14 va returna 23,103867 deci e la puterea 3.14

**LN** - calculează valoarea funcției LN x

Exemple: PRINT LN 10 va afișa 2.3025851  
PRINT LN(EXP(10)) va afișa 10

**BIN** xxxxxxxx - această funcție specifică faptul că urmează un număr în reprezentarea binară pe un octet.  
Are ca argument un sir de 8 cifre binare.

**POINT** are forma: [n] POINT (x,y) și la apel, returnează:  
0 - dacă punctul de pe ecran din poziția x,y are culoarea (INK) identică cu a fundalului (PAPER).  
1 - în caz contrar.

#### Functiile: LEN, STR\$, VAL, VAL\$, CHR\$, CODE

**LEN** Funcția LEN având ca argument un sir sau o expresie de tip sir de caractere returnează numărul caracterelor din sir.

Exemplu: PRINT LEN ("ABCDE") va afișa 5

**STR\$** Funcția STR\$ are argument numeric și returnează un sir de caractere care reprezintă numărul.

Exemplu: PRINT STR\$ 10 va returna 10  
PRINT STR\$ (2E3) va returna 12000

**VAL** Funcția VAL are efectul invers; are ca argument un sir de caractere numerice și returnează un număr.

Exemplu: PRINT VAL "34.756" va afișa 34.75  
PRINT VAL "3E4" va afișa 30000  
PRINT VAL ("2"+"3") va afișa 6

**VAL\$** Funcția VAL\$ este similară dar are ca rezultat tot un sir.

Exemplu: VAL\$ ""abcde"" va afișa "abcde"

**CHR\$** Funcția CHR\$ are ca argument un număr și returnează caracterul care are cod acel număr (v. și anexa A).

Exemplu: PRINT CHR\$(42) va afișa \*

**CODE** Funcția CODE este inversa funcției CHR\$. Ea are ca argument un sir și întoarce codul primului caracter al sirului.

Exemplu: PRINT CODE "\*" va afișa 42

**INKEY\$** Funcția INKEY\$ returnează caracterul corespunzător tastei apăsată ultima dată. Dacă nu s-a apăsat nici o tastă, se returnează sirul vid.

Exemplu: 10 PRINT INKEY\$; :PAUSE 10  
20 GO TO 10  
va afișa caracterul de pe tastă pe care ați apăsat-o sau un sir vid.

**SCREEN\$** Această funcție se folosește în cadrul lui LOAD și SAVE și înlocuiește adresa de început a zonei ecran în memorie și numărul de octetii ai acesteia, precedate de CODE. Este utilă în LOAD și SAVE. SCREEN\$ <=> CODE 16384,6912  
LOAD "SCREEN\$" este echivalent cu LOAD "CODE 16384,6912"

#### Functii definite de utilizator: DEF FN și FN

În afară de funcțiile enumerate pe care calculatorul vi le pune la dispoziție, aveți posibilitatea să vă definiți funcții proprii, corespunzătoare programelor pe care le scrieți. Pentru aceasta se alege un nume, format dintr-o literă sau dintr-o literă urmată de \$ dacă funcția va returna ca valoare un sir de caractere, și o listă de argumente.

Definirea se face cu: [n] DEF FN nume(x1,x2...xn)= expresie unde prin expresie se înțelege o expresie numerică sau de tip sir de caractere, care poate la rîndul ei să conțină apeluri de funcții.

Trebue notat că argumentele funcției sunt în număr de maximum 26 numerice și 26 de tip sir. De asemenea, ca notă specială, trebuie reținut că variabilele funcției sunt numite "legate", în sensul că valabilitatea lor este restrînsă la linia de definire a funcției, neavind nici o legătură cu o altă posibilă variabilă cu același nume care apare în altă parte în program.

De exemplu, în următorul program:

```
10 LET x=0:LET y=10
20 LET a=15
30 DEF FN m(x,y)=x+y+xxy+a
40 DEF FN n()=x+y+xxy+a
50 PRINT FN m(2,5)
60 PRINT FN n()
```

La execuția liniei 50, valorile 2 și 5 sunt trecute pe seama variabilelor x și y din linia 30 fără a altera pe cele din linia 10. Pentru evaluarea funcției m, rezultind valoarea 32 (verificați!), care se afișează. La evaluarea liniei 60, se apelează funcția n definită în linia 40, x și y fiind de data aceasta variabilele libere x și y definite în linia 10 cu valorile 0 și 10 deci rezultatul funcției, deși are aceeași expresie, va fi altul - cel care se va afișa pe ecran (25).

#### Exerciții recapitulative

E 11.1 Să se scrie un program de afișare a valorilor funcțiilor EXP și LN.

E 11.2 Să se afișeze pentru fiecare x valorile expresiei:

$$e = \begin{cases} x-1 & \text{pt. } x < 0 \\ 0 & \text{pt. } x = 0 \\ -x+1 & \text{pt. } x > 0 \end{cases}$$

pentru x între -100 și +100 cu pasul 2.

E 11.3 Folosind funcția BIN să se memoreze într-o variabilă vector codurile binare ale literelor mici de la a la h (vezi anexa la manual), după care să se afișeze caracterele respective.

E 11.4 Să se calculeze și să se afișeze pentru fiecare x folosind funcții utilizator valorile funcției:

$$e = \begin{cases} x+0.5 \ln x & \text{pentru } x \in (0,1] \\ \frac{2x + \ln x}{2} & \text{pentru } x \in (1,10] \\ \frac{2x^2 + 2x + \ln x + \ln x}{2} & \text{pentru } x > 10 \end{cases}$$

E 11.5 Afisați valorile funcției POINT pentru 10 puncte de pe ecran alese aleator.

## CAPITOLUL 11

### PROBLEME

P 11.1 Să se scrie un program care să calculeze rădăcinile ecuației de gradul II (toate cazurile).

P 11.2 Să se calculeze și să se afișeze pentru fiecare  $x$  valorile funcției:

$$f(x) = \begin{cases} \max & \{ 1+xxx, 1-x, \cos x \} \text{ pentru } x \leq 0 \\ e^x & \text{pentru } 0 < x < 1 \\ 0 & \text{în rest pentru} \\ & \text{intervalul } -10 \text{ și} \\ & +2 \text{ cu pasul .2} \end{cases}$$

P 11.3 Desenați pe ecran 3 cicloide.  
Indicație: Cicloida este definită de ecuațiile

$y=r-\cos x$   
 $x=r+\sin x$   
 și se va folosi funcția PLOT  
 Vă recomand:  $x$  între 0 și 22 cu pasul 1/10  
 $r=10, d=20$   
 și în loc de  $y$  se va calcula și folosi un  
 $y=y+10$  pentru a desena cicloida ceva mai sus  
 pe ecran.

### RĂSPUNSURI

R 11.1 10 FOR  $x=-100$  TO  $80$  STEP  $5$   
 20 LET  $E=EXP x$   
 30 LET  $L$=""$   
 40 IF  $x > 0$  THEN LET  $L$ = LN(x) = " + STR$(LN(x))$   
 50 PRINT " $x="; x;" e<sup>↑</sup> x="; 1$"  
 60 NEXT  $x$$

R 11.2 10 FOR  $x=-100$  TO  $100$  STEP  $2$   
 20 IF  $x > 0$  THEN GO TO  $80$   
 30 IF  $x = 0$  THEN GO TO  $60$   
 40 LET  $e=x-1$   
 50 GO TO  $90$   
 60 LET  $x = 0$   
 70 GO TO  $90$   
 80 LET  $e=-x+1$   
 90 PRINT " $x="; x;" e="; e"  
 100 NEXT  $x$$

R 11.3 10 DIM a\$(8)  
 20 LET a\$(1)=CHR\$ BIN 01100001  
 30 LET a\$(2)=CHR\$ BIN 01100010  
 40 LET a\$(3)=CHR\$ BIN 01100011  
 50 LET a\$(4)=CHR\$ BIN 01100100  
 60 LET a\$(5)=CHR\$ BIN 01100101  
 70 LET a\$(6)=CHR\$ BIN 01100110  
 80 LET a\$(7)=CHR\$ BIN 01100111  
 90 LET a\$(8)=CHR\$ BIN 01101000  
 100 FOR  $i=1$  to  $8$   
 110 PRINT a\$(i)  
 120 NEXT  $i$

R 11.4 10 DEF FN f( $x$ )= $x+0.5+LNx$   
 20 DEF FN g( $x$ )= $(2xx+(LNx))/2$   
 30 FOR  $x=.2$  TO  $70$  STEP  $.2$   
 40 IF  $x > 1$  THEN GO TO  $70$   
 50 LET  $e=FN f(x)$   
 60 GO TO  $110$   
 70 IF  $x > 10$  THEN GO TO  $100$   
 80 LET  $e=FN g(x)$   
 90 GO TO  $110$   
 100 LET  $e=(x+1)*FN g(x)$   
 110 PRINT " $x=$  ";  $x$ ; "  $e=$  ";  $e$   
 120 NEXT  $x$

R 11.5 10 FOR  $I=1$  TO  $10$   
 20 LET  $L=INT(RND*175)$   
 30 LET  $C=INT(RND*255)$   
 40 PRINT "POINT(C,L)"; POINT (C,L)  
 50 NEXT  $I$   
 și lansați de mai multe ori cu RUN

P 11.1 10 REM REZOLVAREA ECUAȚIEI DE GRADUL II  
 20 CLS  
 30 PRINT "REZOLVAREA ECUAȚIEI DE GRADUL II, DE FORMA  
 $ax^2+bx+c=0$ "  
 40 INPUT "INTRODUCETI VALOAREA a:"; a  
 45 IF  $a = 0$  THEN GO TO  $190$   
 50 INPUT "INTRODUCETI VALOAREA b:"; b  
 60 INPUT "INTRODUCETI VALOAREA c:"; c  
 70 LET  $D=b*x^2-4*a*c$   
 80 IF  $D \geq 0$  THEN GO TO  $110$   
 90 PRINT "ECUAȚIA NU ARE RĂDĂCINI REALE"  
 100 STOP  
 110 IF  $D > 0$  THEN GO TO  $150$   
 120 LET  $x=-b/(2*a)$   
 130 PRINT "RĂDĂCINĂ UNICĂ, , x="; x  
 140 STOP  
 150 LET  $x1=(-b+SQR(D))/(2*a)$   
 160 LET  $x2=(-b-SQR(D))/(2*a)$   
 170 PRINT "x1="; x1; "x2="; x2  
 180 STOP  
 190 PRINT "a = 0 ecuația degenerată"  
 200 GO TO  $40$

P 11.2 10 FOR  $x=-10$  TO  $2$  STEP  $.2$   
 20 IF  $x > 0$  THEN GO TO  $70$   
 30 LET  $f=1+x*x$   
 40 IF  $f < (1-x)$  THEN LET  $f=1-x$   
 50 IF  $f < \cos x$  THEN LET  $f=\cos x$   
 60 GO TO  $110$   
 70 IF  $x > 1$  THEN GO TO  $100$   
 80 LET  $f=EXP x$   
 90 GO TO  $110$   
 100 LET  $f=0$   
 110 PRINT "x="; x; " f="; f  
 120 NEXT  $x$

P 11.3 10 FOR  $f=0$  TO  $23$  STEP  $.2$   
 20 LET  $x=10*f-20*\sin f$   
 30 LET  $y=(10-20*\cos f)+10$   
 40 PLOT INK2, x, y  
 50 NEXT  $f$

## CAPITOLUL 12

### CUM LUCRĂM DIRECT CU MEMORIA CIP-ULUI

Pentru acei dintre dvs. care au asimilat bine noțiunile prezentate pînă acum și reușesc să scrie ușor programe în BASIC-S, există o mare tentație de a intra tot mai adinc în tainele CIP-ului. Pentru aceasta aveți la dispoziție anumite funcții / instrucțiuni care vă permit să lucrați direct cu memoria CIP-ului, să introduceți informații direct în ea, să citiți conținutul ei și să vedeați cum sunt reprezentate caracterele etc.

### APLICAȚIE PRACTICĂ AP12

#### 1. Introduceți:

```
5 REM program pentru afișarea setului de caractere
10 PRINT "CARACTER", "COD":PRINT
20 FOR c=32 TO 255
30 PRINT CHR$ c,c
40 PRINT
50 NEXT c
RUN
```

Notați codurile pentru următoarele caractere (priviți Anexa A)

caracter	cod
spațiu (blank)	---
1	---
=	---
A	---
a	---
RND	---
SIN	---
PEEK	---
STOP	---
RANDOMIZE	---
COPY	---

#### 2. Introduceți și executați:

```
NEW
20 PRINT "x+y="
30 PRINT "adresa", "conținut"
40 PRINT "-----", "-----"
50 FOR a=23759 TO 23765
60 PRINT a, PEEK a:PRINT
70 NEXT a
RUN
```

Notați rezultatele și folosind Anexa A scrieți caracterele corespunzătoare codurilor:

adresa	conținut	caracter
23759	.	-----
23765		

### NOȚIUNI DE BAZĂ

Memoria internă a CIP-ului o putem imagina ca un fișet cu o mulțime de sertare numerotate unul după altul, începînd cu sertarul avînd numărul 0, urmat de sertarul numărul 1, apoi de cel cu numărul 2 și.a.m.d. pînă la ultimul sertar de la sfîrșitul memoriei.

Un asemenea sertar poartă numele de LOCATIE DE MEMORIE și numărul sau reprezintă ADRESA la care se poate memora orice caracter din setul de 255 pe care le cunoaște CIP-ul.

Caracterele se memorează prin codurile cuprinse în Anexa A, dar de fapt în circuite ele sunt memorate sub formă binară prin combinații între două cifre:

0 - BIT zero  
1 - BIT unu      (bit=Binary digit - cifră binară)

Exemplu:	cod zecimal	caracter	binar
	32	spațiu	00100000
	43	+	00101011
	70	F	01000110
	102	f	01100110
	241	LET	11110001

Observați că pentru a fi exprimat un caracter se folosesc 8 biți (cifre binare) formînd un OCTET (în engleză: BYTE).

Reanalizați cuprinsul memoriei de la AP 12 - exercițiu 2:

adresa	cod	caracter	octet
23759	245	PRINT	11110101
23760	34	"	00100010
23761	120	x	01111000
23762	43	+	00101011
23763	121	y	01111001
23764	61	=	00111101
23765	34	"	00100010

Orice octet poate fi localizat în memorie prin ADRESA sau LOCATIA sa.

Memoria CIP-ului poate cuprinde un număr maxim de 65536 octeți primul avînd adresa 0, iar ultimul 65535 (pentru că numărătoarea începe cu zero).

Capacitatea memoriei se exprimă în KILOOCTEȚI / KILOBYTES 1KO = 1024 octeți.

Memoria maximă a CIP-ului este deci de 64 KO.

Memoria calculatorului dvs. este de două feluri:

RAM (Random Acces Memory) și  
ROM (Read Only Memory)

Informația scrisă în ROM rămîne nealterată la decuplarea alimentării CIP-ului. În schimb, informația din RAM se pierde la scoaterea de sub tensiune.

### Memoria cu BASIC-S

Memoria ROM este invalidată, arhitectura RAM-ului avînd următorul conținut:

BASIC-S	MEMORIE pentru ECRAN	.Variabile de sistem
adr:0 16383	16384	.programme BASIC 23295 23296 65535

#### ► Acces la o locație de memorie

Pentru început să vedem cum putem avea acces la memorie. Introduceți următorul program:

```
10 POKE 32000,128
20 PRINT PEEK 32000
25 STOP
30 POKE 32000,255
40 PRINT PEEK 3200
```

Instrucțiunea din linia 10 introduce în locația de memorie cu adresa 32000 numărul 128, iar instrucțiunea din linia 20 afișează conținutul locației de memorie de la adresa 32000. Lansarea în execuție a programului, cu comandă RUN va avea ca efect afișarea pe ecran a numărului 128.

Executînd însă programul cu RUN 30, pe ecran va fi afișată valoarea 255.

Deci s-a modificat informația scrisă în RAM la adresa 32000 înlocuind valoarea 128 cu 255. Funcția POKE scrie în memorie la adresa specificată un octet de informație, iar funcția PEEK extrage de la adresa specificată un octet de informație

## CAPITOLUL 12

Sintaxa: POKE adresa, cod  
adresa=0...65535 0 <= cod <=255  
PEEK adresa

### RAM VIDEO

Memoria pentru ecran conține informația care va fi afișată pe ecranul televizorului.

Introduceți: 10 CLS  
20 FOR i=16384 TO 16384 + 6144  
30 POKE i, 255  
40 PRINT AT 10,10;i  
50 NEXT i  
RUN

Să lansăm în execuție programul. Ecranul va începe să se umple cu linii negre și în centrul lui va fi afișată adresa locației de memorie ce va fi încărcată cu 255., adică un octet în care toți biți sănt 1. Dacă vom modifica linia 30 înlocuind astfel: 30 POKE i,15

la rularea programului pe ecran va apărea o structură de linii albe și negre, octetul 15 având structura 00001111.

Fiecare caracter este matrice de 8 × 8 pixeli. Fiecare asemenea matrice are atribute de culoare care sănt poziționate începând de la adresa 22528 pe o lungime de 22 × 24.

Fiecare caracter îi este asociat un octet care conține atributele.

Structura octetului de atribute poate fi determinată astfel:

atribut = INK + 8 × PAPER + 64 × BRIGHT + 128 × FLASH

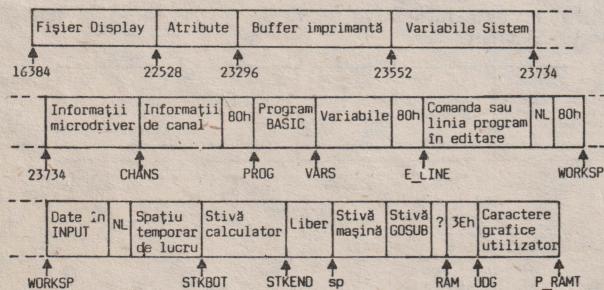
Încercați acum următorul program:

```
10 CLS
20 PRINT AT 10,10;"A"
30 POKE 22528+10*32+10,121
RUN
```

În linia 30 a acestui program am calculat poziția octetului de atribute pentru caracterul tipărit prin comanda din linia 20.

### VARIABLE DE SISTEM

Pentru a vorbi despre zona variabilelor de sistem să prezintăm mai în amănunt structura memoriei.



Variabilele de sistem sănt explicate în anexa F.

### CLEAR

Instrucțiunea CLEAR fixează noul RAM -TOP, adică adresa maximă alocată de calculatorul dvs. pentru un program BASIC. Atenție! La o comandă CLEAR se poate șterge conținutul memoriei.

În plus, comanda CLEAR are următoarele efecte:

- altereați conținuturile variabilelor de sistem;
- șterge VIDEO-RAM
- reseteați pozițiile PLOT
- execută o comandă RESTORE

Sintaxa: CLEAR adresa

Exemplu: 10 CLEAR 40000  
RUN

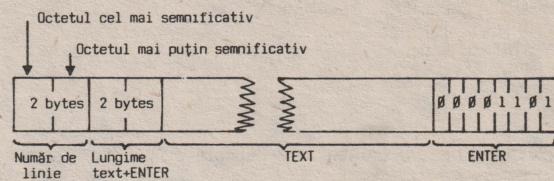
După execuția acestei linii zona de memorie alocată pentru programul BASIC este limitată pînă la adresa 40000.

### NEW

Comanda NEW șterge memoria RAM pînă la adresa RAM -TOP. Zona UDG rămîne neschimbată.

### Liniile BASIC

Informația este stocată în memorie după diferiți algoritmi. De exemplu în continuare este prezentat modul de organizare al unei linii BASIC în memoria calculatorului.



### Cod mașină

Și acum mai mult despre programele scrise în cod mașină. Calculatorul are un procesor care execută anumite comenzi. Comenzi sănt indicate prin codurile lor. Octetii ce reprezintă codurile se găsesc stocăți în memoria calculatorului. De exemplu introduceți următorul program:

```
10 FOR i = 0 TO 17
20 READ adr
30 POKE 32000+i,adr
40 PRINT 32000+i: PEEK(32000+i)
50 NEXT i
60 DATA 62,2,205,1,22,62,22,215,62,10,215,62,10,215,
62,49,215,201
RUN
```

Pe ecran vor apărea în stînga adresele de memorie începînd de la 32000 iar în dreapta conținutul lor. Cei 18 octeti între 32000 și 32017 conțin o rutină echivalentă cu:

PRINT AT 10,10;1

Pentru a executa programul scris în cod mașină se folosește comanda RANDOMISE USR 32000.

### RANDOMISE USR

Să încercăm acum comanda RANDOMISE USR 32000. La execuția programului, pe ecran va apărea cifra 5. Acesta a fost un program scris în cod mașină. Programatorii experti pot folosi un ASAMBLOR adică un program care preia mnemonicele asociate codurilor de instrucțiune și le transformă într-un format direct executabil.

### SAVE CODE

Pentru salvarea pe casetă a rutinelor scrise în cod mașină se folosește instrucțiunea:

SAVE "nume" CODE adresa,nr.octetăi

### LOAD CODE

Pentru încărcarea programului în memorie: LDAD "nume program" CODE adresa de încărcare. Dacă dorîți verificarea unui program salvat introduceți:

VERIFY "nume" CODE

**LUCRUL CU PERIFERICE**

Și acum să vedem cum comunică un calculator cu lumea înconjurătoare.

Ce este un PORT intrare/ieșire

Microprocesorul Z80 consideră perifericele fie ca o zonă de memorie (RAM-VIDEO - pentru ecran) fie ca un port de intrare (tastatură, casetofon) sau de ieșire (BORDER, DIFUZOR, CASE-TOFON). Dacă la RAM-VIDEO am văzut cum puteți avea acces, să discutăm modul de acces al CIP-ului la porturile de intrare/ieșire.

Calculatorul dvs. poate avea în configurație 256 porturi notate de la 0 la 255.

În arhitectura hardware a CIP-ului un alt rol important are portul 238 care permite validitatea / invaliditatea ROM-ului.

**OUT și IN**

Introduceți: NEW  
 10 INPUT "culoarea BORDER?", i  
 20 OUT 254, i  
 30 GO TO 10  
 RUN

Alegind pentru variabila i valori între 0 și 7 vom putea modifica culoarea BORDER-ului. Observați schimbarea BORDER-ului!

NEW  
 10 INPUT i  
 20 OUT 254,10  
 30 OUT 254,0  
 40 PAUSE i  
 50 GO TO 10  
 RUN

Dacă se atribuie lui i valoarea 50 difuzorul va emite un bîrfit la fiecare secundă. Dacă i se modifică, se va schimba corespunzător și perioada semnalului sonor.

Instrucțiunea pentru transmiterea unei date la un port de ieșire are

SINTAXA: OUT adresa,octet      adresa=adresa portului

Pentru a citi de la un port de intrare-ieșire se folosește instrucțiunea: IN adresa

**ÎNTREBĂRI RECAPITULATIVE**

Î 12.1 Care este variabila de sistem care conține adresa ultimului OCTET DIN ZONA RAM adresabilă de procesor.

Î 12.2 Scrieți un program în cod mașină care să încarce locația de memorie 17000 cu valoarea 255. Ce observați la executarea programului? Cum puteți modifica atributele zonei de RAM-VIDEO care a fost modificată la execuția programului?

Î 12.3 Salvați un ecran:  
 a) fără zone de atribut video  
 b) cu zone de atribut

Î 12.4 Scrieți un program care să creeze în zona BORDER o succesiune de linii verzi și roșii.

**R RĂSPUNSURI**

R 12.1 Variabila este P - RAMT și este localizată pe 2 octeți la adresele 23732 și respectiv 23733.

R 12.2 10 POKE 17000,255  
 Pentru a afla octetul care conține informația de atribut a locației la adresa 17000 din VIDEO-RAM folosim următorul program:

```

10 REM modificare informație din VIDEO-RAM
20 POKE 17000,255
30 REM PROGRAM CARE URMĂREȘTE MODIFICAREA ZONEI DE
   ATRIBUTE
40 FOR i=16384+6144 TO 16384+6912
50 POKE i,
60 PRINT AT 10,10;i
70 PAUSE 0
80 REM CONTINUĂ DOAR DACĂ APĂSAM ORICE TASTĂ
90 NEXT i

```

Locația căutată are adresa 22632.

R 12.3 a) Pentru zona de VIDEO-RAM fără atribut se folosește:  
 10 SAVE "1" CODE 16384,6144  
 b) Pentru zona de VIDEO-RAM cu atribut se folosește:  
 10 SAVE "2" CODE 16384,6912

R 12.4    10 REM BORDER VERDE
 20 OUT 254,4
 30 REM BORDER ROȘU
 40 OUT 254,2
 50 GO TO 20

ANEXA A

COD	CARACTER	HEXA	MNEMONIC	DUPĂ CB	DUPĂ ED
0		00	nop	rlc b	
1		01	ld bc,NN	rlc c	
2		03	ld(bc),a	rlc d	
3	nefolosite	03	inc bc	rlc e	
4		04	inc b	rlc h	
5		05	inc b	rlc l	
6	PRINT virgulă	06	ld b,N	rlc(hl)	
7	EDIT	07	rlca	rlc a	
8	cursor stînga	08	ex af,af'	rrc b	
9	cursor dreapta	09	add hl,bc	rrc c	
10	cursor jos	0A	ld a,(bc)	rrc d	
11	cursor sus	0B	dec bc	rrc e	
12	DELETE	0C	inc c	rrc h	
13	ENTER	0D	dec c	rrc l	
14	număr	0E	ld c,N	rrc(hl)	
15	nefolosit	0F	rrca	rrc a	
16	INK control	10	djnz e	rl b	
17	PAPER control	11	ld de,NN	rl c	
18	FLASH control	12	ld(de),a	rl d	
19	BRIGHT control	13	inc de	rl e	
20	INVERSE control	14	inc d	rl h	
21	OVER control	15	dec d	rl i	
22	AT control	16	ld d,N	rl(hl)	
23	TAB control	17	rla	rl a	
24		18	jr e	rr b	
25		19	add hl,de	rr c	
26		1A	ld a,(de)	rr d	
27	nefolosite	1B	dec de	rr e	
28		1C	inc e	rr h	
29		1D	dec e	rr l	
30		1E	ld e,N	rr(hl)	
31	spatiu	1F	rre	rr a	
32		20	jr nz,e	sla b	
33	!	21	ld hl,NN	sla c	
34	"	22	ld(NN),hl	sla d	
35	#	23	inc hl	sla e	
36	\$	24	inc h	sla h	
37	%	25	dec h	sla l	
38	&	26	ld h,N	sla(hl)	
39	'	27	daa	sla a	
40	(	28	jr z,e	sra b	
41	)	29	add hl,hl	sra c	
42	*	2A	ld hl,(NN)	sra d	
43	+	2B	dec hl	sra e	
44	,	2C	inc l	sra h	
45	-	2D	dec l	sra l	
46	.	2E	ld 1,N	sra(hl)	
47	/	2F	cpl	sra a	
48	O	30	jr nc,e		
49	1	31	ld sp,NN		
50	2	32	ld (NN),a		
51	3	33	inc sp		
52	4	34	inc (hl)		
53	5	35	dec(hl)		
54	6	36	ld(hl),N		
55	7	37	scf		
56	8	38	jr c,e	srl b	
57	9	39	add hl,sp	srl c	
58	:	3A	ld a,(NN)	srl d	
59	;	3B	dec sp	srl e	
60	<	3C	inc a	srl h	
61	=	3D	dec a	srl l	
62	>	3E	ld a,N	srl (hl)	
63	?	3F	ccf	srl a	
64	@	40	ld b,b	bit 0,b	in b,(c)
65	A	41	ld b,c	bit 0,c	out(c),b
66	B	42	ld b,d	bit 0,d	sbc hl,bc
67	C	43	ld b,e	bit 0,e	ld(NN),bc
68	D	44	ld b,h	bit 0,h	neg
69	E	45	ld b,l	bit 0,l	retn
70	F	46	ld b,(hl)	bit 0(hl)	im 0
71	G	47	ld b,a	bit 0,a	ld i,a
72	H	48	ld c,b	bit 1,b	in c(c)
73	I	49	ld c,c	bit 1,c	out (c),c
74	J	4A	ld c,d	bit 1,d	adc hl,bc
75	K	4B	ld c,e	bit 1,e	ld bc(NN)
76	L	4C	ld c,h	bit 1,h	reti
77	M	4D	ld c,l	bit 1,l	
78	N	4E	ld c,(hl)	bit 1,(hl)	:
79	O	4F	ld c,a	bit 1,a	ld r,a

COD	CARACTER	HEXA	MNEMONIC	DUPĂ CB	DUPĂ ED
80	P	50	ld d,b	bit 2,b	in d,(c)
81	Q	51	ld d,c	bit 2,c	out(c),d
82	R	52	ld d,d	bit 2,d	sbc hl,de
83	S	53	ld d,e	bit 2,e	ld(NN),de
84	T	54	ld d,h	bit 2,h	
85	U	55	ld d,l	bit 2,l	
86	V	56	ld d,(hl)	bit 2,(hl)	im 1
87	W	57	ld d,a	bit 2,a	ld a,i
88	X	58	ld e,b	bit 3,b	in e,(c)
89	Y	59	ld e,c	bit 3,c	out(c),e
90	Z	5A	ld e,d	bit 3,d	abc hl,de
91	[	5B	ld e,e	bit 3,e	ld de(NN)
92	/	5C	ld e,h	bit 3,h	
93	]	5D	ld e,l	bit 3,l	
94	↓	5E	ld e,(hl)	bit 3,(hl)	im 2
95	—	5F	ld e,a	bit 3,a	ld a,r
96	£	60	ld h,b	bit 4,b	in h,(c)
97	a	61	ld h,c	bit 4,c	out(c),h
98	b	62	ld h,d	bit 4,d	sbc hl,hl
99	c	63	ld h,e	bit 4,e	ld(hl),hl
100	d	64	ld h,h	bit 4,h	
101	e	65	ld h,l	bit 4,l	
102	f	66	ld h,(hl)	bit 4,(hl)	
103	g	67	ld h,a	bit 4,a	rrd
104	h	68	ld l,b	bit 5,b	in l,(c)
105	i	69	ld l,c	bit 5,c	out(c),l
106	j	6A	ld l,d	bit 5,d	adc hl,hl
107	k	6B	ld l,e	bit 5,e	ld hl>NN)
108	l	6C	ld l,h	bit 5,h	
109	m	6D	ld l,l	bit 5,l	
110	n	6E	ld l,(hl)	bit 5,(hl)	
111	o	6F	ld l,a	bit 5,a	rld
112	p	70	ld(hl),b	bit 6,b	in f,(c)
113	q	71	ld(hl),c	bit 6,c	
114	r	72	ld(hl),d	bit 6,d	sbc hl,sp
115	s	73	ld(hl),e	bit 6,e	ld(NN),sp
116	t	74	ld(hl),h	bit 6,h	
117	u	75	ld(hl),l	bit 6,l	
118	v	76	halt	bit 6,(hl)	
119	w	77	ld(hl),a	bit 6,a	
120	x	78	ld a,b	bit 7,b	in a,(c)
121	y	79	ld a,c	bit 7,c	out (c),a
122	z	7A	ld a,d	bit 7,d	ldsp,(NN)
123		7B			
124	:	7C	ld a,h	bit 7,h	
125	-	7D	ld a,l	bit 7,l	
126	-	7E	ld a,(hl)	bit 7,(hl)	
127	©	7F	ld a,a	bit 7,a	
128	■	80	add a,b	res 0,b	
129	■	81	add a,c	res 0,c	
130	■	82	add a,d	res 0,d	
131	■	83	add a,e	res 0,e	
132	■	84	add a,h	res 0,h	
133	■	85	add a,l	res 0,l	
134	■	86	add a,(hl)	res 0,(hl)	
135	■	87	add a,a	res 0,a	
136	■	88	adc a,b	res 1,b	
137	■	89	adc a,c	res 1,c	
138	■	8A	adc a,d	res 1,d	
139	■	8B	adc a,e	res 1,e	
140	■	8C	adc a,h	res 1,h	
141	■	8D	adc a,l	res 1,l	
142	■	8E	adc a,(hl)	res 1,(hl)	
143	■	8F	adc a,a	res 1,a	
144	(a)	90	sub b	res 2,b	
145	(b)	91	sub c	res 2,c	
146	(c)	92	sub d	res 2,d	
147	(d)	93	sub e	res 2,e	
148	(e)	94	sub h	res 2,h	
149	(f)	95	sub l	res 2,l	
150	(g)	96	sub (hl)	res 2,(hl)	
151	(h)	97	sub a	res 2,a	
152	(i)	98	sbc a,b	res 3,b	
153	(j)	99	sbc a,c	res 3,c	
154	(k)	9A	sbc a,d	res 3,d	
155	(l)	9B	sbc a,e	res 3,e	
156	(m)	9C	sbc a,h	res 3,h	
157	(n)	9D	sbc a,l	res 3,l	
158	(o)	9E	sbc a,(hl)	res 3,(hl)	
159	(p)	9F	sbc a,a	res 3,a	

ANEXA A

COD	CARACTER	HEXA	MNEMONIC	DUPĂ CB	DUPĂ ED
160	(q)	A0	and b	res 4,b	ldi
161	(r)	A1	and c	res 4,c	cpi
162	(s)	A2	and d	res 4,d	ini
163	(t)	A3	and e	res 4,e	outi
164	(u)	A4	and h	res 4,h	
165	RND	A5	and l	res 4,l	
166	INKEY\$	A6	and (hl)	res 4,(hl)	
167	PI	A7	and a	res 4,a	
168	FN	A8	xor b	res 5,b	ldd
169	POINT	A9	xor c	res 5,c	cpd
170	SCREEN\$	AA	xor d	res 5,d	ind
171	ATTR	AB	xor e	res 5,e	outd
172	AT	AC	xor h	res 5,h	
173	TAB	AD	xor l	res 5,l	
174	VAL\$	AE	xor (hl)	res 5,(HL)	
175	CODE	AF	xor a	res 5,a	
176	VAL	BO	or b	res 6,b	ldir
177	LEN	B1	or c	res 6,c	cpir
178	SIN	B2	or d	res 6,d	inir
179	COS	B3	or e	res 6,e	otir
180	TAN	B4	or h	res 6,h	
181	ASN	B5	or l	res 6,l	
182	ACS	B6	or (hl)	res 6,(hl)	
183	ATN	B7	or a	res 6,a	
184	LN	B8	cp b	res 7,b	lddr
185	EXP	B9	cp c	res 7,c	cpdr
186	INT	BA	cp d	res 7,d	indr
187	SQR	BB	cp e	res 7,e	otdr
188	SGH	BC	cp h	res 7,h	
189	ABS	BD	cp l	res 7,l	
190	PEEK	BE	cp (hl)	res 7,(hl)	
191	IN	BF	cp a	res 7,a	
192	USR	CO	ret nz	set 0,b	
193	STR\$	C1	pop bc	set 0,c	
194	CHR\$	C2	jp nz,NN	set 0,d	
195	NOT	C3	jp NN	set 0,e	
196	BIN	C4	call nz,NN	set 0,h	
197	OR	C5	push bc	set 0,l	
198	AND	C6	add a,N	set 0,(hl)	
199	<=	C7	rst 0	set 0,a	
200	>=	C8	ret z	set 1,b	
201	<>	C9	ret	set 1,c	
202	LINE	CA	jp z,NN	set 1,d	
203	THEN	CB		set 1,e	
204	TO	CC	call z,NN	set 1,h	
205	STEP	CD	call NN	set 1,l	
206	DEF FN	CE	adc a,N	set 1,(hl)	
207	CAT	CF	rst 8	set 1,a	
208	FORMAT	DO	ret nc	set 2,b	

COD	CARACTER	HEXA	MNEMONIC	DUPĂ CB	DUPĂ ED
209	MOVE	D1	pop de	set 2,c	
210	ERASE	D2	jp nc,NN	set 2,d	
211	OPEN#	D3	out (N),a	set 2,e	
212	CLOSEN#	D4	call nc,NN	set 2,h	
213	MERGE	D5	push de	set 2,l	
214	VERIFY	D6	sub N	set 2,(hl)	
215	BEEP	D7	rst 16	set 2,a	
216	CIRCLE	D8	ret c	set 3,b	
217	INK	D9	exx	set 3,c	
218	PAPER	DA	jp c,NN	set 3,d	
219	FLASH	DB	IN a,(N)	set 3,e	
220	BRIGHT	DC	call c,NN	set 3,h	
221	INVERSE	DD	pref.instr. using ix	set 3,l	
222	OVER	DE	sbc a,N	set 3,(HL)	
223	OUT	DF	rst 24	set 3,a	
224	LPRINT	E0	ret po	set 4,b	
225	LLIST	E1	pop hl	set 4,c	
226	STOP	E2	jp po,NN	set 4,d	
227	READ	E3	ex (sp),hl	set 4,e	
228	DATA	E4	call po,NN	set 4,h	
229	RESTORE	E5	push hl	set 4,l	
230	NEW	E6	and N	set 4,(hl)	
231	BORDER	E7	rst 32	set 4,a	
232	CONTINUE	E8	ret pe	set 5,b	
233	DIM	E9	jp (hl)	set 5,c	
234	REM	EA	jp pe,NN	set 5,d	
235	FOR	EB	ex de,hl	set 5,e	
236	GOTO	EC	call pe,NN	set 5,h	
237	GOSUB	ED		set 5,l	
238	INPUT	EE	xor N	set 5,(hl)	
239	LOAD	EF	rst 40	set 5,a	
240	LIST	FO	ret p	set 6,b	
241	LET	F1	pop af	set 6,c	
242	PAUSE	F2	jp p,NN	set 6,d	
243	NEXT	F3	di	set 6,e	
244	POKE	F4	call p,NN	set 6,h	
245	PRINT	F5	push af	set 6,l	
246	PLOT	F6	or N	set 6,(hl)	
247	RUN	F7	rst 48	set 6,a	
248	SAVE	F8	rst m	set 7,b	
249	RANDOMIZE	F9	ld sp,hl	set 7,c	
250	IF	FA	jp m,NN	set 7,d	
251	CLS	FB	ei	set 7,e	
252	DRAW	FC	call m,NN	set 7,h	
253	CLEAR	FD	pref.instr. using iy	set 7,l	
254	RETURN	FE	cp N	set 7,(hl)	
255	COPY	FF	rst 56	set 7,a	

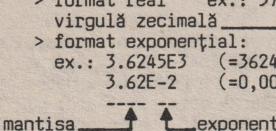
**ANEXA B**

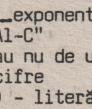
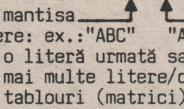
CARACTER	TASTATURĂ
ABS	(CS+SS) G
ACS	(CS+SS) SS+W
AND	SS+Y
ASN	(CS+SS) SS+Q
AT	SS+I
ATN	(CS+SS) SS+E
ATTR	(CS+SS) SS+L
BEEP	(CS+SS) SS+Z
BIN	(CS+SS) B
BLACK	0
BLUE	1
:BORDER	B
BREAK	(CS+SPACE)
BRIGHT	(CS+SS) SS+B
CAT	(CS+SS) SS+9
CHR\$	(CS+SS) U
CIRCLE	(CS+SS) SS+H
:CLEAR	X
CLOSE#	(CS+SS) CS+5
:CLS	V
CODE	(CS+SS) I
:CONTINUE	C
:COPY	Z
CYAN	S
DATA	(CS+SS) D
DEF FN	(CS+SS) SS+1
DELETE	CS+0
DIM	D
DRAW	W
EDIT	(CS+1)
ENTER	CR
ERASE	(CS+SS) SS+7
EXP	(CS+SS) X
FLASH	(CS+SS) SS+V
FN	(CS+SS) SS+2
:FOR	F
FORMAT	(CS+SS) SS+0
:GO SUB	H
:GO TO	G
GRAPHICS	(CS+9)
GREEN	4
:IF	U
IN	(CS+SS) SS+I
INK	(CS+SS) SS+X
INKEY\$	(CS+SS) N
:INPUT	I
INT	(CS+SS) R
INVERSE	(CS+SS) SS+M
INVIDEO	(CS+4)
:LEN	(CS+SS) K
:LET	L

CARACTER	TASTATURĂ
LINE	(CS+SS) SS+3
:LIST	K
LLIST	(CS+SS) V
LN	(CS+SS) Z
:LOAD	J
LPRINT	(CS+SS) C
MERGE	(CS+SS) SS+T
MOVE	(CS+SS) SS+6
:NEW	A
:NEXT	N
NOT	SS+S
:OPEN#	(CS+SS) SS+4
OR	SS+U
OUT	(CS+SS) SS+0
OVER	SS+N
:PAPER	(CS+SS) SS+C
PAUSE	M
PEEK	(CS+SS) O
PI	(CS+SS) M
:PLOT	Q
POINT	(CS+SS) SS+8
:POKE	O
:PRINT	P
:RAND	T
READ	(CS+SS) A
RED	2
:REM	E
RESTORE	(CS+SS) S
RETURN	Y
RND	(CS+SS) T
:RUN	R
:SAVE	S
SCREEN\$	(CS+SS) SS+K
SGN	F
SIN	(CS+SS) Q
SQR	(CS+SS) H
STEP	SS+D
STOP	SS+A
STR\$	(CS+SS) Y
TAB	(CS+SS) P
TAN	(CS+SS) E
THEN	SS+G
TO	SS+F
TR.VIDEO	(CS+3)
USR	(CS+SS) L
VAL	(CS+SS) J
VAL\$	(CS+SS) SS+J
VERIFY	(CS+SS) SS+R
VIDEO INVERS	(CS+4)

CARACTER	TASTATURĂ	
WHITE		7
█	(CS+9)	8
█	(CS+9)	1
█	(CS+9)	2
█	(CS+9)	3
█	(CS+9)	4
█	(CS+9)	5
█	(CS+9)	6
█	(CS+9)	7
!	(ghilimele)	SS+1
#		SS+3
\$		SS+4
%		SS+5
&		SS+6
'	(apostrof)	SS+7
(		SS+8
)		SS+9
*	(asterisc)	SS+B
+		SS+K
,	(virgula)	SS+N
-	(minus)	SS+J
.	(punct)	SS+M
/	(slash)	SS+B
:		SS+Z
;		SS+D
<		SS+R
=		SS+L
>		SS+T
?		SS+C
© (a rond)		SS+2
↖ (slash inv)		SS+Y
↓ (acolada)		SS+D
↑ (subl.)		SS+U
Ł (lira)		SS+O
↶ (acolada)		SS+F
↷ (bara vert)		SS+S
↷ (acolada)		SS+G
⌚		SS+A
≤		SS+P
≥		SS+Q
<>		SS+E
cursor sus	CS+7	SS+W
cursor dr.	CS+8	
cursor st.	CS+5	
cursor jos	CS+6	

LIMBAJUL BASIC-C (MEMENTO)

- Gama numerelor întregi: -32000.... +32000
- Gama numerelor reale :  $4 \times 10^{\pm 39} \dots 10^{\pm 38}$
- Constante numerice : > format intreg ex.: 375  
                         > format real ex.: 375,6  
                         virgulă zecimală   
                         > format exponential:  
                         ex.: 3.6245E3 (=3624,5)  
                         3.62E-2 (=0,0036)
- Constante sir caractere: ex.: "ABC" "A1-C"
- Variabile simple - o literă urmată sau nu de una sau de mai multe litere/cifre
- Variabile indexate - tablouri (matrici) - literă urmată de dimensiuni în paranteze;  
                         ex.: V(3); V(3,7)
- Variabile sir - nume variabilă urmată de \$;  
                         ex.: A\$; z1\$; A\$7; z1\$(I,K)
- Operatori
 

.aritmetici	: + - * / 
.relaționali	: =, <, >, <=, >=, <>
.logici	: AND OR NOT
.de concatenare siruri:	+ 
- Prioritatea în execuțarea operațiilor (descrescătoare):
 

↑	(ridicare la putere)
* , /	(înmulțire, împărțire)
+, -	(adunare, scădere)
=, >, <, <=, >=, <>	
- NOT
- AND
- OR

COMENZI, INSTRUCȚIUNI, FUNCȚII

- Comenzi de execuție program:
 

> CLEAR	> LIST
> CLS (CLEAR SCREEN)	> NEW
> CONT (CONTINUE)	> RUN
- Comenzi pentru lucru cu caseta magnetică:
 

> LOAD	> SAVE
> MERGE	> VERIFY
- Comenzi pentru lucru cu imprimanta:
 

> COPY (OPEN# și CLOSE#)	
> LLIST	
> LPRINT	
- Instrucțiuni de intrare-iesire:
 

> INPUT	> PRINT AT
> LET	> READ, DATA, RESTORE
> PRINT	
- Instrucțiuni de control:
 

> GO TO	> FOR, NEXT
> IF...THEN	> GO SUB, RETURN
- Instrucțiune pentru lucru cu tablouri (matrici):
 

> DIM	
-------	--
- Instrucțiune pentru comentarii:
 

> REM	
-------	--
- Instrucțiuni grafice:
 

> CIRCLE	> PLOT
> DRAW	
- Instrucțiuni de utilizare a culorilor:
 

> ATTR	> INK
> BORDER	> INVERSE
> BRIGHT	> OVER
> FLASH	> PAPER
- Instrucțiune pentru sunete:
 

> BEEP	
--------	--
- Instrucțiune de definire funcție utilizator:
 

> DEF FN	
----------	--

- Instrucțiuni de oprire execuție program:
 

> PAUSE	> STOP
---------	--------

- Funcții standard:

- |           |            |
|-----------|------------|
| > ABS     | > NOT      |
| > ACS     | > OR       |
| > AND     | > PEEK     |
| > ASM     | > PI       |
| > ATM     | > POINT    |
| > BIN     | > RND      |
| > CHR\$   | > SCREEN\$ |
| > CODE    | > SGN      |
| > COS     | > SIN      |
| > EXP     | > SQR      |
| > FN      | > STR\$    |
| > IN      | > TAN      |
| > INKEY\$ | > USR      |
| > INT     | > VAL      |
| > LEN     | > VAL\$    |
| > LN      |            |

COMENZI, FUNCȚII, INSTRUCȚIUNI (în ordine alfabetică)

- ABS număr - calculează valoarea absolută a numărului  
 ACS număr - calculează arccosinusul numărului (radiani)  
 $x\text{AND}y$  - dacă  $y=0$   $x\text{AND}y = x$   
 $y=0$   $x\text{AND}y = 0$   
 $x\$ANDy$  - dacă  $y=0$   $x\$ANDy = x\$$   
 $y=0$   $x\$ANDy = ""$  (șir nul)
- ASN număr - calculează arcsinusul numărului (radiani)  
 ATN număr - calculează arctangentă numărului (radiani)  
 ATTR (linie,coloană) - dă informații despre atributele de culoare ale unui caracter  
 BEEP x,y - scoate o notă muzicală de înălțime y cu o durată de x secunde  
 BIN biți - indică un număr binar  
 BORDER n - colorează marginea imaginii cu culoarea dată de valoarea lui n ( $n=0$  la 7)  
 BRIGHT n - produce afișarea caracterelor: normal ( $n=0$ ), strălucitor ( $n=1$ )  
 CHR\$ - dă caracterul al cărui cod este x, sub formă de sir ("caracter")  
 CIRCLE x,y,r - desenează un cerc de rază "r" cu centrul în punctul de coordonate "x,y"  
 CLEAR n - șterge din memorie toate variabilele introduse anterior prin program și modifică variabila de sistem RAMTOP la adresa n  
 CLS - șterge ecranul și anulează conținutul memoriei ecran, fără a influenta variabilele stabilite prin program  
 CODE x\$ - dă codul primului caracter din sirul x  
 CONT - (CONTINUE) continuă un program opriț temporar prin STOP  
 COPY - copiază primele 22 de linii ale ecranului pe imprimantă  
 COS x - calculează cosinusul unghiului x (x în radiani)  
 DATA cl, c2,... - introduce valori (constante numerice / sir de caractere) pentru variabilele folosite în program  
 $\text{Ex.: } 135 \text{ DATA } 3.2, "ABC", 43$   
 DEF FN n(V1,V2...)=e - definește o funcție nestandard, a utilizatorului  
 DEF FN n\$(V1\$, V2...)=e  
     .n este un număr dat printr-o literă  
     .V1, V2... sint numere de variabile constituind parametrii funcției (maxim 26)  
     .e este o expresie matematică  
 $\text{Ex.: } 10 \text{ DEF FN p(x,y)=x}^{1/2} \cdot 3$   
 $70 \text{ LET A=FN p(3,7)}$   
 DIM a(n1,n2...) - declară tablouri sau (matrici) de numere sau de siruri  
 DIM a\$(n1,n2...)  
     .a/a\$ este numele tabloului  
     .n1,n2... sint dimensiunile tabloului  
 $\text{Ex.: } \text{DIM b}(14) \text{ DIM m}(3,15)$   
 $\text{DIM c}$(7)  $\text{DIM g}$(7,15)  
 DRAW x,y,z - desenează o linie de la poziția curentă a cursorului grafic pînă la punctul de coordonate x și y pixeli  
 $-255 \leq x \leq 255 \quad -175 \leq y \leq 175$   
     .z este unghiul (în radiani) pentru descrierea$$

## ANEXA C

unei linii circulare  
 Ex.: DRAW 255,0 DRAW x-45, y-80  
 DRAW 60,60,PI

EXP x - calculează funcția exponentială la puterea x  
 FLASH n - produce afișarea normală (n=0) sau clipitoare (n=1)

FOR x = n TO m - execută secvența de instrucțiuni care urmează

FOR x = n TO m STEP y (până la NEXT x) de un număr de ori dat de relația:  $(m-n)/(y-1)$   
 .x este numele variabilei (o literă)  
 .n, m și y sunt numere sau expresii aritmetice

Ex.: 30 FOR i=1 TO 12 STEP 5  
 40 FOR j=14 TO 3 STEP -2  
 :  
 :  
 80 NEXT j:NEXT i

GO SUB nr.linie - produce saltul în program la subrutina care începe cu linia indicată și se termină cu prima instrucțiune RETURN  
 După executarea subruteinei programul se reia cu instrucțiunile după GOSUB  
 Ex.: 60 GO SUB 9000

GO TO n - produce salt necondiționat la linia n  
 Ex.: IF x>0 THEN GOTO 10

IF x THEN s - dacă expresia x este adeverată (diferită de zero) se execută s în caz contrar se execută instrucțiunea care urmează după IF  
 Ex.: IF a\$="SF" AND b=0 PRINT b  
 IF T=0 THEN GOTO 100

IN m - citește din memorie octetul din portul cu adresa m  
 INK n - determină culoarea cu care vor fi afișate caracterele ce urmează (n=0 la 7)

INKEY\$ - introduce caracterul primit de la tastatură  
 Ex.: LET a\$=INKEY\$: IF INKEY\$="N" THEN GO TO 80  
 INPUT "c",v1,v2... - permite introducerea datelor de la tastatură sau

INPUT LINE...  
 .c este un sir de caractere optionale care explicită datele ce vor fi introduse  
 .v1,v2... sunt nume de variabile numerice sau sir de caractere  
 Ex.: INPUT R  
 INPUT "INTRODUCETI PRETUL";p  
 INPUT n\$  
 INPUT LINE a\$

INT n - dă o valoare întreagă a numărului n (<=n)

INVERSE n - controlează inversa afișării caracterelor ce urmează  
 .dacă n=0 caracterele sunt afișate video normal  
 .dacă n=1 caracterele sunt afișate video invers

LEN sir - dă numărul de caractere conținute în sir

LET v=e - atribuie variabilei v valoarea expresiei e  
 Ex.: LET i=i+1  
 LET a=32  
 LET NS="NUME"  
 LET SUMA=4.25\*A 2

LIST - listează programul, începând cu prima linie

LIST n - listează programul începând cu prima linie al cărui număr >=; linia n devine linie curentă

LLIST/LLIST n - listează programul pe imprimantă

LN număr - calculează logaritmul natural al numărului

LOAD "nume" - încarcă în memorie programul cu numele specificat LOAD "nume" CODE adresa, număr  
 - încarcă un număr de octeți începând de la adresa specificată

LPRINT listă - scrie la imprimantă

MERGE "nume" - încarcă în memorie un program de pe casetă, dar nu șterge în întregime pe cel vechi, ci înlocuiește numai liniile cu același număr

NEW - șterge din memorie programul și variabilele

NEXT v - produce reluarea execuției unui ciclu FOR pentru valoarea următoare a variabilei v  
 Ex.: FOR k=1 TO 10 STEP 2  
 :  
 :  
 NEXT k'

a OR b - dacă b=0 a OR b = a  
 b>0 a OR b = 1

OUT m,n - inscrie valoarea octetului n în portul de intrare/iesire cu adresa m

OVER n - dacă n=1, producă supratipărirea unui nou caracter deasupra celui vechi, în poziția în care se află prompter-ul de linie

PAPER n - colorează fondul (hîrtia) ecranului cu culoarea dată de n (n=0 la 7)  
 Ex.: BORDER 5:PAPER 7  
 LET bw=2:PAPER bw

PAUSE n - oprește execuția programului pe o perioadă de n/50 secunde (n=0-65535), ecranul rămînind neschimbat. Cind se dă PAUSE 0 reluarea programului se face la apăsarea oricărei taste.

PEEK adresa - citește din memorie și afișează octetul de la "adresa"  
 Ex.: PEEK 23627  
 FOR a=0 TO 20  
 PRINT a;TAB 10;PEEK a  
 NEXT a

PI - introduce valoarea 3,141592265

PLOT x,y - mută cursorul grafic în poziția de coordonate x,y (x=0-255 și y=0-175) și desennează un punct.

PLOT m;x,y - m reprezintă INVERSE/OVER  
 Ex.: PLOT INVERSE 1,30,40  
 PLOT OVER 1,100,100

POKE m,n - scrie valoarea n în octetul cu adresa m din memorie (m=0 la 65535, n=0 la 255)  
 Ex.: POKE 23692,255  
 POKE USR "M" + i

POINT (x,y) - dă informații despre punctul de coordonate (x,y);  
 .valoarea 1 - dacă punctul este vizibil (culoarea cernelii)  
 .valoarea 0 - dacă punctul este stins (culoarea hîrtiei)

PRINT... - afișează pe ecran constante, variabile sau expresii, separate în instrucțiune prin virgulă, punct și virgulă sau apostrof.

PRINT AT 1,c,... - afișează începând cu poziția dată de coordonatele 1 și c (linia=0-21 și coloana=0-31)  
 Ex.: PRINT AT 11,13; "CENTRUL"  
 PRINT A\$(x TO 6), A\$(7-n) to 6

PRINT TAB C,... - afișează începând cu coloana C (C=0-31).  
 Ex.: PRINT AT 3,1;"Capitol"; TAB24;"Pagina"

RND - dă un număr pseudoealeator cuprins între 0 și 1 sau cuprins în orice alt domeniu de valori astfel:  
 xxRND numere între 0 și 8  
 a+xxAND numere între a și x  
 Ex.: 8xRND numere între 0 și 8  
 2.1+0.9\*RND numere între 2.1 și 3  
 5+INT(RND\*20) numere întregi între 5 și 20

RANDOMIZE n - generează o valoare n de start pentru instrucțiunea RND (n=1-65535)

RANDOMIZE sau RANDOMIZE 0 - dă o valoare de start în funcție de timpul trecut de la pornirea calculatorului.  
 Ex.: RANDOMIZE 1  
 FOR n=1 TO 10:PRINT RND  
 NEXT n

READ v1,v2... - atribuie variabilelor v1,v2... valoarea unor expresii succesive din listele instrucțiunilor DATA.  
 Ex.: READ A(x)  
 DATA 7,35,42,83  
 DATA "ART1","ART2"

REM comentariu - permite introducerea de comentarii în program  
 Ex.: REM se execută desenul

RESTORE n - restabilește indicatorul de citire în lista de instrucțiuni DATA cu număr de linie n, astfel incit următoarea instrucțiune READ va începe citirea de la această linie.  
 Ex.: 40 RESTORE 100  
 50 READ a,b  
 60 DATA 40,65,82,96  
 :  
 100 DATA 26,7,"ABC",-52,75

RETURN - instrucțiune obligatorie de încheiere a unei subrute; provoacă intoarcerea execuției programului la prima instrucțiune după GOSUB n.

RUN - lansează în execuție programul începând cu prima linie.

## ANEXA C

## ANEXA D

RUN n - lansează în execuție începînd cu linia n.  
 SAVE "nume" - copiază programul din memorie pe casetă, cu numele indicat.  
 SAVE "nume" LINE n - înregistrează programul pe casetă astfel încît la reîncărcarea lui în memorie va fi lansat automat în execuție începînd cu linia n.  
 SAVE "nume" DATA nume matrice - înregistrează pe casetă un tablou.  
 SAVE "nume" CODE adresă, număr - copiază pe bandă un număr de octeți din memorie începînd de la adresa specificată.  
 SCREEN\$ (x,y) - dă caracterul ASCII aflat în poziție determinantă de coordonatele x (linia) și y (coloana).  
 Ex.: PRINT SCREEN\$ (10,18)  
 SGN x - dă semnul lui x:  
 -1 dacă x<0  
 0 dacă x=0  
 +1 dacă x>0  
 SIN x - calculează sinusul unghiului x (în radiani)  
 Ex.: SIN (n/6\*xPI)  
 SQR x - calculează radical de ordin doi din x (x > 0)  
 Ex.: SQR 4  
 SQR 0.625  
 SQR (n/64)  
 STR\$ x - convertește argumentul x într-un sir  
 Ex.: LET n\$=STR\$ 2E3  
 STOP - oprește execuția programului afișîndu-se mesajul:  
 STOP statement, n:m  
 Execuția programului se reia cu CONTINUE  
 TAN x - calculează tangenta unghiului x (x în radiani)  
 USR adresa - lansează în execuție o subrutină scrisă în cod mașină memorată începînd cu adresa specificată.  
 Ex.: INPUT "rînd"  
 POKE USR "a"+n,rînd  
 URS x\$ - x este o literă între a și u, sau un caracter grafic definit de utilizator. Se obține adresa primului octet pentru caracterul grafic respectiv.  
 VAL "x" - evaluează sirul x ca o expresie numerică.  
 Ex.: VAL "3.5"=15  
 VAL\$"x" - evaluează sirul x ca o expresie sir.  
 VERIFY "nume" - verifică dacă înregistrarea de pe banda magnetică are același conținut cu memoria.

## VARIABILE DE SISTEM

Octeții din memorie de la adresa 23552 la adresa 23733 sunt rezervați pentru operații specifice ale sistemului. Ei pot fi citiți pentru a afla diferite lucruri despre sistem, iar cîteva din ei pot fi și modificați. Acești octeți se numesc variabile de sistem și au cîte un nume, dar nu trebuie confundați cu variabilele utilizate de BASIC. În cazul variabilelor formate din mai mulți octeți primul va fi cel mai puțin semnificativ. Variabilele de sistem sunt date în lista de mai jos. Abrevierile din coloana 1 au următoarea semnificație:

X - această variabilă nu poate fi modificată deoarece sistemul va funcționa eronat  
 N - modificarea acestei variabile nu are un efect asupra funcționării normale a sistemului  
 n - numărul de octeți din variabilă

Tip	Adresă	Nume	Conținut
	---	---	-----
N8	23552	KSTATE	Folosită în citirea tastaturii
N1	23560	LAST K	Reține ultima tastă apăsată
1	23561	REPDEL	Durata (în 1/50 sec.) cît trebuie să fie apăsată tasta pentru a se repeta, inițial 35
1	23562	REPPER	Timpul (în 1/50 sec.) după care se repetă o tastă apăsată, inițial 5
N2	23563	DEFADD	Adresa argumentelor funcțiilor definite de utilizator dacă una dintre ele a fost evaluată
N1	23565	K DATA	Al doilea octet pentru controlul cularor introdus de la tastatură
N2	23566	TVDATA	Controlul cularor, al lui AT și TAB pentru TV
X38	23568	STRMS	Adresa canalului atașat căii
2	23606	CHARS	Adresa generatorului de caractere minus 256 în mod normal în BASIC-S, dar vă puteți genera unul și dvs. făcînd ca CHAR\$ să pointeze la el
1	23608	RASP	Durata sunetului de atenționare
1	23609	PIP	Durata sunetului la apăsarea unei taste
1	23610	ERR NR	Codul de mesaj minus 1
X1	23611	FLAGS	Diferiți indicatori de control ai sistemului BASIC
X1	23612	TVFLAG	Indicatorii asociați cu TV-ul
X2	23613	ERR SP	Adresa elementului din stiva mașinii utilizat ca adresă de introducere în caz de eroare
N2	23615	LIST SP	Adresa de întoarcere la listările automate
N1	23617	MODE	Specifică cursorul (K,L,C,E,G)
2	23618	NEWPPC	Linia la care se sare
1	23620	NSPPC	Numărul instrucțiunii în linie la care se sare
2	23621	PPC	Numărul liniei pentru instrucțiunea în execuție
1	23623	SUBPPC	Numărul instrucțiunii din linie în execuție
1	23624	BORDER	Culoarea border-ului înmulțită cu 8; conține de asemenea atributele folosite pentru partea de jos a ecranului
2	23625	E PPC	Numărul liniei curente
X2	23627	VARS	Adresa variabilelor
N2	23629	DEST	Adresa variabilelor asignate
X2	23631	CHANS	Adresa datelor de canal
X2	23633	CURCHL	Adresa informației curente folosită pentru intrare sau ieșire
X2	23635	PROG	Adresa programului BASIC
X2	23637	NXTLIN	Adresa următoarei linii de program
X2	23639	DATA DD	Adresa ultimului element din lista DATA
X2	23641	E LINE	Adresa comenzi introduse
2	23643	K CUR	Adresa cursorului
X2	23645	CH ADD	Adresa următorului caracter care urmează să fie interpretat
2	23647	XPTR	Adresa caracterului după semnul întrebării
X2	23649	WORKSP	Adresa spațiului de lucru temporar
X2	23651	STKBOT	Adresa inferioară a stivei calculator

**ANEXA D**

X2	23653	STKEND	Adresa de inceput a spatiului liber
N1	23655	BREG	Registrul B al calculatorului
N2	23656	MEM	Adresa spatiului folosit pentru memoria calculatorului
1	23658	FLAGS2	Alți indicatori
X1	23659	DF SZ	Numărul liniilor din partea de jos a ecranului
2	23660	S TOP	Numărul liniei de sus a programului la listare automată
2	23662	OLOPPC	Numărul liniei la care sare CONTINUE
1	23664	OSPPC	Numărul din linie la care sare CONTINUE
N1	23665	FLAGX	Diverși indicatori
N2	23666	STRLEN	Lungimea asigurată sirului
N2	23688	T ADDR	Adresa următorului element din tabela sintaxă
2	23670	SEED	Variabila pentru RND
3	23672	FRANCES	Controlul de cadre pe 3 octeți, cel mai puțin semnificativ primul, incrementat la 20 ms
2	23675	UDG	Adresa primului grafic definit de utilizator
1	23677	COORDS	Coordonata x a ultimului punct plot-at
1	23678		Coordonata y a ultimului punct plot-at
1	23679	P POSN	Numărul poziției de scriere pe ecran
1	23680	PR CC	Octetul mai puțin semnificativ al adresei pentru noua poziție la care se imprimă prin LPRINT
1	23681		Nefolosit
2	23682	ECHO E	Numărul coloanei și al liniei
2	23684	DF CC	Adresa de afișare pe ecran prin PRINT
2	23686	DFCCL	Același lucru pentru partea de jos a ecranului
X1	23688	S POSN	Numărul coloanei pentru PRINT
X1	23689		Numărul liniei pentru PRINT
X2	23690	SPOSNL	Ca S POSN pentru partea de jos a ecranului
1	23692	SCR CT	Numără defilările pe ecran
1	23693	ATTR P	Culoarea curentă
1	23694	MASK P	Folosit pentru culori transparente
N1	23695	ATTR T	Culori temporare
N1	23696	MASK T	Ca MASK P dar temporar
1	23697	PFLAG	Alți indicatori
N30	23698	MEMBOT	Arie memorie calculator
2	23728		Nefolosit
2	23730	RAMTOP	Adresa ultimului octet din aria sistemului BASIC
2	23732	P-RAMT	Adresa ultimului octet de RAM

## C U P R I N S

Introducere .....	pag. 1	Capitolul 9 .....	pag. 36
-Date tehnice		-Să desenăm cu CIP-ul	
-Inventar de licrare		-Aplicație practică AP9	
-Garanție		-INK, PAPER	
-Despre calculatoare		-BORDER	
Instalare și punere în funcțiune.....	pag. 2	-PLOT	
-Alimentare		-DRAW	
-Lista semnalelor prezente la conectorul de extensie		-CIRCLE	
Capitolul 1 .....	pag. 3	-RND	
-Structura CIP-ului		-RAND	
Capitolul 2 .....	pag. 5	-Caractere grafice standard	
-Aplicație practică AP1		-Caractere grafice speciale	
-Literă mari/Litere mici		-Animarea pe ecran	
-Videonormal/Videoinvers		Capitolul 10 .....	pag. 42
Capitolul 3 .....	pag. 7	-Să introducem sunete	
-Cum comunicăm cu CIP-ul		-Aplicație practică AP10	
-Programe în limbaj BASIC		-Codificarea notelor	
-Aplicație practică AP3		-Schimbarea octavei	
Capitolul 4 .....	pag. 12	-Schimbarea duratei	
-Operării, operatori, constante, variabile, expresii		-Schimbarea ritmului	
-Aplicație practică AP4		Capitolul 11 .....	pag. 44
-Operatori		-Funcții	
-Constante numerice		-Aplicație practică AP8	
-Constante sir de caractere		-SIN, COS, TAN, ASN, ACS, ATN	
-Variabile numerice		-PI	
-Variabile sir de caractere		-ABS	
-Notatia exponentială a numerelor		-SGN	
-Operatori relaționali		-INT	
-Operatori logici		-SQR	
-Alipirea sirurilor		-EXP	
Capitolul 5 .....	pag. 18	-LN	
-Intrări, ieșiri, aplicații simple		-BIN xxxxxxxx	
-LET		-POINT	
-INPUT		-LEN	
-READ		-STR\$	
-DATA		-VAL	
-RESTORE		-VAL\$	
-PRINT		-CHR\$	
-PRINT AT		-CODE	
-Macheta ecranului		-JNKEY\$	
-PRINT TAB		-SCREEN\$	
-REM		-DEF FN	
-Salvarea programelor pe casetă		-FN	
Capitolul 6 .....	pag. 25	Capitolul 12 .....	pag. 47
-Condiții sau alternative - IF ... THEN; GO TO		-Memoria CIP-ului	
-GO TO		-Aplicație practică AP12	
-IF ... THEN		-RAM VIDEO	
Capitolul 7 .....	pag. 27	-Variabile de sistem	
-Cum se pot repeta părți din program		-CLEAR	
-Aplicație practică AP7		-NEW	
-FOR ... NEXT		-LINIE BASIC	
Capitolul 8 .....	pag. 30	-Cod mașină	
-Colecții de date		-RANDOMISE USR	
-DIM, GO SUB, RETURN		-SAVE CODE	
-Aplicație practică AP8		-LOAD CODE	
-LISTA		-OUT	
-MATRICE		-IN	
-DIM		Anexa A .....	pag. 50
-Subroutine		Anexa B .....	pag. 52
-GO SUB		Anexa C .....	pag. 53
-RETURN		Anexa D .....	pag. 55





