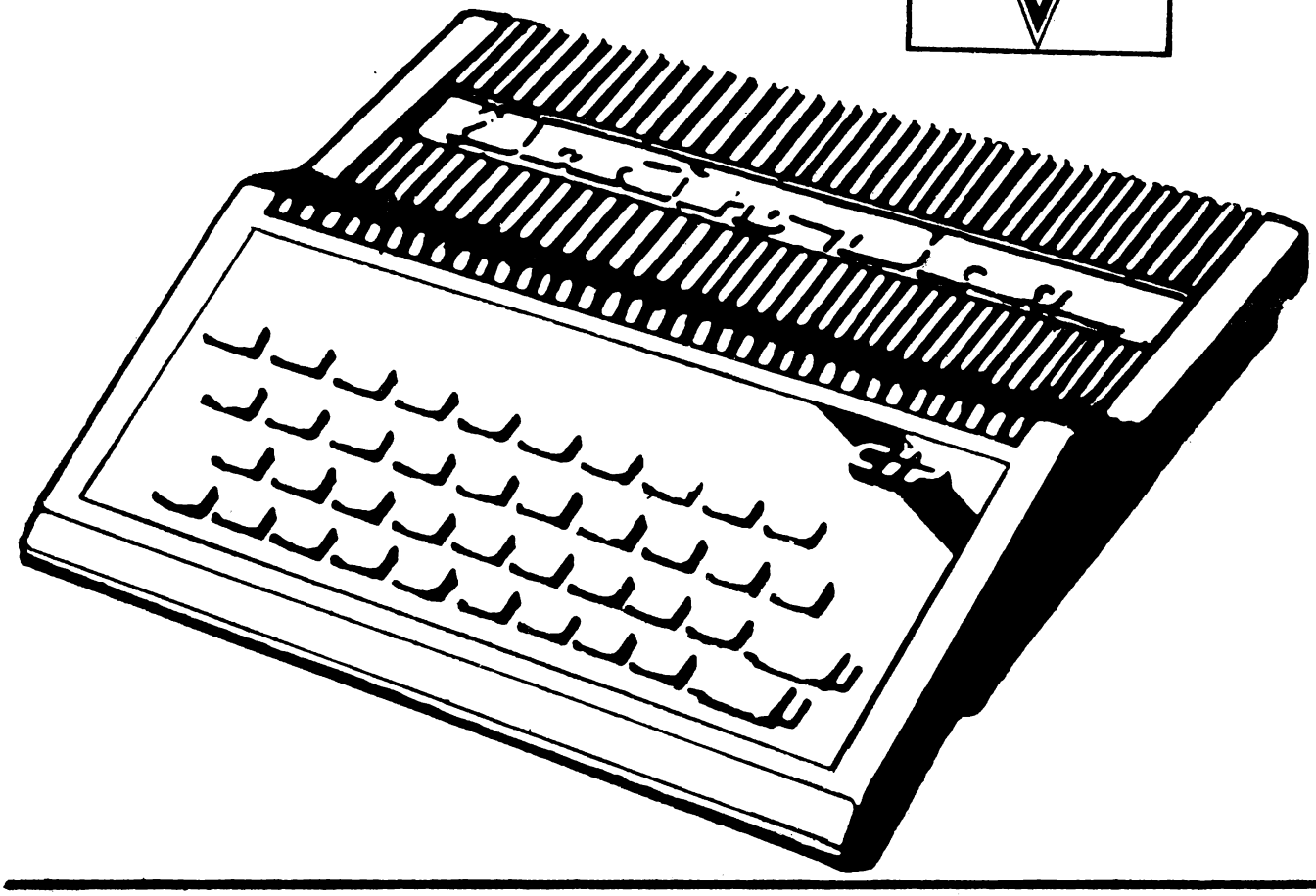
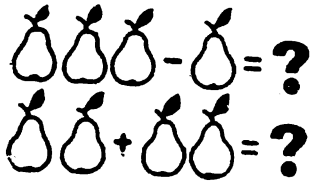
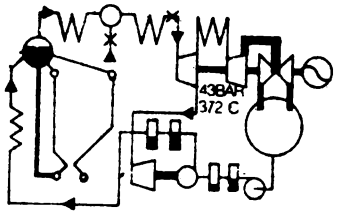
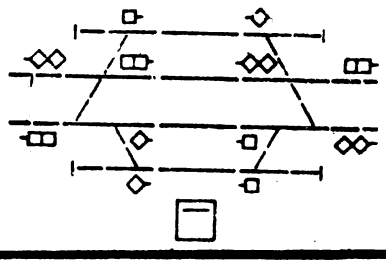
CALCULATOR PENTRU INSTRUIRE 03









manual de utilizare

STIMATE CUMPĂRĂTOR,

Felicitări pentru că v-aţi hotărît să folosiţi această jucărie care, vă va oferi clipe de plăcere şi satisfacţie.
CIP-ul a fost conceput şi vă este oferit ca un sprijin de nădejde în activitatea de instruire şcolară, în aplicaţii practice din activitatea profesională, în industrie, în medicină, în cercetare, în calcule economice, etc..

DATE TEHNICE

Microprocesor	Z 80 A (MMN 80 CPU)
Memorie EPROM	16 Ko
Memorie RAM	64 Ko
Tastatură	QWERTY cu 40 taste
Afişare	TV. alb-negru/color PAL (canalul 8 OIRT)
	monitor alb-negru/color
	8 culori cu două trepte de luminozitate
Rezoluţie	192 x 256 în regim grafic
	24 x 32 în regim alfanumeric
Limbaj	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
- Sursă de alimentare
- Cablu de conectare la TV
- Manual de utilizare
- Reţeaua unităţilor service
- Certificat de garanţie

CONDIŢII DE GARANŢIE

CIP-ul este garantat de ELECTRONICA S.A. 12 luni de la data cumpărării. Ies din garanţie aparatele la care a fost violat sigiliul. Intreţinerea aparatelor în perioada de garanţie se asigură de către S.A. ELECTRONICA - SERVICE prin unităţile de lucru din întreaga ţară nominalizate în "Reţeaua unităţilor service" - AX 1443.

DE CE SE NUMEŞTE CIP ?

Aveţi în faţă un Calculator pentru Instruire Personală programabil denumit mai pe scurt şi mai nostim "CIP", conceput pentru a fi folosit în procesul de învăţămînt elementar, mediu şi superior.
În învăţămîntul preşcolar pe CIP se pot prezenta noţiuni elementare de grafică, muzică, desene animate etc.
În învăţămîntul liceal poate fi folosit eficient în instruire asistată, pentru însuşirea materiilor.
În învăţămîntul superior CIP-ul poate reduce timpul necesar muncii de rutină în realizarea proiectelor.
CIP-ul îl puteţi utiliza pentru însuşirea unor limbaje evoluate de programare a calculatoarelor.
CIP-ul este capabil de multe alte utilizări. Cu el puteţi juca pe ecranul unui televizor: şah, GO, SCRABBLE, fotbal, baschet, puteţi învăţa să conduceţi maşina, puteţi desena în culori pe ecranul televizorului color, puteţi compune muzică etc., CIP-ul dovedindu-se un partener inteligent, răbdător şi perspicace.
Dar pentru unii dintre dumneavoastră, poate mai interesant decît folosirea programelor făcute de alţii va fi realizarea unor aplicaţii proprii, în care imaginaţia şi creativitatea de care veţi da dovadă vă vor face să fiţi foarte încîntaţi, iar CIP-ul se va dovedi colaboratorul inteligent, bun la toate, cu condiţia să fie învăţat ce are de făcut. Deşi poate uneori veţi fi uimit de ce este capabil, reţineţi că tot ce ştie el sînt creaţii umane, CIP-ul fiind numai un calculator personal familial.

CE FEL DE CALCULATOARE PERSONALE EXISTĂ ?

Calculatoarele personale sînt considerate ca fiind de două mari categorii:
- calculatoare personale familiale (HOME COMPUTER), cu un preţ accesibil pentru a putea fi cumpărat "pentru acasă" şi avînd destule posibilităţi de utilizare pentru calcule, desene, instruire, jocuri, etc.;
- calculatoare personale profesionale (PERSONAL COMPUTER) cu posibilităţi foarte mari de utilizare în instituţii sau întreprinderi pentru aplicaţii de cercetare-proiectare, activităţi de biroú, instruire asistată de calculator etc.

CE AVEŢI ÎN FAŢĂ ?

Obiectul minunat cu care doriţi să lucraţi, face parte din categoria calculatoarelor familiale şi veţi constata cu plăcere cît de mari îi sînt posibilităţile de utilizare.

CE PROGRAME POT FI FOLOSITE ?

CIP-ul a fost conceput să înţeleagă şi să execute programe făcute de firme renumite pentru calculatoarele de largă răspîndire SINCLAIR - SPECTRUM. De asemenea, CIP-ul acceptă programe elaborate pe calculatoare româneşti HC-85 şi TIM-S. Reciproc, programele realizate pe CIP vor putea fi executate pe calculatoare SINCLAIR ZX SPECTRUM sau pe calculatoarele româneşti HC-85 şi TIM-S.

CUM SE FOLOSEŞTE ACEST MANUAL ?

Acest manual vă prezintă, simplu şi practic, cunoştinţele necesare pentru a înţelege ce cuprinde un calculator, cum funcţionează şi cum puteţi lucra curent cu el.
Pentru a înţelege fără greutăţi deosebite ce vi se prezintă în continuare, este necesar ca parcurgerea materialului să o faceţi simultan cu folosirea CIP-ului.
Veţi avea foarte multe exemple, nu ezitaţi să le verificaţi pe calculator. Veţi constata că limbajul utilizat este foarte exigent şi prietenos în acelaşi timp, CIP-ul semnalîndu-vă orice eroare, neexecutînd decît comenzile corecte.

CIP-ul VĂ AJUTĂ SĂ VĂ CUNOAŞTEŢI !

De fapt calculatorul este o oglindă a intelectului dumneavoastră, executînd cele ce-i veţi da corect şi clar, dar sancţionînd şi orice lipsă de atenţie, de logică sau de cunoaştere.
Manualul se adresează tuturor, fără a cere o pregătire prealabilă în electronică sau informatică. El nu vă arată cum se construieşte un calculator ci din ce este format, cum lucrează şi cum îl puteţi utiliza pentru diverse aplicaţii.

Fiecare capitol este structurat astfel:
- aplicaţie practică;
- noţiuni de bază;
- exemple de programe;
- întrebări recapitulative şi exerciţii;
- probleme propuse pentru rezolvare;
- răspunsuri.

Acolo unde în text aveţi lăsat spaţiu liber subliniat, vă solicităm să introduceţi un răspuns al dumneavoastră bazat pe noţiuni explicate anterior.
Fiind un manual de autoinstruire, nimic nu vă forţează să-l studiaţi contra cronometru. Apreciaţi posibilităţile dumneavoastră de asimilare, care pot varia de la o zi la alta. Vă indicăm să nu studiaţi mai puţin de o oră, dar nici mai mult de trei-patru ore pe zi.
Materialul este astfel redactat încît să asimilaţi cunoştinţele noi pas cu pas, cu minimum de efort, încît veţi fi plăcut impresionat de posibilităţile dumneavoastră.

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

INSTALARE ŞI PUNERE ÎN FUNCŢIUNE

▸ In cazul în care aparatul a fost depozitat în condiţii de umiditate excesivă, sau diferenţa dintre temperatura aerului în care a fost depozitat şi cea a camerei în care este instalat este mare, trebuie să treacă minimum 3 ore pînă la prima punere în funcţiune;
▸ Nu introduceţi sursa de alimentare în spaţii închise în timpul funcţionării;
▸ Nu astupaţi orificiile de aerisire ale sursei de alimentare şi ale CIP-ului în timpul funcţionării;
▸ Nu instalaţi CIP-ul şi sursa de alimentare lîngă vase cu lichid întrucît 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 porniţi decît după cel puţin 12 ore, eventual chemaţi service-ul;
▸ Înainte de curăţarea exterioară, aparatul va fi deconectat de la reţea. Aparatul se va curăţa cu o ţesătură moale uşor umezită într-o soluţie de săpun. Nu se vor utiliza diluanţi puternici sau benzină, deoarece atacă finisajul.

ATENŢIE! ▸ NU LĂSAŢI CIP-ul SĂ FUNCŢIONEZE NESUPRAVEGHEAT ▸ NU FOLOSIŢI CIP-ul ÎN ÎNCĂPERI CU UMIDITATE EXCESIVĂ ▸ NU PERMITEŢI COPIILOR SĂ INTRODUCĂ MONEZI SAU ALTE OBIECTE METALICE ÎN APARAT, PRIN FANTELE DE VENTILAŢIE ▸ NEGLIJAREA ACESTOR REGULI POATE DUCE LA SUPRAÎNCĂLZIREA APARATULUI SAU CHIAR LA INCENDIEREA LUI.

SURSA DE ALIMENTARE

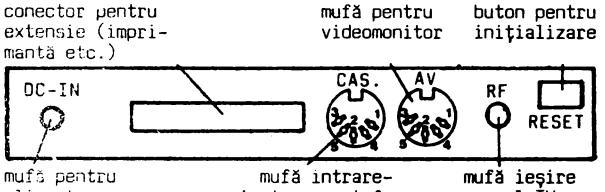
CIP-ul se alimentează prin sursa de alimentare la tensiunea de 5 V cc. Sursa de alimentare a CIP-ului se alimentează de la reţeaua de curent alternativ de 220 Vef. În momentul conectării la reţea, asiguraţi-vă că priza este bine fixată în doză, iar ştecherul intră fix în priză.

STRICT INTERZIS !

▸ Să folosiţi cordoane de reţea cu izolaţia compromisă.
▸ Să introduceţi două tripluştechere în aceeaşi priză.

▸ Să înlocuiţi siguranţele cu alte valori decît cele prevăzute.
▸ Să folosiţi sursa de alimentare cu capacul desfăcut, întrucît riscaţi să vă electrocutaţi.

Alegeţi un loc de lucru care să vă permită un spaţiu confortabil de aşezare pentru CIP, televizor (la o distanţă corespunzătoare) şi casetofon.
Instalarea este extrem de simplă. Vă trebuie o priză triplă şi ... puţină atenţie. Priviţi din spate CIP-ul şi veţi observa următoarele mufe/conectori:



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

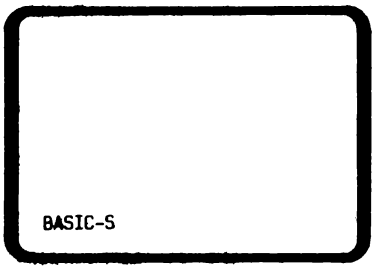
Mufă AV
5 - ieşire video
4 - ieşire audio
2 - masă
3 - ieşire +5 Vcc.
1 - neconectat

EFECTUAŢI URMĂTOARELE OPERAŢII:

▸ introduceţi fişa alimentatorului în mufa DC-IN a CIP-ului;

ATENŢIE ! FACEŢI ÎNTOTDEAUNA ACEASTĂ OPERAŢIE CU ŞTECHERUL ALIMENTATORULUI SCOS DIN PRIZĂ, ALTFEL RISCAŢI SĂ SE ARDĂ SIGURANŢA ALIMENTATORULUI.

▸ introduceţi cablul de casetofon în mufa notată CAS a CIP-ului;
▸ introduceţi cablul TV în borna de antenă a televizorului şi în mufa RF a CIP-ului;
▸ porniţi televizorul;
▸ introduceţi ştecherul alimentatorului în priză (220 V);
▸ acordaţi televizorul pe canalele 6-12 pînă obţineţi o imagine stabilă cu următorul text: BASIC-S.



DACĂ NU APARE IMAGINEA DESCRISĂ MAI SUS VERIFICAŢI:

▸ dacă aţi montat corect cablurile şi dacă mufele sînt bine împinse în contacte;
▸ încercaţi acordarea televizorului pe canalul necesar;
▸ acţionaţi butonul RESET din spate-stînga. Acest buton provoacă iniţializarea tuturor circuitelor fără a fi oprită alimentarea electrică.
▸ scoateţi din priză şi reintroduceţi ştecherul alimentatorului.

Dacă nici una din acţiunile de mai sus nu au avut ca rezultat obţinerea mesajului corect, există ceva defect în configuraţia CIP-ului şi veţi proceda conform condiţiilor de garanţie ale produsului.

LISTA SEMNALELOR PREZENTE LA CONECTORUL DE EXTENSIE

Faţa A	A15	A13	D7	NC	SLOT	D0	D1	D2	D6	D5	
Nr.	1	2	3	4	5	6	7	8	9	10	
Faţa B	A14	A12	VCC(+5V)	NC	SLOT	GND	GND	CLK(3,5MHz)	A0	A1	
Faţa A	D3	D4	INT	NMI	HALT	MREQ	IORQ	RD	WR	NC	WAIT
Nr.	11	12	13	14	15	16	17	18	19	20	21
Faţa B	A2	A3	IORGE	GND	VIDEO	NC	NC	NC	BUSRQ	RESET	A7

Faţa A	NC	NC	M1	RFSH	A8	A10	VINH
Nr.	22	23	24	25	26	27	28
Faţa B	A6	A5	A4	ROMINH	BUSACK	A9	A11

OBSERVAŢII:
1. Faţa A este faţa plantată.
2. Semnalele sînt numerotate privind dinspre faţa A, de la stînga la dreapta, ţinînd calculatorul în poziţie normală de lucru.
3. Asignarea semnalelor la conectorul de extensie este identică cu cea de la calculatorul SINCLAIR SPECTRUM, exceptînd tensiunile de -5 V, +12 V, -12 V şi 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 operaţii de "intrare" se introduc:
- programul de prelucrare, prin care CIP-ul este instruit cum să rezolve problema;
- datele problemei, adică valori pentru A, B şi C.

Ce este programul de prelucrare ?

Programul cuprinde o mulţime de instrucţiuni (comenzi) scrise într-un limbaj pe care-l înţelege calculatorul şi ordinea în care vor fi executate.
Instrucţiunile programului corespund unor paşi elementari în care se rezolvă problema.
În exemplul dat, instrucţiunile pot fi comentate astfel:

1 - cere / citeşte valori pentru B şi C
2 - calculează (B - C)
3 - cere / citeşte valoarea lui A
4 - calculează D = A - (B - C)
5 - afişează / scrie (ecran / hîrtie) valoarea lui D
6 - mergi la punctul 1 (pentru a repeta secvenţa cu alte valori pentru A, B şi C)

CIP-ul rezolvă problema conform programului introdus şi rezultatele prelucrării făcute în circuitele electronice, sînt comunicate prin operaţii de ieşire.

Cum se soluţionează o problemă pe CIP ?

Pentru realizarea unei aplicaţii pe calculator, se parcurg următoarele etape:
- formularea problemei
- stabilirea intrărilor (date furnizate CIP-ului) şi ieşirilor (ce trebuie să furnizeze calculatorul)
- stabilirea modului de rezolvare
- verificarea programului pe calculator
- executarea curentă a programului.

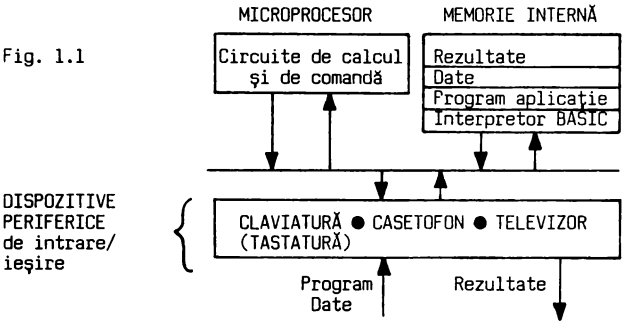
CE SE ÎNŢELEGE PRIN : HARDWARE şi SOFTWARE

Specialiştii denumesc echipamentele ce alcătuiesc calculatorul printr-un singur cuvînt: HARDWARE, iar programele sînt denumite: SOFTWARE.

Configuraţia HARDWARE a CIP-ului

Cîteva noţiuni generale despre fiecare componente şi despre funcţionarea lor în ansamblu sînt prezentate în cele ce urmează.
Schema de funcţionare a componentelor hardware ale unui calculator este următoarea (fig. 1.1):

Fig. 1.1



Schema funcţională descrisă se regăseşte concretizată în configuraţia hardware a calculatorului. Configuraţia minimă a CIP-ului cuprinde:
- microprocesor (tip Z80)
- memorie internă
- circuite de control a afişării
- circuite de interfaţă
- difuzor
- tastatură (claviatură)

Se adaugă următoarele "periferice" de intrare / ieşire :
- casetofon
- televizor sau monitor video.

Ce ar fi bine dacă aţi avea ?

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

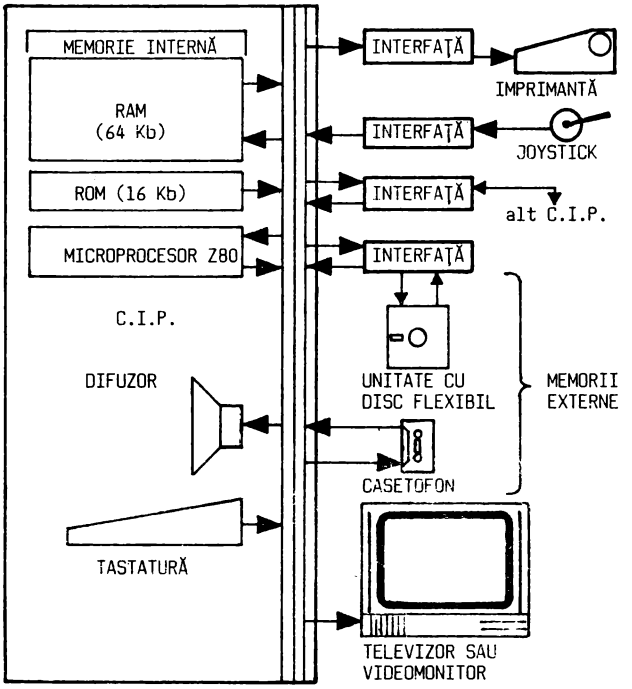


Fig. 1.2.

Microprocesorul

Microprocesorul cuprinde pe o mică pastilă de siliciu (în engleză CHIP), piese electronice în miniatură care descifrează, verifică şi execută comenzile şi calculele prevăzute în program.

Memoria internă

În memoria internă se află un interpretor care transformă programul aplicaţiei scris într-un limbaj uşor de învăţat (limbajul BASIC) într-un limbaj intern mult mai complicat, specific fiecărui tip de calculator (limbaj în "COD MAŞINĂ") care va putea fi executat de microprocesor. Interpretorul permite deci introducerea în memorie (fig.1.1) a diverselor programe de aplicaţii (SOFTWARE aplicativ) şi executarea lor.
Programele solicită datele necesare şi acestea sînt memorate într-o altă zonă a memoriei interne. În acest fel, CIP-ul ştie ce are de făcut şi cu ce să lucreze. Microprocesorul decodifică comenzile şi instrucţiunile şi le execută folosind datele disponibile.
Rezultatele prelucrării sînt (fig.1.1) expediate la dispozitivele periferice de ieşire.

Aşa după cum probabil aţi reţinut, în memoria internă se introduc informaţii prin operaţii de "încărcare" sau de "scriere", iar din memorie se "citesc" sau se "extrag" informaţiile introduse.
CIP-ul dvs. utilizează două tipuri de memorii:

CAPITOLUL 1

Memoria ROM

ROM (Read - Only - Memory) este o memorie al cărei conţinut este stabilit din fabricaţie şi care nu poate fi schimbat, ci poate fi numai "citit". Veţi constata că, după întrerúperea alimentării electrice a CIP-ului, la repunerea în funcţiune, programul înscris în ROM vă va sta la dispoziţie fără a fi nevoie să-l introduceţi dvs. O astfel de memorie se spune că este "nevolatilă" pentru că nu îşi pierde conţinutul la întreruperea tensiunii de alimentare.

Memoria RAM

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

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

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

Dispozitive periferice

Dispozitivele periferice sînt cele care permit introducerea programelor şi datelor sau comunicarea informaţiilor de ieşire pentru utilizator.
Sînt denumite "periferice" spre deosebire de microprocesor, care mai este denumit şi "unitate centrală" a aparatului.
CIP-ul necesită ca periferice în configuraţia minimă:

- tastatură
- televizor sau monitor vìdeo
- difuzor
- casetofon

Într-o configuraţie extinsă, prin achiziţionarea unor circuite de legătură specială (interfaţă), se mai pot ataşa:

- imprimantă
- unitate cu disc flexibil
- joystick

Tastatura

Claviatura CIP-ului (denumită TASTATURĂ) este asemănătoare cu a maşinii de scris (formată din TASTE), iar operaţia de utilizare se numeşte "TASTARE". Veţi afla la capitolul 3 că ea are un mod deosebit de funcţionare şi de utilizare.

Televizorul

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

Difuzorul

CIP-ul are încorporat un difuzor care vă va semnala încărcarea programelor, sau prin care puteţi obţine diferite sunete (melodii).

Casetofonul

Caseta magnetică este folosită ca suport de MEMORIE EXTERNĂ, deoarece pe ea se pot "salva" (înregistra) programele şi datele din memoria internă, pentru a preveni "volatilizarea" lor la întreruperea tensiunii de alimentare.
Pentru a "încărca" programe de pe casetă şi pentru a le "salva" pe casetă, folosiţi un casetofon obişnuit, dar în bună stare de funcţionare.
Un casetofon uzat, cu o mecanică defectă, vă va produce multe clipe neplăcute cînd, avînd o casetă cu nişte jocuri/programe frumoase, nu veţi reuşi să le încărcaţi sau veţi obţine mesajul "Tape loading error" indicînd o eroare de bandă.
O mică îndoitură a benzii, insesizabilă în cazul casetei cu muzică nu este acceptată de calculator.
În general, vă recomandăm să utilizaţi acelaşi casetofon la "salvarea" şi la "încărcarea" programelor, pentru a evita incompatibilităţi datorită nealinierii capetelor la două aparate deosebite.
Caseta magnetică prezintă avantajul unui preţ redus, dar are şi dezavantajul dificultăţii de acces la o anumită informaţie, deoarece permite numai o căutare secvenţială (program după program) de la începutul benzii, fapt care cere timp şi... răbdare.

Discul flexibil

Discul flexibil este un suport de memorie externă care înlătură dezavantajul casetei, el permiţînd accesul direct şi imediat la orice informaţie stocată.
Unitatea de disc flexibil este însă un dispozitiv mai scump, şi deci mai dificil de procurat.

Imprimanta

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

Joystick-ul

Joystick-ul permite mai uşor deplasarea pe ecran în orice direcţie, cu ajutorul unei manete, decît prin tastatură cu ajutorul căreia puteţi obţine numai "sus" / "jos" / "dreapta" / "stînga" / "foc".

Întrebări recapitulative

Î 1.1 Ce informaţii trebuie să introducem în calculator pentru a rezolva o problemă / a realiza un joc / a desena pe ecran ?
Î 1.2 Ce echipamente hardware trebuie să aveţi la dispoziţie pentru a lucra cu CIP-ul ?
Î 1.3 Casetofonul nu este permanent necesar. Cînd aveţi nevoie de el ?

Răspunsuri

R 1.1 Programul aplicaţiei şi datele necesare.
R 1.2 CIP-ul plus televizor plus casetofon plus o priză multiplă plus cabluri de legătură.
R 1.3 a). Cînd dorim să încărcăm programe "memorate" pe casetă.
b). Cînd am introdus prin tastatură programe mai mari pe care este bine să le "salvăm" pe casetă înainte de oprirea alimentării CIP-ului.

Presupunînd că sînteţi "începător", executaţi aplicaţia care urmează pentru a învăţa treptat utilizarea aparatului.

Aplicaţia practică AP.1

1. Priviţi tastatura şi localizaţi următoarele taste :
   - ▸ CAPS SHIFT (CS)
   - ▸ SYMBOL SHIFT (SS)
   - ▸ ENTER (ENT)
   - ▸ SPACE
   - ▸ Ø (cifră zero)

În continuare ele vor fi notate în text sub forma prescurtată din paranteze.
Tasta "SPACE" o folosiţi atunci cînd trebuie să introduceţi un "spaţiu" sau "blank" (în carte vom folosi pentru spaţiu/blank următorul desen:"␣". În exemple, cînd veţi întîlni acest semn, apăsaţi tasta SPACE. Cifra zero apare, atît pe tastatură cît şi pe ecran, tăiată cu o linie pentru a putea fi deosebit de litera "O".
Avînd la dispoziţie interpretorul BASIC-S puteţi comunica cu CIP-ul prin intermediul tastaturii. Poate la început vi se pare complicat dar prin explicaţiile pe care vi le voi da în continuare, vă puteţi antrena şi veţi constata ce ingenios este concepută tastatura.

2. Apăsaţi:

```
<ENT>        (tasta ENTER)
```

În colţul stînga jos al ecranului apare cursorul clipitor, cuprinzînd litera K .
CIP-ul aşteaptă nerăbdător ca dvs. să-i introduceţi o comandă în limbajul BASIC-S.

3.Introduceţi:

```
                 BORDER      4        <ENT>
                 ↑           ↑         ↑
apăsaţi tasta   "B"       tasta 4   tasta (ENTER)
```

I-aţi comandat CIP-ului să afişeze un contur (Border) de culoare verde (culoarea numărului 4). El v-a executat comanda şi v-a raportat că totul a fost în ordine (OK!).
Aţi realizat primul dialog cu CIP-ul.
Reţineţi că una din principalele caracteristici ale BASIC-ului este acest mod permanent de lucru "conversaţional" (sau "interactiv"), interpretorul fiind tot timpul gata să răspundă la mesajele/comenzile/instrucţiunile dvs. expediate către el prin apăsarea tastei ENTER <ENT>.
Deci cît timp nu aţi acţionat această tastă <ENT> CIP-ul (interpretorul BASIC) nu ştie ce aţi introdus dvs. prin tastatură (deşi informaţiile apar pe ecran) şi abia cînd acţionaţi <ENT> , el primeşte şi interpretează mesajul dvs. şi răspunde corespunzător.

4. Introduceţi o altă comandă care vă permite să "afişaţi" pe ecran litere/cifre/semne, în timp ce exersaţi utilizarea tastaturii. Afişaţi alfabetul:

```
PRINT    "abcdefghijklmnopqrstuvwxyz"       <ENT>
↑        ↑                          ↑         ↑
|        └─────── ghilimele ────────┘         |
apăsaţi   apăsaţi simultan tastele          apăsaţi
tasta "p"  SYMBOL SHIFT şi P                 ENTER
```

Dacă aţi greşit o literă o puteţi şterge tastînd în acelaşi timp CAPS SHIFT şi "O" (zero) ţinînd CS apăsat. În continuare vom nota această apăsare simultană cu semnul "+". Deci cînd vi se indică CS + O dvs. veţi apăsa în acelaşi timp pe CAPS SHIFT şi pe tasta Ø.
Limbajul BASIC, aşa după cum veţi constata, este destul de "prietenos" adică permite un dialog destul de uşor între utilizator şi CIP, dar are regulile lui stricte conform cărora vă poate înţelege. Dacă dvs. nu respectaţi aceste reguli, CIP-ul refuză să primească mesajul dvs., afişînd un semn de întrebare, sau vă răspunde cu "VARIABLE NOT FOUND" (variabilă negăsită). Dacă aţi ajuns într-o astfel de situaţie, ca neiniţiat aveţi două posibilităţi:
- ▸ ştergeţi tot rîndul, apăsînd pe CS + Ø;
- ▸ acţionaţi butonul RESET din spate.

apoi retastaţi rîndul (nu uitaţi ghilimelele), şi expediaţi mesajul cu ENTER.

5. Introduceţi alfabetul cu litere mari:

```
PRINT "ABCDEFGHIJKLMNOPQRSTUVWXYZ"       <ENT>
       ↑
       └Apăsaţi CS şi 2 şi apoi ABC...
```

6. Tastaţi cifrele cu un spaţiu între ele:

```
PRINT  "1␣2␣3␣4␣5␣6␣7␣8␣9␣Ø"              <ENT>
          ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
apăsaţi tasta "SPACE"
```

7. Introduceţi o linie cu următoarele semne prin tastările simultane indicate:

```
PRINT "!C=$%&'()<>;-+=:?/*.,"            <ENT>
```

!	(SS + 1)
C (a rond)	(SS + 2)
= (diez)	(SS + 3)
$	(SS + 4)
%	(SS + 5)
& (AND)	(SS + 6)
' (apostrof)	(SS + 7)
(	(SS + 8)
)	(SS + 9)
< (mai mic)	(SS + R)
> (mai mare)	(SS + T)
;	(SS + O)
- (minus)	(SS + J)
+	(SS + K)
=	(SS + L)
:	(SS + Z)
?	(SS + C)
/ (slash)	(SS + V)
*	(SS + B)
. (punct)	(SS + M)
, (virgulă)	(SS + N)

Remarcaţi că efectul tastei <ENT> este expedierea celor afişate, către interpretor, dar în acelaşi timp şi saltul cursorului la începutul unei linii noi. La alte tipuri de calculatoare această tastă este notată cu "RETURN" sau cu "CR" CARRIAGE RETURN - retur de car, prin analogie cu introducerea la începutul unei linii de maşina de scris).

8. Tastaţi alături următoarele litere/cifre/semne pentru a observa diferenţele grafice între ele:

```
PRINT "ØØØOOO''',,,IIIlll"            <ENT>
```

Ø (cifra zero) O (litera O) ' (apostrof) , (virgulă)
I (litera I) l (cifra 1)

9. Menţinînd apăsată orice tastă, literă/cifră, semnul respectiv se repetă pînă în momentul cînd încetăm apăsarea. Verificaţi, presupunînd că aveţi de subliniat un titlu:

apăsaţi CS + 2

```
PRINT  "Lecţia ␣ 1"        <ENT>
PRINT  "__________"        <ENT>
```

Pentru a sublinia prin repetarea semnului "_" (SS + J), nu este nevoie să apăsaţi de mai multe ori cele două taste, ci este suficient să le menţineţi apăsate pînă ajungeţi sub cifra 1 (dacă aţi depăşit-o ştergeţi cu CS + Ø înainte de a apăsa <ENT>).

10. Apăsaţi tasta "V" şi obţineţi:

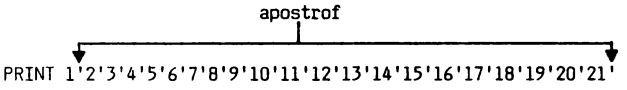
```
CLS        <ENT>
```

Observaţi (poate cu surprindere) că se şterge tot ecranul.

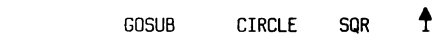
CAPITOLUL 2

Aţi introdus o comandă de "ştergere" a ecranului pe care interpretorul BASIC a înţeles-o şi a executat-o prompt (CLS - Clear Screen).

11. Numerotaţi liniile ecranului astfel:



12. Priviţi mai atent la o tastă, de exemplu la tasta "H". Deasupra ei sînt scrise patru caractere ce pot fi obţinute cu această tastă:



Încercaţi obţinerea lor apăsînd tastele indicate mai jos, observînd litera afişată în cursorul clipitor.
Atenţie! după afişarea fiecărui caracter, ştergeţi cu CS + Ø pentru a obţine din nou cursorul K .

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

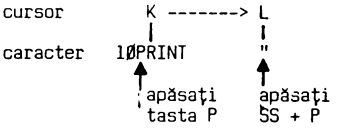
Pentru a obţine "CIRCLE" este nevoie de două tastări succesive:
1 - CS + SS - CS şi SS apăsate simultan
2 - SS + H - SS şi H apăsate simultan

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

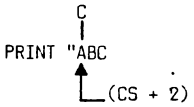
K (KEY WORD - cuvînt cheie) pentru comenzi/instrucţiuni
L (LETTERS - litere) pentru litere mici şi cifre
C (CAPITALS - majuscule) pentru litere mari
E (EXTENDED - mod extins) pentru caractere/comenzi
G (GRAPHICS - mod grafic) pentru simboluri grafice

Cum obţineţi litere mari/mici ?

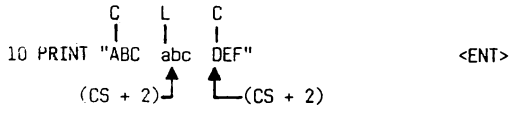
13. Tastaţi următoarele caractere observînd pe ecran litera din cursor şi caracterele afişate:



Apăsaţi simultan (CS + 2) şi cursorul devine C permiţînd introducerea literelor mari:

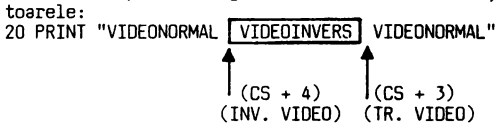


Reapăsaţi simultan (CS + 2) şi cursorul redevine L, putînd introduce litere mici:



Cum obţineţi VIDEONORMAL/VIDEOINVERS ?

14. Atunci cînd doriţi să apară pe ecran informaţii care să iasă în evidenţă faţă de altele, puteţi folosi modul de afişare videoinvers, adică în loc de caractere negre pe fondul alb al ecranului (videonormal), să apară caracterele albe pe fond negru (videoinvers). Tastaţi următoarele:

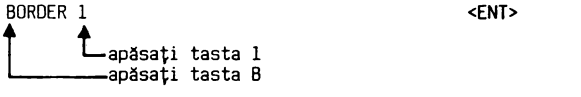


Aţi remarcat că trecerea dintr-un mod de afişare în altul se face tastînd (CS + 4) sau (CS + 3).
Dacă la apăsarea tastei <ENT> după linia 10 sau 20, vă apare un semn de întrebare în linie, BASIC-S vă semnalizează că aveţi o greşală. Greşala constă probabil în faptul că nu aţi introdus " după PRINT sau la sfîrşitul liniei. Într-un asemenea caz, ştergeţi linia cu DEL (CS + Ø) şi retastaţi-o atent.
Oricum reţineţi că BASIC-S nu acceptă informaţii neconforme cu regulile lui şi vă semnalizează prompt orice eroare prin " ? ".
Dacă aţi introdus corect linia 10 şi 20, pe ecran trebuie să aveţi:

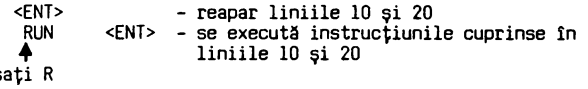
```
10 PRINT "ABC abc DEF"
20 PRINT "VIDEONORMAL VIDEOINVERS VIDEONORMAL"
```

Nu vă impacientaţi că linia 20 continuă dedesubt şi că nu este frumos aliniată cu partea de sus. BASIC-S nu ţine neapărat la frumuseţe, ci la corectitudine şi el se descurcă cu linii nealiniate dar corecte.

15. Introduceţi:



Aţi introdus o comandă BASIC-S care a fost executată imediat, avînd ca rezultat ştergerea celor două linii de pe ecran şi apariţia unui chenar (o bordură) de culoare închisă (albastră). Apăsaţi:



Pe ecran apare rezultatul execuţiei:

Caracterele sînt scrise în dreptul tastelor, prin trei culori corespunzătoare modului în care pot fi obţinute. După o perioadă, lucrînd curent, vă va fi foarte uşor să introduceţi orice caracter, numai prin simpla asociere vizuală între caracterul dorit, poziţia tastei şi culoare.
La început, de mare ajutor vă va fi Anexa B în care sînt indicate în ordine alfabetică, toate caracterele şi tastările necesare pentru obţinerea lor.
Pentru utilizarea tabelului din anexă vă fac cîteva precizări:
- cuvintele cheie reprezentînd comenzi sau instrucţiuni, sînt precedate de ":" (priviţi în Anexa B)

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

Ele se obţin 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 obţin astfel:

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

16. Exersaţi utilizarea Anexei B completînd rîndurile următoare:

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

Întrebări recapitulative

Răspundeţi în scris pe o foaie separată şi confruntaţi apoi răspunsurile dvs. cu cele corecte prezentate în continuare:

Î.2.1. Ce taste veţi folosi pentru:

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

Î.2.2. Care este efectul acţionării :

a) butonului RESET
b) tastei ENTER

Î.2.3. Cum opriţi CIP-ul?

Răspunsuri

R.2.1. a) SPACE
b) (CS+O)
c) (SS+1) (SS+C) (SS+K) (SS+J) (SS+O)
d) V <ENT>

R.2.2. a) Provoacă iniţializarea tuturor circuitelor fără a fi oprită alimentarea electrică

b) Expediază textul afişat către interpretorul BASIC şi mută cursorul la începutul liniei următoare

R.2.3. Întrerupînd alimentarea electrică.

CUM COMUNICĂM CU CIP-ul?

Din capitolul 1 aţi reţinut că un calculator electronic ştie să vă îndeplinească o dorinţă , în măsura în care l-aţi învăţat cum să o facă. Pentru a-l instrui, trebuie să-i introduceţi în memorie o succesiune logică de INSTRUCŢIUNI / COMENZI formînd un PROGRAM.
Pentru ca să înţeleagă programul şi să execute instrucţiunile comenzile, acestea trebuiesc exprimate într-un limbaj pe care-l "ştie" calculatorul.
CIP-ul este astfel construit încît să înţeleagă limbajul BAŞIC. BASIC-ul este un limbaj de programare cu o utilizare foarte largă datorită uşurinţei cu care poate fi învăţat şi multiplelor lui posibilităţi de utilizare.

De ce se cheamă BASIC

Numele limbajului BASIC provine din iniţialele cuvintelor în limba engleză:

Beginner's All purpose Symbolic Instruction Code,

care în traducere liberă arată că este destinat tuturor aplicaţiilor şi este pentru începători.
Într-adevăr, acest limbaj este mai uşor de învăţat decît limba engleză, el avînd un vocabular simplu, cu puţine cuvinte, totuşi cu reguli foarte stricte.
Chiar dacă nu cunoaşteţi limba engleză, nu va fi greu să învăţaţi vocabularul BASIC-ului deoarece CIP-ul se va dovedi un "încăpăţînat", refuzînd să treacă peste greşelile dvs. şi nu va executa decît ceea ce îi introduceţi corect.
Un program BASIC este constituit dintr-o mulţime de instrucţiuni scrise în acest limbaj şi introduse în CIP pentru a fi executate.

Cum execută CIP-ul un program?

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

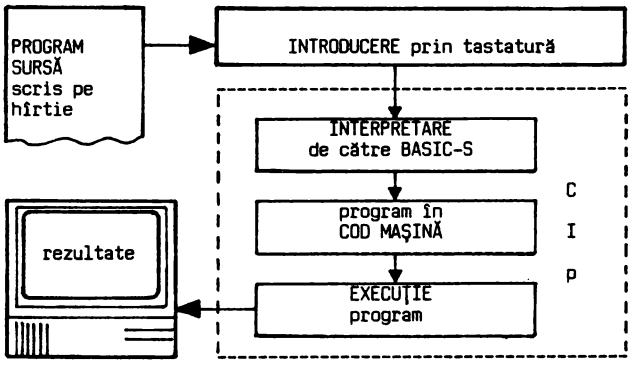


Fig. 3.1

Cum codifică CIP-ul comenzile şi instrucţiunile

Instrucţiunile / comenzile / datele, pentru a putea fi prelucrate în circuitele electronice ale calculatorului, sînt reprezentate sub formă de CARACTERE într-un cod în care sînt numerotate de la Ø la 255.
Priviţi în anexa A a manualului şi veţi găsi în coloana din stînga un număr zecimal reprezentînd codul şi imediat în coloana din stînga un număr zecimal reprezentînd codul şi imediat în coloana din dreapta "caracterul" respectiv. De exemplu:

COD	CARACTER
13	ENTER (comanda ENTER)
32	spaţiu (spaţiu sau blank)
43	+ (semnul +)
53	5 (cifra 5)
65	A (litera A)
97	a (litera a)
245	PRINT (instrucţiunea PRINT)
255	COPY (instrucţiunea COPY)

După cum observaţi, CARACTERELE pot fi:

- semne speciale (aritmetice / de punctuaţie)
- litere mari
- litere mici
- caractere grafice (▌ ▙ ...)
- instrucţiuni (PRINT, READ ...)
- comenzi (RUN, LOAD, ...)

De exemplu, cînd vom introduce următoarea instrucţiune, conţinutul ei va fi codificat astfel:

```
caractere:  PRINT  ␣  a  +  b
              ↓    ↓  ↓  ↓  ↓
coduri   :   245   32 97 43 98
```

Ca începător, dvs. nu aveţi decît să concepeţi, să scrieţi şi să introduceţi programul BASIC-sursă, restul fiind grija CIP-ului. Mai tîrziu, după iniţierea în BASIC, dacă veţi dori, puteţi învăţa să lucraţi direct în COD MAŞINĂ.

CUM ARATĂ UN PROGRAM BASIC ?

Deocamdată, să vedem cum arată un mic program scris în limbajul BASIC:

```
1Ø  BORDER  2               <-----┐
2Ø  INPUT   "a=?";a         <-----┤ linii de
3Ø  INPUT   "b=?";b         <-----┤ program
4Ø  PRINT   "S=";a+b:STOP   <-----┘
```

Programul este constituit din LINII DE PROGRAM numerotate (linia nr.1Ø, linia nr.2Ø, ş.a.m.d.).
BASIC-S admite numere de linii cuprinse între 1 şi 9999. Sînt admise numai numere întregi şi de obicei, numerotarea se face din 1Ø în 1Ø, pentru a putea intercala ulterior eventuale linii noi.

Cum se încheie introducerea unei linii?

Avînd calculatorul pornit introduceţi programul de mai sus. După ce aţi tastat conţinutul unei linii, acţionaţi <ENT> pentru ca linia să fie memorată şi pentru a se produce saltul la o linie nouă. În timpul introducerii, linia este afişată la baza ecranului.

Linia de program memorată

La acţionarea tastei <ENT>, interpretorul BASIC-S verifică dacă linia a fost corect introdusă şi dacă o găseşte astfel, permite memorarea ei, semnalată pe ecran prin afişarea în partea superioară a ecranului.

Se memorează o linie cu erori?

Dacă după <ENT> apare în linie un semn de întroducere, el semnalează o eroare. Într-o astfel de situaţie, ştergeţi cu (CS+Ø) linia pînă la capăt şi retastaţi-o corect.
După introducerea liniei 4Ø, pe ecran veţi avea programul afişat ca în Fig.3.2 :

```
1Ø  BORDER  2
2Ø  INPUT   "a=?";a
3Ø  INPUT   "b=?";b
4Ø  PRINT   "S=";a+b:STOP



[K]
```

Fig.3.2

Linie de program cu execuţie imediată

În acest moment programul este memorat şi aşteaptă o comandă pentru a fi executat. Introduceţi comanda RUN acţionînd tasta "R". Ea va fi afişată în dreptul cursorului clipitor K şi după <ENT> va declanşa execuţia programului. Reţineţi că o linie nenumerotată (de ex. RUN) este executată imediat fără a fi memorată.
Pentru a anticipa ce se va întîmpla prin execuţia programului, să îl comentăm linie cu linie:

```
1Ø BORDER  2          - afişează un chenar roşu al ecranului
2Ø INPUT "a=?";a   }  - cere valori pentru a şi b şi le
3Ø INPUT "b=?";b   }    memorează
4Ø PRINT "S=";a+b:STOP - afişează suma S=a+b şi opreşte
                         execuţia programului
```

Cum i-aţi introdus comanda RUN, CIP-ul prompt a trecut la treabă. El execută prima linie (colorează chenarul) şi va cere o valoare pentru "a". Introduceţi de exemplu:

a=?1Ø <ENT>

Acum va cere o valoare pentru "b":

b=?2Ø <ENT>

Aproape instantaneu vă afişează rezultatul execuţiei întregului program (Fig.3.3.).

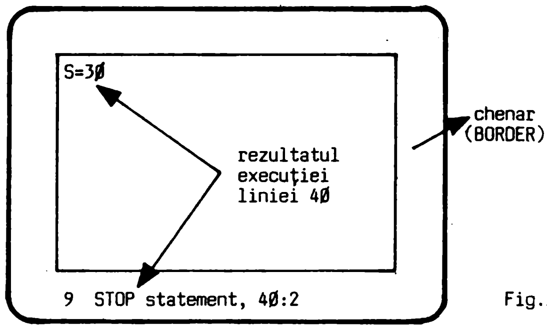


Fig.3.3.

Linii de program: - simple; multiple

Să analizăm puţin formatul unei linii de program.
Linia de program poate fi:

▸ simplă - cu o singură instrucţiune

```
  1Ø BORDER 2          <ENT>
  ↗      ↑               ↑
număr instrucţiune   terminator de
linie                   linie
```

▸ multiplă - cu mai multe instrucţiuni separate între ele prin ":" (două puncte)

```
4Ø PRINT "S=";a+b:STOP       <ENT>
   ─────────────── ────
          ↑          ↑
   o instrucţiune    └altă instrucţiune
```

Programul de mai sus poate fi deci introdus pe o singură linie multiplă. Verificaţi tastînd:

```
1Ø BORDER 2: INPUT "a=?";a: INPUT "b=?"; b:
PRINT "S="; a+b:STOP    <ENT>
```

Remarcaţi că vechea linie 1Ø este înlocuită cu noua linie multiplă. Pentru a executa programul, să ştergem liniile 20 3Ø, 4Ø devenite inutile.

Cum ştergem o linie?

Ştergerea unei linii complete se obţine foarte uşor prin introducerea numărului ei urmat de <ENT>.

```
2Ø  <ENT>    - linia 2Ø dispare
3Ø  <ENT>    - linia 3Ø dispare
4Ø  <ENT>    - linia 4Ø dispare
```

Daţi din nou comanda de execuţie a programului:

```
RUN    <ENT>
a=?1Ø  <ENT>
b=?2Ø  <ENT>
```

Rezultat:

```
S=3Ø
9 STOP  statement, 1Ø:5
```

Rezultatul calculului este acelaşi, dar diferă mesajul de oprire a programului. La prima execuţie, oprirea s-a făcut prin instrucţiunea STOP, a doua instrucţiune din linia 4Ø.

```
        9 STOP statement, 4Ø:2
        ↑                 ↑ ↑
număr al                  │ └─a doua instrucţiune
mesajului                 └───linia 4Ø
```

La a doua execuţie oprirea s-a făcut prin instrucţiunea STOP, a cincea în linia 1Ø:

```
        9 STOP statement, 1Ø:5
                           ↑ ↑
            linia 1Ø ──────┘ └──── a cincea instrucţiune
```

Înainte de a trece la aplicaţia practică, să recapitulăm ce aţi învăţat din limbajul BASIC pînă acum:

- informaţiile sînt 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 linii numerotate (care sînt executate numai după o comandă specială) şi numerotate (care sînt executate imediat);
- pe o linie multiplă se pot introduce mai multe instrucţiuni/comenzi, separate prin ":" ;
- execuţia unui program se obţine prin comanda RUN;
- modificarea unei linii se obţine prin introducerea noului conţinut cu acelaşi număr de linie;
- ştergerea unei linii se obţine prin introducerea numai a numărului ei (linie vidă).

APLICAŢIE PRACTICĂ AP 3

1. Anulaţi programul anterior, introducînd comanda imediată:

```
NEW              <ENT>
↑
└── tasta "A"
```

2. Tastaţi:

```
1Ø  LET  a=7       <ENT>
2Ø  LET  b=13      <ENT>
3Ø  LET  c=a+b     <ENT>
4Ø  PRINT  c       <ENT>
5Ø  STOP           <ENT>
```

Dacă aţi greşit la introducere, linia nu este admisă cînd acţionaţi <ENT> şi eroarea vă este semnalată cu "?" clipitor.
Să simulăm acum o astfel de eroare pentru a vedea cum procedaţi. Tastaţi linia 3Ø astfel:

```
3Ø  LET  c+a+b          <ENT>
          ↑
          └─în loc de "=" aţi introdus "+", tastele
            fiind alăturate. Eroarea vă este
            semnalată astfel:
```

```
3Ø  LET c [?] +a+b [L]
```

Ştergeţi linia cu DEL (CS+Ø) şi reintroduceţi conţinutul corect.

3. Introduceţi comanda de ştergere a ecranului (vezi capitolul 2)

```
> _ _ _ _       <ENT>
```

4. Ecranul fiind şters, să afişăm programul aflat în memorie. Introduceţi comanda de listare a lui:

```
LIST          <ENT>
```

Pe ecran reapare programul.

5. Instrucţiunile LET pot fi traduse astfel

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

Instrucţiunea PRINT afişează valoarea lui c.
Încercaţi să scrieţi ce va fi afişat după execuţia programului:

```
--------

9 STOP statement, _ _ _ _:_ _
```

6. Introduceţi comanda de execuţie a programului şi verificaţi/corectaţi răspunsul dvs:

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

Rezultat:

```
--------

9 STOP statement, _ _ _ _:_ _
                        ↑    ↑
  numărul liniei cu ────┘    └a cîta instrucţiune
  instrucţiunea STOP          pe linie.
```

7. Tastaţi:

```
1Ø  LET  a=127        <ENT>
```

Observaţi că vechea linie 1Ø este înlocuită cu noul conţinut introdus.

8. Ce rezultat veţi obţine după execuţia programului?

```
------------------
```

9. Executaţi programul şi comparaţi rezultatul obţinut cu cel intuit de dvs.

```
------------------
```

10.Tastaţi:

```
5Ø          <ENT>
```

11.Comparaţi programul afişat cu cel anterior. Ce observaţi?

12.Executaţi programul şi analizaţi ce se afişează:

```
14Ø
Ø OK,  4Ø:1
↑
└numărul mesajului
```

Observaţi că CIP-ul este un aparat "descurcăreţ". Chiar fără să-i spuneţi unde să se oprească (fără linia 5Ø - STOP), el a executat programul şi v-a dat un mesaj că totul a decurs bine (OK) oprindu-se singur la prima instrucţiune a ultimei linii (4Ø:1).

13. Introduceţi comanda:

```
NEW        <ENT>
```

Transcrieţi mesajul obţinut:

----

14. Încercaţi să afişaţi programul:

```
LIST       <ENT>
```

15. Încercaţi să executaţi programul:

```
RUN        <ENT>
```

Nu mai puteţi obţine nimic pentru că programul a fost şters din memorie ca urmare a comenzii NEW, în vederea introducerii unui nou program.

16. Reintroduceţi programul:

```
1Ø  LET  a=7     <ENT>
2Ø  LET  b=13    <ENT>
3Ø  LET  c=a+b   <ENT>
4Ø  PRINT c      <ENT>
5Ø  STOP         <ENT>
```

17. Tastaţi:

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

18. Priviţi programul afişat şi veţi constata că "ascultătorul" CIP a memorat cuminte noile instrucţiuni la locul potrivit: linia 35 între 30 şi 4Ø şi linia 45 între 4Ø şi 5Ø. Vom spune că liniile 35 şi 45 au fost "INSERATE" (intercalate) în program.
Mai puteţi remarca ceva în linia 45

```
45  >  PRINT   d
    ↑
    └──prompter de linie curentă
```

Semnul ">" reprezintă un prompter (indicator) de LINIE CURENTĂ şi el arată întotdeauna ultima linie introdusă sau corectată.
Linia curentă poate fi modificată prin editarea (afişarea) ei la baza ecranului, printr-o comandă de editare tastînd CS+1.
Incercaţi chiar acum următoarea secvenţă de operaţii:

BORDER	5	<ENT>	- apare chenarul, dar dispare programul de pe ecran
		<ENT>	- programul este reafişat
(CS+1)			- 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 4Ø
(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

Prompter-ul de linie poate fi deplasat în sus (CS+7), sau în jos (CS+6) avînd astfel posibilitatea să stabiliţi ca linie curentă, orice linie a programului.
Verificaţi deplasarea prompter-ului de linie tastînd (CS+7) sau (CS+6).

19. Analizaţi programul afişat şi încercaţi să intuiţi care vor fi rezultatele execuţiei lui:

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

20. Introduceţi comanda de execuţie şi comparaţi rezultatele.

21. Executaţi următoarea secvenţă:

<ENT>	- programul este reafişat
RUN	- programul este şters de pe ecran şi apar rezultatele

22. Reintroduceţi linia 3Ø cu cîte un spaţiu / blank între caractere astfel;

```
30 LET  c ␣ = ␣ a ␣ + ␣ b     <ENT>
          ↑   ↑   ↑   ↑
          └─spaţii────┘
```

Executaţi programul! Observaţi că totul decurge normal. BASIC-ul ignoră deci spaţiile.

23. Ştergeţi din program liniile 45 şi 35 introducînd două linii vide.

24. Modificaţi liniile 1Ø şi 2Ø retastîndu-le astfel:

```
1Ø  INPUT  "a=?";a     <ENT>
2Ø  INPUT  "b=?";b     <ENT>
```

25. Modificaţi linia 4Ø prin editare astfel:

- stabiliţi linia curentă dînd comanda

```
LIST  4Ø   <ENT>
```

Apăsînd a doua oară pe <ENT> reapare programul cu prompterul în linia 4Ø
-coborîţi linia curentă în poziţia de editare cu (CS+1);
-deplasaţi cursorul spre dreapta cu (CS+8), sau spre stînga cu (CS+5) şi introduceţi ghilimelele astfel încît linia să arate astfel:

```
4Ø  PRINT  "c=a+b=";c    <ENT>
```

26. Noul program instruieşte CIP-ul astfel:

1Ø	INPUT	"a=?";a	- cere valorile lui a şi b şi le memorează
2Ø	INPUT	"b=?";b	
3Ø	LET	c=a+b	- calculează c=a+b
4Ø	PRINT	"c=a+b=";c	- afişează c=a+b
5Ø	STOP		- opreşte execuţia programului

27. Executaţi programul:

```
RUN          <ENT>
a=?1Ø        <ENT>
b=?2Ø        <ENT>
c=a+b=3Ø
9  STOP statement, 5Ø:1
```

28. Repetaţi:

```
RUN          <ENT>  -rezultatul anterior dispare
a=?9Ø        <ENT>
b=?1Ø        <ENT>
c=a+b=1ØØ
STOP
```

29. Încercaţi altă formă de execuţie a programului:

```
GOTO  1Ø   <ENT>  - rezultatul anterior nu mai dispare de
                    pe ecran
```

tasta "G"

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

Iar:

```
GOTO 1Ø        <ENT>
a=?2530        <ENT>
b=?39870       <ENT>
c=a+b=42400
STOP
```

Observaţi că execuţia programului se poate face cu RUN şi în acest caz valorile unei execuţii anterioare sînt anulate, sau cu o instrucţiune

```
GOTO  n
```

de trimitere la o linie "n" a programului de unde doriţi să înceapă execuţia.
În acest ultim caz, valorile anterioare nu sînt anulate.

30. Înlocuiţi conţinutul liniei 5Ø cu următorul:

```
5Ø GOTO 1Ø  <ENT>
```

Acum programul se va executa astfel:

```
1Ø (cere o valoare pentru a)
2Ø (cere o valoare pentru b)
3Ø (calculează c=a+b)
4Ø (afişează c=a+b=valoarea lui c)
5Ø (mergi la linia 1Ø)
RUN

a=?5Ø        <ENT>
b=?15        <ENT>
c=a+b=65     <ENT>
a=?
```

deoarece aţi înlocuit instrucţiunea de oprire STOP cu instrucţiunea GOTO ("mergi la") programul nu s-a mai oprit din execuţie şi va cere în continuare alte şi alte valori pentru a şi b. Încercaţi !

31. În momentul în care doriţi să întrerupeţi această execuţie fără sfîrşit a programului, la solicitarea unei valori pentru a sau b, introduceţi:

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

Programul va fi oprit cu mesajul:

```
H  STOP  in INPUT
```

32. Anulaţi programul existent în memorie şi introduceţi:

```
NEW
1Ø  PRINT "multe salutări, prietene!"
2Ø  PAUSE Ø
3Ø  GOTO  1Ø
```

Executaţi programul:

```
RUN    <ENT>
```

Veţi primi mesajul:

```
multe salutări, prietene!
```

Apăsaţi pe orice tastă şi veţi fi din nou salutat:

```
multe salutări, prietene!
```

În continuare, la apăsarea oricărei taste, veţi primi la nesfîrşit salutări. Dar pînă cînd?. Pînă veţi dori să întrerupeţi primirea de mesaje, determinată de fapt, de execuţia repetată a programului (se spune că programul "ciclează", adică a intrat într-un ciclu fără sfîrşit).
Introduceţi comanda de întrerupere BREAK (CS+SPACE) şi execuţia programului va fi întreruptă, afişîndu-se mesajul:

```
L. BREAK into program, 2Ø:1
```

Apăsaţi <ENT> pentru a fi reafişat programul şi priviţi-l cu atenţie. Ce s-a întîmplat?
Ultima instrucţiune a programului GOTO 1Ø a determinat de fiecare dată următorul ciclu:

```
1Ø   - tipăreşte mesajul
2Ø   - pauza pînă la prima tastare
3Ø   - mergi la 1Ø
```

Comanda BREAK va permite întreruperea execuţiei unui program în orice moment, cu excepţia cazului cînd CÎP-ul aşteaptă un răspuns la o instrucţiune INPUT, cînd puteţi întrerupe execuţia numai prin STOP, aşa cum aţi văzut în exemplul precedent.
Execuţia unui program întreruptă prin BREAK/STOP, poate fi reluată cu instrucţiunea CONTINUE (prescurtat CONT). Încercaţi!
O ultimă noutate în acest capitol:
Instrucţiunea PAUSE permite oprirea execuţiei programului, la dorinţa programatorului, astfel:

```
PAUSE  n  - opreşte execuţia pe o perioadă de (n/50)
             secunde
PAUSE Ø - opreşte execuţia pînă la apăsarea oricărei
BREAK      <ENT>
_____      <ENT>  - ştergeţi ecranul
_____             - reluaţi programul întrerupt

multe salutări, prietene!
_____             - întrerupeţi programul
_____             - ştergeţi programul din memorie
```

Recapitulare: operaţii şi comenzi pentru elaborarea şi execuţie 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 poziţionează prompter-ul de linie curentă cu ↓ (CS+6) sau cu ↑ (CS+7).
  .se deplasează cursorul pe linie cu → (CS+8) sau ← (CS+5), se şterg (CS+Ø) caracterele nedorite şi se introduc cele noi.
  .se apasă <ENT>.
- Execuţia unui program:
  RUN sau RUN nr. linie
  GOTO nr. linie
- Afişarea unui program
  LIST sau LIST nr. linie
- Se afişează un ecran încheiat cu
  scroll? (alt ecran?) puteţi răspunde:
  .apăsînd orice tastă, pentru continuarea afişării;
  .tastînd "N" pentru a renunţa la afişarea programului
- Ştergerea unui program din memorie:
  NEW
- Întreruperea execuţiei unui program:
  BREAK (CS+SPACE)
  STOP (CS+A)
  execuţia se reia cu:
  CONT
  PAUSE n -opreşte execuţia pe o perioadă de n/50 sec.
  PAUSE Ø -opreşte execuţia programului pînă se apasă orice tastă

- Ştergerea ecranului
  CLS

ÎNTREBĂRI RECAPITULATIVE

Î 3.1 Cum ştergeţi o linie dintr-un program BASIC?
Î 3.2 Cum inseraţi o linie nouă?
Î 3.3 Cum schimbaţi conţinutul unei linii deja introdusă?
Î 3.4 Care este comanda de afişare a unui program memorat?
Î 3.5 Cum ştergeţi din memorie un program memorat?
Î 3.6 Instrucţiunile fără număr de linie au nevoie de comanda RUN pentru a fi executate?
Î 3.7 Cum este indicată linia curentă şi cum puteţi stabili o linie curentă?
Î 3.8 Cum deplasaţi la stînga şi la dreapta cursorul într-o linie de editare?
Î 3.9 Cum puteţi întrerupe execuţia unui program şi cum reluaţi execuţia?
Î 3.10 Cum opriţi execuţia unui program pînă la prima tastare?

RĂSPUNSURI

R 3.1 a)Introducînd o linie vidă cu acelaşi număr de linie. b)Ştergînd înapoi cu DEL (CS+Ø) caracter după caracter.
R 3.2 Cu un număr de linie intermediar.
R 3.3 Introducînd o linie cu acelaşi număr, dar cu un nou conţinut.
R 3.4 LIST.
R 3.5 Cu ajutorul comenzii NEW.
R 3.6 Nu, ele sînt imediat executate.
R 3.7 Linia curentă este indicată de prompter-ul de linie şi 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?

Operaţii; Operatori; Constante; Variabile; Expresii

Deoarece acum aţi făcut "primii paşi" în BASIC, puteţi trece la noţiuni de detaliu ale limbajului.
O precizare importantă!
În continuare nu veţi mai avea specificată acţionarea tastei ENTER. Nu uitaţi însă ca de fiecare dată să o tastaţi, pentru că altfel cursorul va clipi uimit la sfîrşitul liniei de program, şi informaţiile tastate de dvs. vor rămîne numai pe ecran, fără a fi cunoscute de calculator.

APLICAŢIE PRACTICĂ AP4

1. Introduceţi următorul program:

```
1Ø INPUT "x=";x:PRINT"x=";x
2Ø INPUT "y=";y:PRINT"y=";y
3Ø LET z=x+y
4Ø PRINT "======"
5Ø PRINT "z=";z
6Ø STOP
```

Aşa cum probabil v-aţi dat seama, în linia 30 se face o operaţie de adunare între două numere, care vă sînt cerute în liniile 10 şi 20.
Executaţi programul:

(nu uitaţi ENTER!)

x=14.7 (în BASIC numerele zecimale se scriu cu punct, nu cu virgulă)
y=15.3
======
z=3Ø

2. Schimbaţi linia 30 astfel:

```
3Ø LET z=x-y
```

Înainte de a executa programul, scrieţi mai jos care va fi rezultatul dacă veţi introduce x=20 şi y=6

z=_ _ _ _ _

Acum executaţi programul şi comparaţi rezultatul cu cel presupus de dvs.

3. Introduceţi:

```
3Ø LET z=x*y
```

Afişaţi programul.
Executaţi programul cu x=8 şi y=3

z=_ _ _ _

Ce operaţie aritmetică se notează cu "*"? _ _ _ _ _ _ _ _ _

4. Introduceţi:

```
3Ø LET z = x / y
```

Afişaţi programul şi executaţi-l atribuind variabilelor valorile: x=36 şi y=4

z=_ _ _ _

Ce operaţie aritmetică se notează cu "/"? _ _ _ _ _ _ _ _ _

5. Încercaţi:

```
3Ø LET z=x+y-y/4
```

Afişaţi programul şi studiaţi-l puţin. Ce rezultat credeţi că se va obţine dacă introduceţi x=6 şi y=12

z=_ _ _ _ _ _

Executaţi programul şi comparaţi rezultatul cu cel anticipat

6. Anulaţi programul şi introduceţi:

```
1Ø LET   a=5*5
2Ø LET   b=5 ↑ 2        (" ↑ " se obţine cu SS+H)
3Ø PRINT "a=";a
4Ø PRINT "b=";b
5Ø STOP
```

Executaţi programul şi veţi remarca obţinerea aceluiaşi rezultat. Prin " ↑ " în BASIC se notează ridicarea la putere. sau exponenţierea, aşa încît în linia 2Ø se calculează "cinci la puterea a doua" ceea ce este echivalent cu linia 1Ø.

7. Schimbaţi:

```
1Ø LET' a=5*5*5
2Ø LET  b=5 ↑ 3
```

Cît credeţi că se va obţine?

a=_ _ _ _

b=_ _ _ _

Executaţi programul şi confruntaţi rezultatele.

8. Completaţi operatorii aritmetici folosiţi pentru următoarele operaţii:
   scădere_ _ _ _ _
   exponenţiere_ _ _ _ _ _
   înmulţire_ _ _ _ _ _
   împărţire_ _ _ _ _ _
   adunare_ _ _ _ _ _

9. Anulaţi vechiul program şi introduceţi o instrucţiune imediată (fără număr de linie):

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

notaţi rezultatul a=_ _ _ _ _ _ _ _=_ _ _ _

Să analizăm puţin ce s-a întîmplat:
- -instrucţiunea a fost executată imediat (fără RUN);
- -instrucţiunea PRINT a executat calculul expresiei şi apoi a afişat rezultatul;
- -operaţiile s-au făcut într-o anumită ordine.

Instrucţiunea respectivă este transcrierea în limbajul BASIC a expresiei aritmetice:

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

CIP-ul a efectuat mai întîi înmulţirea şi împărţirea şi apoi a făcut adunarea.

10. Introduceţi, notaţi, analizaţi şi comparaţi 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. înmulţirea/împărţirea
3. adunarea

Dacă întîlneşte paranteze evaluează prioritar conţinutul lor

11. Introduceţi şi notaţi rezultatele:

```
PRINT  2*1ØØ                  _ _ _ _ _ _ _
PRINT  2*1ØØ*1ØØ              _ _ _ _ _ _ _
PRINT  2*1ØØ*1ØØ*1ØØ          _ _ _ _ _ _ _
PRINT  2*1ØØ*1ØØ*1ØØ*1ØØ      _ _ _ _ _ _ _
```

Remarcaţi că rezultatul ultimei operaţii a fost afişat sub forma 2E+8 în loc de: 2ØØØØØØØØ.

12. Ştergeţi ecranul (CLS) şi introduceţi:

```
                              rezultat
                              --------
PRINT 2/1ØØ                   .Ø2
PRINT 2/(1ØØ*1ØØ)             _ _ _ _ _
PRINT 2/(1ØØ*1ØØ*1ØØ)         _ _ _ _ _
```

Ultimul rezultat în loc de Ø.ØØØØØ2 a fost afişat 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 şasea poziţie (cu zerouri în faţă).

13. Scrieţi în forma acceptată de interpretorul BASIC următoarele expresii aritmetice. Cum credeţi că vor fi afişate rezultatele ?

```
     7
1Ø*1Ø      _ _ _ _ _ _ _
     6
1Ø:1Ø      _ _ _ _ _ _ _
```

Introduceţi instrucţiunile corespunzătoare şi verificaţi forma de afişare a rezultatelor.

14. Introduceţi:

```
1Ø  PRINT  17
2Ø  LET a=17:PRINT  a
3Ø  LET a=39:PRINT  a
```

Executaţi programul!
În linia 1Ø cerem calculatorului să afişeze numărul 17. În linia 2Ø cerem acelaşi lucru, dar prin intermediul literei "a" căreia îi vom putea da şi alte valori. Limbajul BASIC vă permite să lucraţi atît cu valori CONSTANTE (neschimbate) cît şi cu VARIABILE (nume care pot lua diferite valori). Constantele şi variabilele sînt admise în anumite condiţii despre care veţi afla multe în continuarea acestui capitol. Dacă nu veţi respecta aceste condiţii (din necunoaştere sau neatenţie) interpretorul BASIC va refuza să le ia în considerare semnalînd eroarea cu un semn de întrebare în linia de editare.

15. Încercaţi validitatea unor constante numerice introducînd valorile date pentru x şi notînd ce se afişează:

```
NEW
1Ø INPUT "x=";x:PRINT x
2Ø GO TO 1Ø
RUN
```

valori pentru x:	ce se afişează:
123	123
3.14	_ _ _ _ _
12345678	_ _ _ _ _
123456789	_ _ _ _ _
Ø.123	_ _ _ _ _
.123	_ _ _ _ _
+21Ø	_ _ _ _ _
-21Ø	_ _ _ _ _
12abc	12 ? abc (eroare nu sînt admise litere)

Ştergeţi literele şi valoarea va fi admisă.

16. Schimbaţi linia 1Ø:

```
x=STOP      <ENT>  (STOP obţineţi cu SS+A)
1Ø INPUT    "x=";x$:PRINT x$
RUN
```

valori pentru x:	ce se afişează:
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.

17. Acţionaţi butonul RESET şi introduceţi cu atenţie:

```
1Ø INPUT "caracter 1:␣";a$,"caracter 2:␣";b$
2Ø IF  a$ > b$  THEN PRINT a$;"␣ > ␣";b$:GO TO 1Ø
3Ø IF  a$ < b$  THEN PRINT a$;"␣ < ␣";b$:GO TO IØ
4Ø PRINT  a$;"␣ = ␣";b$:GO TO 1Ø
```

Programul compară două caractere introduse la cerere şi afişează relaţia dintre ele, folosind operatorii BASIC

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

Compararea se face pe baza codurilor ASCII ale caracterelor, indicate în anexa A. Priviţi 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: % < ?.
Verificaţi executînd programul şi confruntînd 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. Introduceţi:

```
RESET
1Ø CLS:INPUT "a=(5...1Ø)=␣";a:PRINT "a=␣";a
2Ø IF a>=5 AND a<=1Ø THEN CIRCLE 125, 85, 3Ø
   :PAUSE 2ØØ: GO TO 1Ø
3Ø IF a<5 OR a>1Ø  THEN PRINT AT 18,1;FLASH 1;
"număr în afara intervalului!":PAUSE 3ØØ:GO TO 1Ø
```

Programul vă cere să-i introduceţi un număr cuprins între 5 şi 1Ø şi în funcţie de răspunsul dvs. reacţionează astfel:
1) dacă numărul introdus va fi mai mare decît sau egal cu 5 (a>=5) ŞI (în engleză AND) mai mic sau egal cu 1Ø (a<=1Ø), adică dacă va fi cuprins în intervalul (5,1Ø), CIP-ul vă va oferi ca premiu un cerc în mijlocul ecranului (cu instrucţiunea CIRCLE);
2) dacă numărul introdus va fi mai mic decît 5, SAU (în engleză OR) mai mare decît 1Ø, CIP-ul vă va atrage atenţia clipind că numărul nu se încadrează în valorile premiabile (cu instrucţiunea FLASH 1).
Executaţi programul şi verificaţi vigilenţa calculatorului (de ex. cu a=7, apoi a=3).
Pe scurt, programul poate fi comentat astfel:

```
-şterge ecranul
-linia 1Ø
     . cere un număr între 5 şi 1Ø
     . afişează numărul introdus
-linia 2Ø
     . dacă numărul este între 5 şi 1Ø desenează un
       cerc
     . aşteaptă 4 secunde şi apoi întoarce-te la
       linia 1Ø
-linia 3Ø
     . dacă numărul este mai mic decît 5 sau mai
       mare decît 1Ø afişează clipitor "număr în
       afara intervalului"
     . aşteaptă 6 secunde şi apoi întoarce-te la
       linia 1Ø.
```

19. Încercaţi acum dacă CIP-ul este suficient de "deştept" încît să înţeleagă şi informaţii spuse invers. În programul următor să-i cereţi să se descurce în următoarea situaţie:
- îi memoraţi printr-o variabilă şir numele unei păsări cunoscute: "cuc";
- el să vă ceară un nume de pasăre;
- dacă vreţi să-i încercaţi răbdarea nu-i daţi la început ce ştie el, ci orice alte nume de păsări, caz în care să vă ceară cu insistenţă pasărea pe care o ţine el minte;
- cînd îi veţi satisface, dorinţa, să afişeze bucuros cîntecul cucului !

Introduceţi:

```
RESET
1Ø CLS:LET p$="cuc"
2Ø INPUT "o pasare:␣";x$
3Ø IF NOT p$=x$ THEN PRINT "alta pasare!":PAUSE 5Ø:
   GO TO 1Ø
4Ø PRINT AT 1Ø,12;FLASH 1;"cucu!!!":PAUSE 2ØØ:
   GO TO 1Ø
```

În program este utilizat operatorul logic NOT (în româneşte "nu") pentru a pune condiţia: dacă p$ NU este egal cu x$ ATUNCI solicită din nou un şir pentru x$.
Şi acum după ce aţi constatat în dialog direct cu CIP-ul cum calculează, dacă nu toate noţiunile vă sînt foarte clare, ele vor fi reluate cu explicaţii suplimentare şi în exerciţii/întrebări la sfîrşitul fiecărui capitol.

NOŢIUNI DE BAZĂ. OPERATORI

Limbajul BASIC foloseşte următoarele tipuri de operatori:
a) operatori aritmetici:
- + adunare
- - scădere
- * înmulţire
- / împărţire
- ↑ ridicare la putere (exponenţiere)

b) operatori relaţionali
- = egalitate
- <> neegalitate
- > mai mare decît
- < mai mic decît
- >= mai mare decît sau egal cu
- <= mai mic decît sau egal cu

c) operatori logici
- AND "şi"
- OR "sau"
- NOT negaţie

d) operator de alipire (concatenare) şiruri
- + plus

PRIORITĂŢI ÎN EXECUŢIE

Prioritatea (ordinea) în execuţie a operaţiilor este următoare:
1- expresiile din paranteze
2- ↑ (ridicarea la putere/exponenţierea)
3- *,/ (înmulţirea,împărţirea)
4- +,- (adunarea, scăderea)
5- =,<,>,<=,>=,<>
6- NOT
7- AND
8- OR

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

De exemplu expresia $3 + \frac{4^2}{8} - 2$ se scrie în limbajul BASIC 3+4 ↑ 2/8-2 şi se execută astfel:

```
1. 3+16/8-2
2. 3+2-2
3. 5-2
rezultat:3
```

Încercaţi:

```
NEW
PRINT 3+4 ↑ 2/8-2
```

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

Expresia în limbajul BASIC se scrie astfel:

```
(3*(4*5+6↑2)+7)*(4↑2-3)
 |  |___1|  |   |___|
 |______2___|     3
```

Ordinea în execuţie este:

1- 6↑2
2- 4*5+36
3- 3*56+7
4- 4↑2
5- 16-3
7- 175*13

Calculaţi singuri şi notaţi rezultatul: _ _ _ _ _ _ _
Verificaţi cu o instrucţiune PRINT imediată.
Probabil v-aţi întrebat deja: 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:

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

În BASIC se poate scrie în două feluri:

(a↑2+b↑2)*0.5 sau SQR(a↑2+b↑2)

CONSTANTE NUMERICE

Constantele numerice pot avea:

-format întreg:
  16 sau +16
  -305
-format zecimal (real):
  6.25 (echivalent cu 6,25)
  0.95 sau .95 (pentru 0,95)
-format exponenţial:
  E ± n (echivalent cu $10^{\pm n}$) unde n este un număr întreg cuprins între -38 şi +37

Exemple:

```
-1234=-1.234E+3=-.1234E4
5000000=5E6=5E+6
-.0000123=-1.23E-5=-123E-7
```

Introduceţi şi verificaţi afişarea aceluiaşi rezultat:

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

CONSTANTE ŞIR DE CARACTERE

Orice şir de caractere alfanumerice introduse între "" constituie o constantă şir.
Exemple:

```
"abc"
"NR.␣CRT"
"Nu opriti banda, se incarca programul!"
```

Dacă aveţi nevoie să folosiţi în şirul de caractere chiar caracterul " (ghilimele), îl introduceţi de două ori pentru fiecare apariţie în şir.

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

VARIABILE NUMERICE

Variabilele numerice pot fi formate din oricîte caractere alfanumerice, cu condiţia să înceapă cu o literă:

Exemple: a
p237
material58

Introduceţi:

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

În acest program aveţi:

-constante numerice: 6 şi 8
-constante şir: "suprafaţa="
-variabile numerice: lungimea, lăţimea, suprafaţa

VARIABILE ŞIR DE CARACTERE

Puteţi defini în program variabile cărora să le puteţi atribui diferite şiruri de caractere alfanumerice şi numerice.
Numele unei variabile şir trebuie să conţină o singură literă urmată de semnul $.
Exemple:

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

Un şir de caractere poate fi utilizat şi parţial sub formă de SUBŞIR. Manevrarea subşirurilor se face cu:

S=(n1 TO n2)
- S este o constantă şir sau o variabilă şir
- n1 primul caracter al subşirului
- n2 ultimul caracter al subşirului.

Fie şirul: a$="abcde"
Verificaţi manevrarea lui introducînd următoarele instrucţiuni cu execuţie imediată:

	Ce se afişează
CLS	
LET a$="abcde"	
PRINT 12345	12345
PRINT a$	abcde
PRINT a$(3)	c
PRINT a$ (2 TO 4)	bcd
PRINT a$ (␣TO 3)	abc
PRINT a$ (2 TO␣)	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 şirul are numai 5 caractere

Iată un exemplu mai sugestiv.
Dacă introducem într-un şir lunile anului prescurtate în trei litere astfel: l$="ian feb mar apr mai iun"
Putem extrage pentru un eventual calendar subşiruri.
Introduceţi şi verificaţi!

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

NOTAŢIA EXPONENŢIALĂ A NUMERELOR

Numerele foarte mari şi cele foarte mici pot fi exprimate sub o formă exponenţială astfel:

```
1234000000=1.234e+9
           ↑     ↑
   mantisa─┘     exponent (se poate nota cu "E" sau "e")
```

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

```
3.7e12= 3700000000000
        └──────────→ 12 poziţii
```

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

```
56e-8=0,00000056
8 poziţii<----┘
```

Exemplul următor vă va ajuta să înţelegeţi mai bine reprezentarea exponenţială:

```
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 afişa un rezultat sub formă exponenţială, dvs. veţi putea obţine forma zecimală obişnuită destul de uşor, luînd în considerare semnul exponentului. Dacă este + scrieţi punctul zecimal la dreapta cu atîtea poziţii cîte arată exponentul.

```
9.876543E8  este  987654300
                  └--------►8 poziţii
```

Dacă semnul este negativ, scrieţi punctul zecimal la stînga cu numărul indicat de poziţii.

```
23e-6  este 0,000023
   6 poziţii<----┘
```

OPERATORI RELAŢIONALI

Un program BASIC se execută "secvenţial" linie după linie, dar aveţi posibilitatea să-i spuneţi calculatorului, ca în anumite condiţii să nu mai respecte secvenţa crescătoare a numerelor de linii. Pentru aceasta veţi putea folosi instrucţiunea de testare condiţionată IF...THEN... (în traducere: DACĂ...ATUNCI...).

IF condiţie THEN acţiune

Cînd condiţia este îndeplinită se execută acţiunea indicată după THEN.
Condiţia (sau testul) care urmează după IF o puteţi pune folosind operatori relaţionali.

operator	semnificaţie	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 instrucţiunea IF veţi putea utiliza operatori logici:
AND (în traducere: ŞI)
OR (SAU)
NOT (NU)

AND

Folosind AND între două condiţii, impuneţi ca amîndouă să fie îndeplinite pentru ca să se execute ce urmează după THEN

IF condiţia 1 AND condiţia 2 THEN...

Pentru a înţelege mai bine, închipuiţi-vă o ţeavă prin care curge apa şi care are două robinete R1 şi R2:

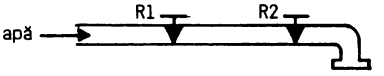


Fig.4.1

Ele pot fi în următoarele condiţii:

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

Condiţiile se pun deci astfel:

DACĂ R1=deschis ŞI R2=deschis ATUNCI apa curge

În BASIC veţi scrie:

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

OR

Operatorul logic OR îl veţi folosi atunci cînd din două condiţii este suficient una să fie îndeplinită, pentru ca să se execute ce urmează după THEN.
Închipuiţi-vă acum ţeava de apă ramificată astfel:

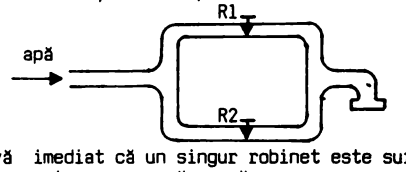


Fig.4.2

Se observă imediat că un singur robinet este suficient să fie deschis pentru ca apa să curgă.
Condiţiile se pun astfel:

DACĂ R1=deschis SAU R2=deschis ATUNCI apa curge

În BASIC veţi scrie:

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

NOT

Operatorul NOT se foloseşte înaintea unei condiţii 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 condiţie THEN.....

ALIPIREA (CONCATENAREA) ŞIRURILOR

Puteţi reunii două şiruri de caractere într-unul singur, folosind operatorul de concatenare (alipire)"+"

```
a$=b$+c$
```

Presupunînd că variabilele şir sînt:

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

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

se va afişa pe ecran: Întreprinderea ELECTRONICA

ÎNTREBĂRI RECAPITULATIVE ŞI EXERCIŢII

Î 4.1 În execuţia operaţiilor aritmetice există o ordine (prioritate). Scrieţi în ordine inversă ce operaţii se execută:

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

Î 4.2 Scrieţi forma BASIC pentru calculul expresiilor:

1) $d = \frac{a \times b}{c} + 10$

2) $e = \frac{(a+5)\,b^n}{2{,}7\,(c - \frac{}{d}) + 1}$

3) $y = ax^2+bx+c$

Î 4.3 Găsiţi 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 afişat 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 afişate următoarele numere:

a) 1200000000 _____
b) 0,000000444 _____

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

a) 3.001E+10 _____
b) 1.989E-6 _____

Î 4.7 Date fiind următoarele variabile: M,M$,Abc,P123,X$, şir,doi,S$ precizaţi care sînt:

a) variabile numerice:_____
b) variabile şir:_____

Î 4.8 Ca urmare a instrucţiunii:

```
IF a<=13 AND b>5 THEN PRINT x
```

se afişează x dacă:
1) a=13;b=13 _ _ _ (răspundeţi da/nu)
2) a=20;b=20 _ _ _ (da/nu)

Î 4.9 Dacă introduceţi:

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

după execuţia programului veţi primi răspunsul "sînt bucuros!"?

a) _ _ _ (da/nu)
b) De ce? _

Î 4.10 Atenţie o întrebare puţin mai dificilă!
Judecaţi cu atenţie şi încercaţi să răspundeţi, apoi introduceţi programul şi verificaţi-vă raţionamentul iniţial. 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 afişat 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 "introduceţi 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 "ROŞU":GOTO 20
 80 PRINT "ALB":GOTO 20
 90 PRINT "VERDE":GOTO 20
100 STOP
```

a)Ce se va afişa după execuţie dacă:
y=2 _ _ _ _ _ _ _ _ _ _ _ _ _ _
y=1 _ _ _ _ _ _ _ _ _ _ _ _ _ _
y=3 _ _ _ _ _ _ _ _ _ _ _ _ _ _
b)Care sînt în program constantele numerice?
c)Care sînt variabilele numerice?
d)Care sînt constantele şir?
e)Ce se întîmplă dacă introduceţi pentru y o valoare diferită de 1, 2 sau 3 ?

RĂSPUNSURI

R 4.1 4) adunarea/scăderea
3) înmulţirea/împărţirea
2) exponenţierea
1) expresiile din paranteze

R 4.2 1) d=(a*b)/c+10
2) e=((a+5)*b^n)/(2.7*(c-d/b)+1)
3) y=a*x^2+b*x+c

R 4.3 a) exponentul nu este număr întreg
b) virgula nu este admisă
c) exponent prea mare (>37)
d) exponent greşit (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$

R 4.8 1) da 2) nu

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

R 4.10 Linia 30 a paragrafului se citeşte aşa:

DACĂ NU este x diferit de y (adică x=y) afişează "AAAA" şi întoarce-te la linia 5.
În cazul în care condiţia pusă în linia 30 nu este îndeplinită (adică x este diferit de y) programul trece la linia 40 şi afişează mesajul corespunzător.
Deci răspunsurile corecte sînt:

a) BBBB
b) AAAA
c) BBBB

R 4.11 a) ALB
ROŞU
VERDE
b) 1, 2 şi 3
c) y
d) "ROŞU","ALB","VERDE"
e) execuţia programului este oprită prin STOP (linia 100)

INTRĂRI, IEŞIRI, APLICAŢII SIMPLE

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

În acest capitol cunoştinţele dvs. de BASIC vor fi îmbunătăţite, prin detalii asupra posibilităţilor de introducere a datelor, de obţinere a rezultatelor şi prin exemple de programe comentate.
Treptat, vă veţi însuşi noţiuni despre realizarea unor programe proprii bine gîndite şi clare, despre stocarea lor pe caseta magnetică şi despre o operaţie pe care o veţi executa foarte frecvent şi anume încărcarea unor jocuri/programe de pe casetele magnetice.

1. Anulaţi programul existent în memorie. Aveţi două posibilităţi:
-prin acţionarea tastei RESET
-prin instrucţiunea NEW

2. Introduceţi şi executaţi 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 şi cursorul clipeşte întrebător în colţul ecranului aşteptînd o valoare pentru variabila numerică "a".

a=2
b=4
c=6
Scrieţi ce se afişează:
_ _ _ _
_ _ _ _ _ _ _ _ _

Reţineţi că cele trei instrucţiuni INPUT au permis introducerea prin interogare, de valori pentru variabilele numerice a, b şi c.

3. Anulaţi liniile 10 şi 20 şi introduceţi din nou comanda de execuţie

```
10   <ENT>
20   <ENT>
RUN  <ENT>
c=9
```

Notaţi mesajul afişat: _ _ _ _ _ _ _ _ _ _

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

4. Schimbaţi linia 30 punînd toate variabilele, separate prin virgule:

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

Aveţi 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 clipeşte cerînd valoarea lui "a", după care sare la mijlocul liniei şi cere pe "b" şi apoi sare în linia următoare cerînd pe "c".

a=10 b=4.3
c=5.7
Rezultat: _ _ _ _ _ _ _

5. Introduceţi:

```
30 READ a,b,c (READ obţineţi cu SS+CS şi apoi A)
LIST
```

Veţi avea pe ecran:

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

Notaţi mesajul afişat: _ _ _ _ _ _ _ _ _ _ _ _ _ _

Explicaţia este următoarea: instrucţiunea READ este o altă posibilitate de a introduce date, direct în program sub formă de constante numerice fixe.
Deocamdată neavînd aceste date, execuţia liniei 30 READ... a generat mesajul de mai sus, care vă atrage atenţia că nu mai sînt date.

6. Acum introduceţi:

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

Avînd datele cunoscute, CIP-ul execută fără ezitare adunarea şi va comunica rezultatul..

7. Anulaţi linia 35 şi introduceţi 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
```

Observaţi că deşi linia DATA a fost aplicată în altă parte în program (chiar şi înaintea instrucţiunii READ) BASIC-S acceptă datele şi execută programul.

8. Anulaţi programul anterior şi introduceţi:

```
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
```

Notaţi ce se afişează: _ _ _
_ _ _
_ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Instrucţiunea READ a "citit" cîte două valori din DATA, în linia 20 a făcut împărţirea, linia 30 a determinat afişarea rezultatului şi totul a fost reluat cu altă pereche de date pînă cînd acestea au fost epuizate.

9. Anulaţi linia 50 şi introduceţi:

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

Se afişează: 6
2
7
Out of DATA,10:1

Puteţi pune deci mai multe instrucţiuni DATA într-un program? (DA/NU) _ _ _ _ probabil că aţi răspuns corect: da! pot fi mai multe instrucţiuni DATA amplasate oriunde în program.
10. Anulaţi programul şi introduceţi:

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

Notaţi rezultatul:_ _ _ _ _ _

11. Schimbaţi linia 110:

```
110 PRINT "G"
RUN
```

Ce se afişează acum ca rezultat? _ _ _ _ _ _ _

De reţinut:
-PRINT constantă numerică -afişează valoarea constantei
-PRINT "constantă şir" -afişează conţinutul constantei şir

12. Dacă introduceţi:

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

Ce credeţi că se va afişa după execuţie? _ _ _ _ _ _ _ _ _
Executaţi programul şi verificaţi răspunsul dvs.

13. Anulaţi programul anterior şi introduceţi:

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

Executaţi programul şi notaţi rezultatul:_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _

14. Schimbaţi:

```
20 PRINT x,
```

Observaţi că apare o virgulă după x. Executaţi programul şi notaţi din nou rezultatul:

_ _ _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _

15. Mai schimbaţi linia 20 astfel:

```
20 PRINT x;    (; după x)
RUN
```

Notaţi modul de afişare:_ _ _ _ _ _ _ _

Reţineţi deci modul de afişare diferit după cum este încheiată instrucţiunea PRINT:

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

16. Adăugaţi şi schimbaţi în program:

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

.Linia 5 a numerotat cele 32 (0-31) poziţii ale unei linii de afişare;
.În linia 20 s-a introdus TAB 3, care impune afişarea rezultatelor începînd cu poziţia 3.

Obsevaţi pe ecran obţinerea celor de mai sus.

17. Să schimbăm poziţia de afişare şi datele:

```
20 PRINT TAB 15;x
40 DATA 8,100,3210,9.6
```

Rezultatele sînt afişate începînd cu coloana 15.

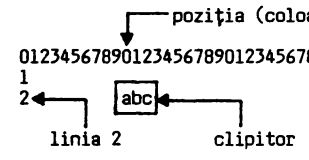
18. Adăugaţi şi schimbaţi în program:

```
8  PRINT 1:PRINT 2 ←numerotează la margine linia 1
                     şi linia 2
10 READ x$
20 PRINT AT 2,10;FLASH1;x$
                    └►afişează clipitor
►afişează începînd cu linia 2,poziţia (coloana) 10
30 DATA "abc"
40 <ENT> - pentru a şterge 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"
```

Executînd programul veţi obţine următorul rezultat:



Noţiuni de bază

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

LET

Instrucţiunea LET atribuie unei variabile "v", valoarea unei expresii "e" Sintaxa comenzii: LET v=e

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

Exemple de instrucţiuni LET:
```
LET d=14.3
LET L$ = "APRILIE"
LET x=(p+m)/n
LET k=k+1
```

Ultimul exemplu pare imposibil dar îl puteţi înţelege analizînd următorul program:

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

După execuţie se afişează: 5678910...
Iniţial k ia valoarea 5 care este afişată. La execuţia liniei 30, valoarea anterioară este crescută cu 1, astfel încît a doua oară va fi afişată valoarea 6 etc.

INPUT

Instrucţiunea INPUT (în traducere: introduceţi) permite introducerea datelor direct de la tastatură.

Sintaxa: INPUT "c";v1;v2
.c este un şir opţional de caractere care explicitează datele;
.v1,v2... sînt nume de variabile numerice/şir de caractere.

Spre deosebire de LET, instrucţiunea INPUT permite deci introducerea conversaţională a datelor. Cînd CIP-ul o întîlneşte, opreşte execuţia programului şi cursorul clipeşte întrebător pînă cînd dvs. răspundeţi prin:

a) introducerea datelor solicitate, după care execuţia programului continuă
b) introducerea instrucţiunii STOP în vederea renunţării la execuţie.

Exemple:
```
INPUT  x  - introduce pentru variabila  x,  valoarea
            tastată
INPUT "a=?";a - afişează a=? şi introduce valoarea
                tastată;
INPUT a;b;c -solicită pe rînd valori pentru a,b şi c
```

Puteţi deci cere introducerea datelor pentru mai multe variabile printr-o singură instrucţiune INPUT.
Variabilele pot fi separate prin punct şi virgulă sau prin virgulă:
```
INPUT  m;n  - cere date pentru m şi cursorul rămîne
              în aceiaşi poziţie.
INPUT  m,n - cere date pentru m şi apoi sare în
             coloana 15 pentru n
```

READ

Instrucţiunea READ (în traducere: citeşte) atribuie unor variabile v1, v2, ... valoarea unor expresii/date succesive din listele instrucţiunilor DATA. Sintaxa: READ v1, v2...

DATA

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

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

Pentru a înţelege mai bine ce efect au cele două instrucţiuni, încercaţi exemplul de mai sus astfel:

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

Instrucţiunea READ diferă de INPUT prin faptul că programul nu se opreşte să aştepte ca datele să fie introduse prin tastatură, ci le caută automat în instrucţiunea DATA.
Instrucţiunea DATA poate fi plasată oriunde în program, chiar şi înaintea instrucţiunii READ.
Se obişnuieşte ca toate instrucţiunile DATA să fie puse la sfîrşitul programului, înaintea instrucţiunii STOP.
Pot fi mai multe instrucţiuni DATA şi nu este necesar să fie grupate în program. Instrucţiunea READ citeşte pe rînd valori pentru variabilele specificate, din DATA începînd cu numărul de linie cel mai mic. Dacă numărul de variabile depăşeşte pe cel al datelor se afişează mesajul: Out of DATA şi programul este oprit.

RESTORE

Instrucţiunea RESTORE (în traducere: restabileşte!) permite folosirea aceluiaşi set de date, de către mai multe instrucţiuni READ. Sintaxa: RESTORE n
Un indicator de citire este restabilit la instrucţiunea DATA cu număr de linie n, astfel încît următoarea instrucţiune READ va începe citirea de la acea linie.
Exemplu:
```
10 READ a,b,c - citeşte valori din liniile 50 şi 60
20 RESTORE  50 - restabileşte ca linie de citire
                 linia 50
30 READ z,x - va citi din nou linia 50
40  PRINT  a'b'c'z'x'  (introduceţi apostrof între
                        variabile)
50 DATA 1,2
60 DATA 3,4
70 DATA 5,6 ← de aici nu citeşte
80 STOP
RUN
```

```
Rezultat:      1
               2
               3
               1
               2
```

Notă: Separarea variabilelor din instrucţiunea PRINT cu apostrof, are ca efect afişarea fiecăreia într-o linie nouă.

PRINT

Afişează pe ecran constante, variabile sau expresii calculate.

Sintaxa:

```
PRINT n        - afişează valoarea constantei numerice
PRINT "şir"    - afişează şirul dintre ghilimele
PRINT x;y;a$   - afişează x, y şi a$ în aceeaşi linie
                 (fără spaţii între ele)
PRINT x,y,a$   - afişează alternativ începînd din
                 coloanele 0 şi 16
PRINT x'y'a$   - afişează fiecare valoare în linie nouă
```

Se pot face calcule cu ajutorul instrucţiunii PRINT. De exemplu:

```
PRINT x*y,z
```

va afişa din coloana 0 rezultatul înmulţirii dintre x şi y din coloana 16 valoarea lui z.

Încercaţi:

```
NEW
10 LET x=5:LET y=6:LET z=7
20 PRINT "5*6=";x*y,z
RUN
```

Notaţi ce se afişează: _ _ _ _ _ _ _ _ _ _

Schimbaţi şi adăugaţi liniile:

```
20 PRINT x,y
25 PRINT <------- afişează o linie vidă
30 PRINT x;y
35 PRINT
40 PRINT x'y
RUN

5       6 <---  separare cu virgulă
56        <---  separare cu punct şi virgulă
5}        <---  separare cu apostrof
6J
```

PRINT AT l,c

Forma PRINT AT ... (în traducere afişează la ...) o puteţi folosi cînd vreţi să afişaţi ceva într-o anume zonă a ecranului. Afişarea începe cu poziţia dată de:

```
l (0-21) - număr linie
c (0-31) - număr coloană
```

Încercaţi să afişaţi în mijlocul ecranului un mesaj:

```
PRINT AT 11,9;"OPRIŢI BANDA"
          ↑   ↑
linia 11──┘   └──începînd din coloana 9
```

Afişarea pe ecran poate fi programată în două moduri:
-grafică - pentru desene (va fi explicată în capitolul 9)
-alfanumerică - pentru caractere

Macheta ecranului

În afişarea alfanumerică ecranul este împărţit în 24 linii orizontale şi 32 coloane (Fig.5.1).
Afişarea unui caracter poate fi poziţionată oriunde pe ecran, prin două coordonate: număr linie, număr coloană.
Liniile 22 şi 23 sînt folosite pentru editarea liniilor noi de program sau a celor în care doriţi să faceţi modificări.
Încercaţi următoarea linie multiplă PRINT care va produce un dreptunghi în mijlocul ecranului, în care va fi scris şirul "CIP":

tastaţi:CS+9 şi CS+8 retastaţi CS+9 şi apoi SS+P

```
10 PRINT AT 7,11;"■■■■■■■■■";AT 8,11;"■ ■■■■■■■ ■ ";
   AT 9,11;"■ ■ CIP ■ ■";AT 10,11;"■ ■■■■■■■ ■";AT 11,
   11; "■■■■■■■■■"
```

Caracterul ■ îl obţineţi cu (CS+9) CS+8 (Anexa B).

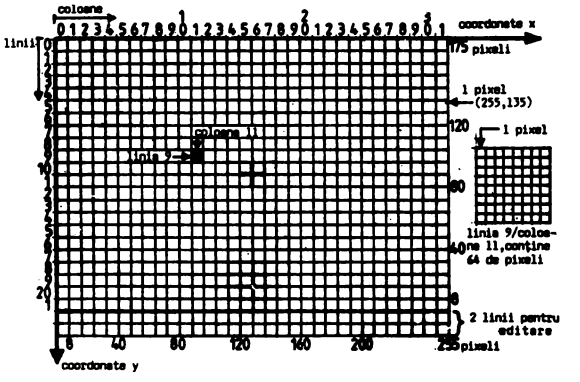


Fig. 5.1.

PRINT TAB

Poziţionarea pe liniile ecranului se mai poate obţine şi cu forma: PRINT TAB c
unde c reprezintă coloana din care să înceapă afişarea. Dacă c este mai mare decît 31 afişarea se face în continuare pe linia următoare.

Introduceţi şi executaţi:

```
NEW
10 PRINT "01234567890123456789012345678901"
20 PRINT TAB 5;" ↑ ";TAB 8;" ↑ ";TAB 25;" ↑ ";TAB 32;
   "*";TAB 35;"*"
RUN
```

REM

Instrucţiunea REM (prescurtare de la REMARK) permite introducerea în program a unor comentarii utile celui care analizează un program. REM nu este luată în considerare la execuţia programului, dar apare ori de cîte ori programul este listat, uşurînd foarte mult înţelegerea lui.
Este bine să vă obişnuiţi să introduceţi asemenea comentarii în programele dvs.
Pentru a arăta utilizarea instrucţiunii REM şi avantajele ei, comparaţi cele două exemple care urmează. Ambele programe produc aceleaşi rezultate, dar unul din ele este "comentat"

```
 5 LET w=0
10 INPUT x,y,z
20 LET w=(x+y+z)/3
30 PRINT w
40 STOP

10 REM nume program:PRG
20 REM calculează media aritmetică
30 REM introducerea numerelor
40 INPUT x,y,z
50 REM calculul mediei
60 LET w=(x+y+z)/3
70 REM afişarea rezultatului
80 PRINT w
90 STOP
```

CAPITOLUL 5

EXEMPLE DE PROGRAME

Stimate cititor, consider că sînteţi suficient de pregătit să începeţi să analizaţi şi să scrieţi programe din ce în ce mai complicate.
În acest capitol vom analiza cîteva exemple, vom enunţa altele pe care să le realizaţi singuri.
Curaj! veţi constata că aţi reţinut mai multe cunoştinţe decît vă închipuiţi.
Mai întîi să recapitulăm:

- -un program este format din instrucţiuni
- -o linie de program poate conţine una sau mai multe instrucţiuni separate prin ":"
- -fiecare linie trebuie numerotată
- -instrucţiunea REM se foloseşte pentru comentarea (documentarea) programului
- -pentru introducerea datelor se folosesc instrucţiunile:
  .LET
  .INPUT
  .READ...DATA (RESTORE)
- -pentru calcul se folosesc instrucţiunile:
  .LET
  .PRINT
- -pentru afişarea rezultatelor se folosesc instrucţiunile:
  .PRINT
  .PRINT AT
  .PRINT TAB

Iar acum cîteva sugestii, pentru ca să realizaţi programe bune şi... frumoase. Este indicat să urmaţi următoarele faze:

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

Ştiu că sînteţi nerăbdător să faceţi cît mai repede programe, dar nu săriţi rîndurile următoare prin care vă semnalez cîteva greşeli făcute de începători. Unele greşeli sînt simple, cele făcute chiar la introducere (tastare greşită, erori de sintaxă) şi vrînd nevrînd va trebui să le corectaţi pentru că CIP-ul vigilent nu le acceptă.
Sînt însă erori mai subtile, de logică, şi acestea sînt sesizabile doar prin obţinerea unor rezultate necorespunzătoare. Pentru ele trebuie să faceţi o "depanare" a programului, acţiune despre care vom discuta în alt capitol.
Pentru evitarea greşelilor de introducere vă recomand:

- -respectaţi sintaxa cerută de BASIC
- -tastaţi <ENT> la sfîrşitul fiecărei linii de program
- -nu tastaţi litera O în loc de cifra Ø (zero)
- -fiţi atent la numărul de linie, să nu îl repetaţi pe cel al uneia existente, pentru că aceasta va dispare
- -anulaţi 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 şi 16.

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

Ieşiri: x= valoare lui x.

Se doreşte deci nu numai afişarea valorii rezultatului, ci a unei forme explicite: x= rezultat

Algoritmul problemei:
1.solicitarea şi memorarea datelor de intrare
2.calculul expresiei
3.afişarea 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 afişarea rezultatului
80 PRINT "x=";x
90 STOP
```

Două precizări:
1. - la expresii cu paranteze trebuie să verificaţi ca numărul parantezelor deschise "(" să fie egal cu cel al parantezelor închise ")".
2. - dacă vă aflaţi în execuţia unei instrucţiuni INPUT şi dintr-un motiv oarecare doriţi să întrerupeţi introducerea, singura posibilitate este de a şterge înapoi cu DELETE pînă se opreşte cursorul şi să daţi STOP <ENT>.

Introduceţi programul, notaţi pe o hîrtie datele introduse şi rezultatul: x=_ _ _ _ _ _

Schimbaţi linia 60 : 60 LET x=(2*(a+b)↑2+3*c↑2)↑(m/(n+1))

Executaţi programul cu aceleaşi date şi notaţi din nou rezultatul: x=_ _ _ _ _ _

Observaţi că rezultatele diferă deşi nu v-a fost semnalată nici o eroare la introducere. De ce diferă rezultatele?.Care este cel bun? (nu calculaţi cu creionul, ci analizaţi programul)

Exemplul 5.2.

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

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

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

- Prelucrări:
TOTAL = 0 - valoarea iniţială a variabilei TOTAL
TOTAL = TOTAL + c - se însumează totalul anterior cu cantitatea nou introdusă

- Codificarea programului:

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

Ce s-a întîmplat?
A fost afişat capul de tabel, dar sînteţi avertizaţi că nu a fost găsită variabila în linia 90. Deoarece variabila c tocmai aţi introdus-o rezultă că nu a fost găsită variabila total.

CIP-ul "v-a prins" cu o greşeală de logică. I-aţi cerut ceva pe care aţi uitat să i-o definiţi mai înainte.
Corectaţi greşeala introducînd ceea ce de fapt aţi prevăzut la prelucrări: decalarea şi iniţializarea variabilei "TOTAL".

```
25 REM iniţializarea variabilei de totalizare
28 LET total = Ø
RUN
```

Acum programul se execută. Introduceţi cîteva cantităţi şi verificaţi:

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 afişat cu 8 cifre:

2.0000011955 --------► 2.0000012

SALVAREA PROGRAMELOR PE CASETĂ

Deoarece în continuare vă voi propune să realizaţi singuri programe pe "teme" date, veţi dori eventual să le păstraţi şi după oprirea calculatorului ca să le revedeţi altădată.
Pentru aceasta va fi nevoie să le salvaţi pe o casetă magnetică (memoria externă a CIP-ului), de unde să le încărcaţi din nou în memorie oricînd doriţi.
Pregătiţi deci casetofonul, o casetă (atenţie, ce veţi înregistra va şterge ceea ce aveţi deja pe casetă).
Puneţi cablu de legătură între CIP şi casetofon.

Înregistrarea programelor pe casetă

Cînd doriţi să salvaţi un program procedaţi astfel:
a) - fiind în mod comanda (cursor K) introduceţi comanda:

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

Exemplu:

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

Obţineţi mesajul: Start tape, than press any key
adică: Porniţi banda, apoi apăsaţi orice tastă
b) - fără să porniţi banda apăsaţi orice tastă.
L-aţi "păcălit" pe CIP, el începe transmiterea programului către casetofon şi puteţi astfel regla nivelul de înregistrare. Acesta este bine să fie cît mai mare, chiar peste limita recomandată pentru muzică.
c) - mai introduceţi odată comanda

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

şi acum ascultaţi-i sfatul, pornind întîi banda şi după cîteva secunde (pentru a se crea un spaţiu pe bandă între programe) apăsaţi orice tastă. Dacă totul merge normal obţineţi confirmarea prin OK (totul în ordine).

Un program salvat numai cu:

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

la încărcarea de pe casetă va fi lansat în execuţie prin comanda RUN, spre deosebire de cazul în care introduceţi în comanda SAVE numărul primei linii din program (ca în exemplul de mai sus), cînd programul va fi lansat automat în execuţie.
Alte forme ale comenzii SAVE le găsiţi în anexa C.

VERIFY

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

a) - derulaţi banda la începutul înregistrării
b) - introduceţi comanda: VERIFY "nume program" <ENT>
c) - porniţi banda
Dacă programul a fost înregistrat corect, pe ecran apare: Program nume... şi la sfîrşitul verificării se afişează
Ø OK

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

Încărcarea programelor de pe casetă

Aşa cum v-am mai 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 veţi fi frecvent în situaţia de a avea nişte jocuri sau programe formidabile pe o casetă şi veţi fi nerăbdători să le executaţi pe CIP-ul dvs.
Pentru încercarea unui program de pe casetă procedaţi astfel:
a) - introduceţi
LOAD " " - dacă doriţi să fie încărcat primul program care urmează pe bandă
LOAD "nume program" - pentru a fi căutat şi încărcat programul cu numele specificat în comandă
b) - porniţi banda
Dacă totul este în ordine, pe ecran se afişează numele programului şi urmează dungile clipitoare însoţite de un sunet specific.
c) - opriţi banda la apariţia mesajului "OK" sau a celui dat de program.
Dacă după pornirea benzii nu apar numele programului şi dungile orizontale, sau dacă apare mesajul: "Tape loading error" opriţi banda, derulaţi-o rapid înapoi şi reluaţi secvenţa de încărcare. După 2-3 încercări nereuşite, puteţi renunţa 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 zgîriată.

O ultimă soluţie pe care o puteţi încerca este aceea de a utiliza alt casetofon.

Întrebări recapitulative şi exerciţii

Î 5.1 Ce se va afişa dacă se execută următorul program?

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

Scrieţi primele 12 valori rezultate şi apoi executaţi programul pentru a vă verifica (întrerupeţi cu BREAK).

Î 5.2 Care este scopul instrucţiunii REM?

Î 5.3 Dacă introduceţi într-un program instrucţiunea READ, ce altă instrucţiune trebuie să utilizaţi obligatoriu?

Î 5.4 Ce va fi afişat dacă se execută următorul program?

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

Î 5.5 Completaţi valorile lipsă:
În afişarea alfanumerică ecranul este considerat împărţit în _ _ _ coloane şi _ _ _ linii.

Î 5.6 Ce instrucţiune folosiţi pentru a afişa începînd din diferite coloane?

CAPITOLUL 5

Î 5.7 Diferă afişarea rezultatelor prin liniile 30, 40 şi 50 din următorul program?

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

Î 5.8 Ce tastaţi pentru a afişa programul dacă CIP-ul aşteaptă un răspuns după o instrucţiune INPUT?

Î 5.9 Ce instrucţiune permite utilizarea aceluiaşi set de date, de către mai multe instrucţiuni READ?

Î 5.10 Scrieţi în format BASIC expresiile:

a) $2\,(X_1 + 3X_2)$

b) $X = \dfrac{5{,}4 + y\sqrt{u+y}}{(a+b)^{2/y}}$

Î 5.11 Scrieţi instrucţiunile BASIC pentru:
a) a creşte c7 cu 0,01
b) a da valoarea "**eroare**" variabilelor T$ şi U$.

Î 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) Scrieţi un program pentru calculul suprafeţei (S) şi volumului (V) al unei sfere cu raza r. Rezultatele să fie afişate astfel:

r=____ S=____ V=____

2) Scrieţi un program pentru calculul volumului unei cutii paralelipipedice având lungimea L, lăţimea l şi înălţimea h. Datele de intrare să fie cerute astfel:

```
Lungimea (cm)?
Lăţimea (cm)?
Înălţimea (cm)?
```

Rezultatele să fie afişate 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 Instrucţiunea REM permite introducerea unor comentarii care să ajute la înţelegerea 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 afişarea 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*(x1 + 3*x2)
b) x=(5.4 + y*(u+v)↑0,5)/(a+b)↑2/y

R 5.11
a) LET C7 = C7 + 0,1
b) LET T$ = "**eroare**"
LET U$ = "**eroare**"

R 5.12
- în linia 20 lipseşte LET
- linia 30 trebuie să devină 50 pentru că p nu este cunoscut
- corect se scrie: LET p = 2*(l+h)

CUM INTRODUCEM CONDIŢII SAU ALTERNATIVE ÎN PROGRAM ?

Adeseori veţi simţi nevoia în programe să determinaţi CIP-ul să nu mai execute instrucţiunea care urmează, ci să execute un salt la o instrucţiune aflată peste un număr de linii înainte sau înapoi.
Un asemenea salt poate fi introdus în program, să fie executat în anumite condiţii sau necondiţionat.

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 sînt introduse de autorul programului/jocului sub forma unor alternative condiţionate sau nu.

1.Introduceţi 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 citiţi astfel: "Dacă a este mai mic decît 8 ATUNCI MERGI (sari) la linia 120".
Analizaţi cu atenţie programul şi scrieţi pe o hîrtie ce credeţi că va fi afişat.
Executaţi programul şi verificaţi-vă.

2.Schimbaţi linia 110: 110 LET a=6
Ce se va obţine?. Executaţi programul şi verificaţi!

3.Schimbaţi şi introduceţi:

```
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 decît sau egal cu 46 ATUNCI MERGI la linia 150".
CIP-ul ascultător aşa va face, dar cît timp a este mai mic decît 46 va executa linia următore (145) care îl trimite înapoi la linia 120.
Analizaţi programul şi rezultatele pentru a înţelege ce s-a întîmplat.

4.Completaţi semnele prin care veţi pune în program condiţia 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.Analizaţi cu atenţie următorul program:

```
10 PRINT "introduceţi 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 "SCUFIŢA ROŞIE": GO TO 20
80 PRINT "HARAP ALB": GO TO 20
90 PRINT "PINOCHIO": GO TO 20
100 STOP
```

Ce valoare trebuie să introduceţi pentru ca să se afişeze:

```
HARAP ALB        n=_ _ _ _
PINOCHIO         n=_ _ _ _
SCUFIŢA ROŞIE    n=_ _ _ _
```

Anulaţi vechiul program, introduceţi pe cel de mai sus şi executaţi-l cu valorile anticipate de dvs. verificîndu-vă!

6.Programul cere să se introducă pentru n valorile 2, 4, 6. Ce credeţi că se va întîmpla dacă introduceţi altă valoare? Formulaţi răspunsul şi verificaţi!

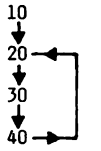
NOŢIUNI DE BAZĂ

GO TO Instrucţiunea GO TO produce un salt necondiţionat 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 şi continuă linie cu linie. Instrucţiunea GO TO (în traducere: "mergi la") permite întreruperea acestei secvenţe printr-un transfer /salt la o linie de program specificată.
Programul de mai sus se execută în următoarea ordine a liniilor:



"bucla" sau "ciclu" de program

Deoarece nu este pusă nici o condiţie în linia 40, GO TO constituie o instrucţiune de salt necondiţionat.
Linia 40 produce saltul execuţiei înapoi la linia 20, astfel că linia 50 de oprire a programului nu ar fi executată niciodată.
Întrucît programul nu se opreşte singur, se spune că "a intrat într-o buclă fără sfîrşit" sau "ciclează".
Întreruperea execuţiei se poate face după cum deja ştiţi prin:

- STOP - la întrebarea dată de INPUT
- BREAK

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

Dacă expresia x este adevărată (condiţia este îndeplinită) se execută s, în caz contrar se execută instrucţiunea care urmează după IF.

Exemple:

```
IF j=3 THEN i=i+1:GO TO...
IF a>0 AND a<10 THEN GO TO (3+a*100)
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 execuţie calculatorul ajunge să verifice dacă:

```
4 * (1 + 5)  <  (50 - 25)
24 < 25
```

În acest caz condiţia este îndeplinită şi se va executa ceea ce urmează după THEN. Dacă după calculul expresiilor rezultă o condiţie neîndeplinită, s-ar executa instrucţiunea cu numărul de linie imediat superior după IF.
Instrucţiunea de transfer condiţionat IF THEN este deosebit de utilă, dînd posibilitatea calculatorului să sară în program oriunde doriţi.
După cum poate aţi remarcat, programele luate de exemplu pînă acum, au avut o mare deficienţă: atunci cînd includeau cicluri de repetare a unui număr de linii, nu există altă cale de a fi oprite decît prin STOP/BREAK/RESET introdus de dvs.
Saltul condiţionat (cu IF...THEN) permite o soluţie mai elegantă de a stopa un program în anumite condiţii.

CAPITOLUL 6

EXEMPLE DE PROGRAME

Exemplul 6.1

Să se realizeze un program care să afişeze începînd din coloanele 0 şi 16, numerele:

```
 3      6
 9     12
15     18
21     24
```

După ce ultimul număr (24) a fost afişat, execuţia să se oprească automat.
Analizînd puţin problema dată, observăm că fiecare număr este mai mare cu 3 decît cel anterior.
Programul poate fi realizat în mai multe variante. Cea mai puţin elegantă este:

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

Poate unii zîmbiţi cu superioritate, dar reţineţi 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 soluţie prea bună pentru că cere multă atenţie la introducere şi nu permite uşor o eventuală schimbare a numerelor, sau a spaţiilor de afişare.

O altă soluţie 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 afişarea în coloanele 0 şi 16, iar linia 40 a condiţionat stoparea programului.

O a treia soluţie se obţine schimbînd:

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

Introduceţi şi executaţi fiecare variantă analizînd apoi cu atenţie programele şi 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 şti care este), să introducem un număr - indicator care să semnaleze calculatorului că este ultimul, şi să se oprească programul.
Să alegem ca indicator valoarea "0" pentru că tot nu ar influenţa media ce se calculează.
Trebuie deci pusă în program condiţia ca atunci cînd este citită valoarea 0 programul să se oprească.

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

$$m = \frac{s}{n}$$

Codificarea programului:

```
 5 REM nume program: MEDAR
10 REM iniţializare s şi n
20 LET s=0: LET n=0
30 REM citire date
40 READ x
50 REM testare sfîrşit 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 afişarea 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 amănunt: la scrierea liniei 60 se lasă necompletat numărul liniei la care să se facă saltul (GO TO ...) pînă cînd se ştie care este acea linie.
Se pot pune în program atîtea instrucţiuni DATA cîte sînt necesare pentru a cuprinde toate numerele, avînd grijă să introducem 0 după ultimul număr.

ÎNTREBĂRI RECAPITULATIVE ŞI EXERCIŢII

Î 6.1 Ce se va afişa după execuţia 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
```

Î 6.2 Indicaţi greşelile din următorele instrucţiuni:

a) IF D3 >= 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

Î 6.3 Scrieţi instrucţiunile pentru următoarele condiţii:

a) dacă x < 3 afişează "negru", în caz contrar afişează "alb"
b) dacă şirul A$ este "da" salt la 90, dacă nu, salt la 200
c) dacă t=0 salt la 110, dacă nu, creşte x cu i şi salt la 40
d) dacă x este cuprins între 1 şi 4, salt la linia (1000 + x)

Probleme

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

10, să se afişeze diferenţa dintre cel mai mare şi 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 şi GO TO inversate şi 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>=1 AND x<=4 THEN GO TO (1000 + x)
```


CUM SE POT REPETA PĂRŢI DIN PROGRAM

Veţi învăţa în continuare o nouă şi 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 repetiţii/cicluri, folosind instrucţiuni de transfer condiţionat sau necondiţionat. BASIC are însă şi instrucţiuni speciale care să înceapă, să execute şi să termine automat cicluri în program.

APLICAŢIE PRACTICĂ AP 7

1. Introduceţi:

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

Notaţi rezultatele execuţiei:

_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _

2. Tastaţi:

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

Remarcaţi că se obţin aceleaşi rezultate.

3. Schimbaţi linia 10:

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

Analizaţi rezultatele! Observaţi că sînt afişate din 2 în 2, adică cifra pusă la cuvîntul STEP (în traducere: PAS)

4. Schimbaţi din nou:

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

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

5. Tastaţi:

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

Notaţi rezultatele:

_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _

6. Încercaţi acum :

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

Notaţi mesajul obţinut şi încercaţi să-l explicaţi:

_ _

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

Introduceţi:

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
```

Notaţi rezultatele şi să încercăm acum să interpretăm puţin programul:

- linia 10 afişează numele variabilelor
- linia 20 introduce o linie vidă
- linia 30 se citeşte: "pentru a egal cu 1 la 3" şi deschide un ciclu pentru variabila a
- linia 40 se citeşte: "pentru b egal cu 2 la 5" şi deschide un ciclu pentru variabila b
- linia 60 se traduce: "următoarea valoare pentru b" şi comandă reluarea ciclului variabilei b, cu linia 40. Pentru a=1, b ia succesiv valorile 2, 3, 4, 5 (priviţi la rezultate)
- linia 70 produce salt la linia 30 şi trecerea variabilei a la valoarea următoare: a=2 reluîndu-se ciclul variabilei b. Acelaşi lucru se întîmplă şi pentru a=3 după care programul este stopat de linia 80.

8. Modificaţi:

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

Încercaţi să scrieţi ce se va obţine şi apoi executaţi programul şi confruntaţi rezultatele cu cele anticipate.

NOŢIUNI DE BAZĂ

```
FOR
.
.
.
NEXT
```

Instrucţiunile FOR...NEXT permit execuţia instrucţiunilor 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 sînt numere sau expresii aritmetice.

Exemple:

a)
```
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)
```

b)
```
FOR i=a TO a+b STEP d
.
.
NEXT i
```

c)
```
FOR x = (abc - 2) TO 7 STEP -1
FOR j = 1 TO i
.
.
NEXT j : NEXT x
```

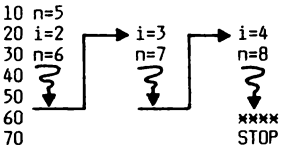
d)
```
FOR i = 0 TO INT (6.9 * RND)
.
.
NEXT i
```

Să vedem puţin mai în detaliu cum lucrează instrucţiunile FOR...NEXT

Introduceţi:

```
NEW
10 LET n=5
20 FOR i = 2 TO 4
30 LET n = n+1        } Se repetă pentru
40 PRINT i, n         } i=2 i=3 i=4
50 NEXT i
60 PRINT : PRINT "****"
70 STOP
RUN
```

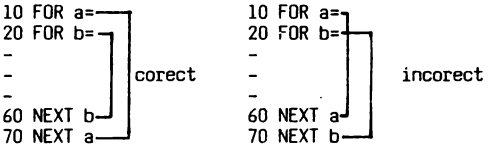
Iată rezultatele după execuţia fiecărei linii:



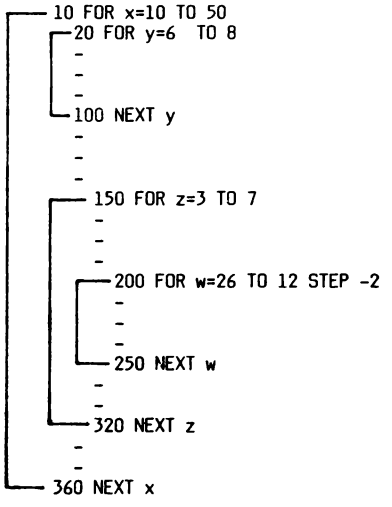
La început n primeşte valoarea 5, apoi calculatorul intră în prima execuţie a ciclului, variabila i luînd valoarea 2. Se execută liniile 30 şi 40 cuprinse în ciclu şi linia 50 care produce "saltul" execuţiei înapoi la linia 20 şi i devine 3. Bucla se repetă pînă cînd i atinge valoarea limită 4 cu care calculatorul iese din ciclu şi se execută instrucţiunea imediat următoare, adică linia 60.
Clauza STEP±s determină creşterea, respectiv scăderea variabilei cu "s" la fiecare execuţie 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 puteţi introduce oricîte cicluri FOR...NEXT, respectînd următoarele condiţii:
- un ciclu deschis cu FOR trebuie închis neapărat cu NEXT, deoarece în caz contrar execuţia are loc o singură dată, pentru prima valoare a lui x
- buclele să nu se intersecteze



Iată alt exemplu de bucle multiple organizate corect:



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
```

Introduceţi programele (aveţi grijă! NEW înainte de al doilea) şi executaţi-le verificînd 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) şi programul să calculeze media lor.
Afişarea numerelor să se facă la marginea ecranului, unul sub altul, şi la sfîrşit sub o linie de total să se afişeze 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: MEDAR1
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
```

Executaţi 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 sînteţi atît de bine pregătit, încît puteţi descifra conţinutul memoriei calculatorului. Despre lucrul intim cu memoria principală veţi afla în capitolul 12. Acum anticipăm însă puţin cu cîteva oţiuni.
Memoria principală o putem imagina ca un dulap cu multe sertare numerotate unul după altul, începînd cu sertarul avînd numărul 0 şi continuînd din 1 în 1 pînă la sfîrşitul memoriei.
Într-un asemenea "sertar" calculatorul memorează un caracter şi numărul sertarului reprezintă o ADRESĂ.
Caracterele, vă amintiţi, sînt reprezentate prin codurile cuprinse în anexa A.
Dacă introduceţi o linie program:

```
PRINT "x+y=";5
```

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

adresa	conţinut	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 instrucţiune care citeşte conţinutul memoriei la orice adresă dorim şi îl afişează pe ecran:

```
PEEK a      unde "a" reprezintă adresa
```

Iată un program care vă afişează adresele de mai sus şi conţinutul lor:

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

Executaţi programul începînd cu linia 100:

```
160 GO TO 100
```

Analizaţi rezultatele, aşa-i că este interesant!

ÎNTREBĂRI RECAPITULATIVE ŞI EXERCIŢII

Î 7.1 Ce se va afişa după execuţia următoarelor programe:

a)
```
10 FOR a=20 TO 0 STEEP -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 Indicaţi greşelile din următoarele linii:

a) FOR a ↑ 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 Scrieţi buclele FOR...NEXT pentru:

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

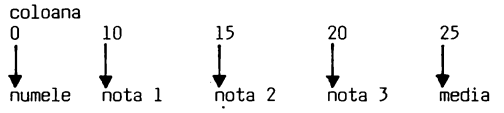
CAPITOLUL 7

Probleme

1. Scrieţi un program pentru calculul şi afişarea următoarelor numere ridicate la cub:

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

2. Scrieţi un program care să citească în DATA numele a 5 elevi, notele obţinute de fiecare la 3 materii şi să se calculeze media. Afişarea să se facă în următorul format:



Indicaţie: utilizaţi 5 enunţuri DATA sub forma: DATA "ANDREI";7,9,10 numele fiind de maximum 10 caractere.

3. Scrieţi un program pentru numerotarea coloanelor şi 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 şir
c) variabila de la FOR nu se poate pune la STEP
d) variabile diferite la FOR şi 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 COLECŢII DE DATE?

Pe măsură ce dvs. deveniţi un programator tot mai experimentat, începeţi să aveţi pretenţia justificată să cunoaşteţi şi posibilităţile mai deosebite ale calculatorului la care lucraţi.

Tablouri de numere/şiruri

Una din aceste posibilităţi este aceea de a folosi, într-un mod relativ simplu, colecţii mari de date.
Pentru o mai uşoară înţelegere a noţiunilor, să le comentăm puţin înainte de a trece la aplicaţia practică.

DIM. GOSUB. RETURN

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

Tablou de date

Într-o asemenea situaţie se lucrează cu un TABLOU DE DATE.
Un TABLOU reprezintă o colecţie de date identificată printr-un singur nume de variabilă.
Un tablou conţine ELEMENTE şi poate fi cu o singură dimensiune sau poate avea două dimensiuni.
Un tablou unidimensional se mai numeşte LISTA de elemente.

Fie următorul şir 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 conţine valoarea 14, elementul 2 conţine 8 ş.a.m.d.

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

a (se citeşte "a indice 1") conţine "14",
1
a conţine "11", iar a conţine "␣".
4 6

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

	1	2	3	4	
1	3	-1	10	9	← b 1,4
2	2	4	-2	6	← b 2,4

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

```
b    = 10 - element din rîndul 1, coloana 3
 1,3
b    = -2          b    = 9      b    = ___
 2,3                 1,4           2,1
b    = ___         b    = ___
 1,2                 2,4
```

În BASIC indicii se scriu în paranteză:

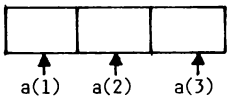
```
a   se scrie a(1)   = 14
 1
a   se scrie a(8)   =  3
 8          a(3)   = __
            a(5)   = __
b   se scrie b(1,3) = 10
 1,3
b   se scrie b(2,3) =  2
 2,3        b(2,4) = __
             ↑  ↑
          rîndul coloana
```

APLICAŢIE PRACTICĂ AP8

Introduceţi:

```
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
```

Notaţi rezultatele în: lista "a"



2. Adăugaţi linia:

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

Aţi obţinut afişarea conţinutului celor trei elemente ale listei şi suma primelor două.

3. Tastaţi:

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

4. Modificaţi linia 50 pentru a fi afişate nummai primele 2 elemente.

```
50 _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
RUN
```

5. Introduceţi:

```
50 FOR i=1 TO 4
RUN
```

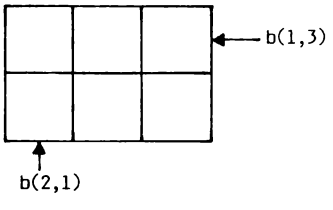
Notaţi mesajul obţinut _
Vi s-a semnalat că aţi cerut un al 4-lea element al unei liste declarată doar cu 3 elemente prin instrucţiunea:

```
DIM a(3)
```

6. Anulaţi programul existent şi introduceţi:

```
 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
```

Completaţi rezultatele în tabloul "b"



7.Schimbaţi programul astfel:

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

Notaţi rezultatele:

```
b(1,1)=_ _ _      b(1,2)=_ _ _      b(1,3)=_ _ _
b(2,1)=_ _ _      b(2,2)=_ _ _      b(2,3)=_ _ _
```

8. Schimbaţi programul:

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

Notaţi ce se obţine: număr element = 1 _ _ _ _
= 4 _ _ _ _
= 9 _ _ _ _
=10 _ _ _ _
=11 _ _ _ _

9. Anulaţi liniile 25, 27 şi schimbaţi:

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

Obţineţi conţinutul listei de caractere.

10. Introduceţi:

```
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. Tastaţi:

```
 5 REM tablou cu 5 rînduri şi 2 coloane
10 DIM t(5,2)
15 REM r=rînd, 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 afişarea 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
```

Comparaţi ce s-a afişat, cu ce conţin enunţurile DATA.

12. Introduceţi alt program:

```
NEW
10 REM program cheltuieli rechizite
15 REM ==================
20 REM afişare 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;"bucăţi";TAB18;"preţ";TAB25;"cost"
110 PRINT "**********************************"
120 PRINT
130 RETURN
150 INPUT "obiect?"␣;o$
152 IF o$="gata" THEN GO TO 90
155 INPUT "bucăţi? ␣ ";b,"preţ unitar? ␣ ";p
160 RETURN
210 LET cost = b*p
220 PRINT o$;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"
bucăţi? 3           preţ? 3.5

obiect? "creioane"
bucăţi? 12          preţ? 0.9

obiect? "linii"
bucăţi? 15          preţ? 2

obiect? "gata"
.............rezultate.....
STOP
```

În acest program au apărut două instrucţiuni BASIC pe care nu le-aţi întîlnit pînă acum:

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

Aceste instrucţiuni permit o structurare a programelor astfel încît să se evite scrierea repetată a anumitor părţi (denumite subrutine sau subprograme).
Analizaţi programul de mai sus.
El este structurat cu o parte principală cuprinsă între prima linie (10) şi linia STOP (99) şi alte părţi secundare denumite SUBRUTINE (subprograme).
În cadrul programului principal se fac trimiteri la subprogramele (subrutinele) pentru:
- afişarea capului de tabel
- introducerea rechizitelor cumpărate
- calculul costului acelor rechizite (cheltuieli)
- calculul sumei totale cheltuite

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

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

Subrutina este apelată prin linia: 30 GO SUB 100

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

```
.
35 LET...
```

Încercaţi să urmăriţi singuri logica programului şi completaţi informaţiile lipsă:
Calculul totalului este determinat de linia: _GOSUB_ _ _.
Subrutina începe cu linia _ _ _ şi se termină cu linia 330 _ _ _ _ după care se execută linia 99.
Introducerea rechizitelor cumpărate, a cantităţii şi a preţurilor corespunzătoare se face ca urmare a liniei : _ _ _ _ _ GO SUB _ _ _ _ _ _ _.
Subrutina începe cu linia _ _ şi se termină cu linia _ _ _ _ RETURN după care se execută linia _ _ REM_ _ _ _ _ _ _ _ _.
Aţi remarcat probabil că pentru a termina execuţia introduceţi cuvîntul "gata" (adică s-au terminat cumpărăturile), Obiectul? "gata". Se afişează rezultatele. STOP.

NOŢIUNI DE BAZĂ

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

LISTA

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

```
          l(i)
nume listă ↑ ↑ poziţia/numărul elementului în listă
```

MATRICE

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

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

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

DIM

Înainte de a fi utilizat un tablou, în program se pune o instrucţiune de rezervare a spaţiului corespunzător în memorie.
Instrucţiunea DIM (de la DIMENSION - în engleză) declară tablouri de numere sau de şiruri 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 intersecţia cărora se află elementul

Cu o instrucţiune DIM se defineşte numai un singur tablou:

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

Instrucţiunea DIM are următoarele defecte:

1 - rezerva pentru tablou spaţiul corespunzător
DIM a(7) - spaţiu pentru 7 elemente
DIM t(3,2) - spaţiu pentru 6 elemente (3*2)
2 - iniţializează elementele tabloului cu zero
Încercaţi:
```
10 DIM a(3)
20 PRINT a(1);a(2);a(3)
RUN
```
3 - şterge orice alt tablou cu acelaşi nume

În progam puteţi da acelaşi nume unui tablou şi unei variabile simple, fără să apară eroare deoarece numele tabloului este întotdeauna indexat. Este deci permis să aveţi:

```
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 rînd, toate fiind de aceiaşi lungime (completate sau scurtate pînă la aceeaşi lungime declarată):

```
          DIM  g$(3,6)
                  ↑ ↑
   3 şiruri───────┘ └──── 6 caractere în fiecare şir
```

```
Tastaţi:  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
```

```
Obţineţi conţinutul matricii:  abcdef   ←---- rîndul 1
                               şirul2   ←---- rîndul 2
                               mdgip␣   ←---- rîndul 3
                               ↑_________6 caractere
```

```
Completaţi:  60 PRINT g$(1,4)        - se va afişa d
             70 PRINT g$(2,4)        - se va afişa _ _
             80 PRINT g$(3,3 TO 5)   - se va afişa gip

         şirul 3←┘     └→caracterele 3 la 5 (3,4 şi5)
```

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

Ce credeţi că va afişa linia 90? _ _ _ _ _ _ _ _ _ _ _ _ _ _
Executaţi programul şi verificaţi!

În liniile 80 şi 90 au apărut două lucruri noi:

1 - se pot oferi odată mai multe caractere ale unui şir sub forma: m TO n

```
                       m TO n
                       ↑    ↑
de la acest caracter───┘    └pînă la acest caracter (inclusiv)
```

2 - se pot alipi caractere din diferite şiruri prin localizarea şi însumarea lor.

SUBRUTINE

Un program BASIC dacă are anumite părţi care se repetă, poate fi structurat astfel încît aceste părţi să fie scrise o singură dată şi să fie apelate ori de cîte ori este nevoie. Astfel de părţi de program sînt denumite SUBRUTINE.

GO SUB

Instrucţiunea GO SUB nr.linie produce saltul în program la subrutina care începe cu linia indicată şi se termină cu prima instrucţiune RETURN.

RETURN

Instrucţiunea RETURN este obligatorie pentru încheierea unei subrutine şi provoacă întoarcerea execuţiei programului la prima instrucţiune 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              - sfîrşit program principal
200 REM subrutina     - început subrutină
.
.
280 RETURN            - sfîrşit subrutină
```

Linia 280 produce întoarcerea la programul principal la liniile 90 şi 140.
În general subrutinele sînt amplasate spre sfîrşitul programului.
Este permis apelul unei subrutine 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
```

```
Programul principal         Subrutina 1        Subrutina 2
   10 ....
   20 ....  ──────────────→100 .....
   30 ....←──┐              110 .....  ───────→200 ......
   40 STOP   └─────────────120 .....←─────────210 ......
```

EXEMPLE DE PROGRAME

Exemplul 1.

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

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

Variabile: 1 - numele listei
n - numărul elementului din listă
s - suma elementelor

Codificarea programului:

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

Exemplul 2.

Program pentru însumarea valorilor dintr-un tablou bidimensional.

Analizarea problemei:

Fie un tablou cu un număr oarecare de linii şi coloane dat la cerere. Programul să citească datele de calcul într-unul sau mai multe enunţuri DATA, memorîndu-le în tabloul dimensionat corespunzător. Să se facă suma pe linii şi coloane, precum şi totalul general.
Pentru a înţelege mai bine problema să considerăm, ca un exemplu particular, un tablou cu 4 linii şi 5 coloane, care să conţină:

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

Bineînţeles totalul general pe linii şi pe coloane trebuie să fie acelaşi.
Afişarea rezultatelor să se facă astfel:
1. tabloul cu date
   conţinutul primei linii
   .
   .
   .
   conţinutul ultimei linii
   .totalizarea pe linii
   linia 1 total:
   .
   .
   .
   linia 1 total:
   total general=
2. tabloul cu date (vizualizat din nou pentru a verifica uşor totalizarea verticală)
   . totalizarea pe verticală
   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(l) - 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 şi 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 afişarea 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 afişarea 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 afişarea 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
```

Analizaţi programul şi sub aspectul scrierii lui (cu linii decalate) astfel încît să fie cît mai uşor de urmărit.
Atenţie! Dacă vi se afişează mesajul: E Out of DATA, 60:1 nu aţi introdus în linia 500 date într-un număr egal cu: număr linii * număr coloane.

ÎNTREBĂRI RECAPITULATIVE ŞI EXERCIŢII

Î 8.1 Ce tipuri de tablouri (listă/matrice) sînt următoarele şi cîte elemente conţin?

a) DIM A$(20)
b) DIM q(5,10)
c) DIM w$(10,16)
d) DIM F(8)

Î 8.2 Ce erori conţin următoarele enunţuri?

a) DIM a(6.8,14)
b) DIM b(k,-3)
c) DIM x(7), y(15)
d) DIM l

Î 8.3 Scrieţi instrucţiunile necesare pentru executarea următoarelor operaţii (folosind cicluri FOR NEXT):
a) să încarce cu "1" o matrice cu 10 linii şi 10 coloane şi să se afişeze în format de 10*10

b) să genereze, să încarce şi să se tipărească o listă cu următorul conţinut:

2	4
8	16
32	64

Î 8.4 Fie o listă a(8). Să se afişeze indicele (i) şi valoarea (v) pentru toate elementele mai mici decît 15, cuprinse în:

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

Afişarea să se facă începînd din coloanele 3 şi 12.

```
coloana 3        coloana 12
   ↓                 ↓
  i=1               v=1
```

Î 8.5 Ce va afişa 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

1. Pentru un tablou numeric "a" cu x linii şi x coloane, scrieţi un program care să calculeze produsul: a(1,1)*a(2,2)*a(3,3)*...*a(x,x)

2. Program pentru afişarea elementelor coloanei a şasea a unui tablou T(8,10).

3. Program pentru afişarea liniei a treia a unui tablou d(5,8).

4. În instrucţiuni DATA puneţi notele obţinute de 15 elevi la 5 materii.
   Realizaţi 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 afişate 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 instrucţiune DIM se defineşte 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(l),
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 veţi cunoaşte alte cîteva posibilităţi ale calculatorului cu care lucraţi:
> moduri de afişare care să pună în evidenţă numai anumite informaţii de pe ecran;
> realizarea de desene pe ecran
> obţinerea desenului de animaţie

Instrucţiuni de utilizare a culorilor. Instrucţiuni grafice. Numere aleatoare. Caractere definite de utilizator

Cip-ul permite colorarea imaginilor, atît alfanumerice cît şi grafice astfel încît dacă puteţi utiliza un televizor/monitor color, lucrul cu calculatorul devine şi mai fascinant.
Puteţi folosi opt culori prin acţionarea în anumite instrucţiuni a cifrelor de la 0 la 7 astfel:

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

Pe ecranul unui televizor/monitor alb-negru aceste culori apar ca nuanţe de la negru la gri deschis.
Cele opt culori pot fi utilizate în următoarele instrucţiuni:

BORDER 0...7 - colorarea conturului (bordurii) ecranului
PAPER 0...7 - colorarea fondului (hîrtiei)
INK 0...7 - colorarea caracterului/punctului afişat

Stabiliţi legătura cu televizorul color şi executaţi aplicaţia practică, observînd cu atenţie ce se întîmplă.

APLICAŢIE PRACTICĂ AP9

1. Introduceţi următoarele instrucţiuni (nenumerotate) şi notaţi ce culori se obţin pe contur.

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

2. Tastaţi:

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

3. Introduceţi următoarea instrucţiune şi veţi obţine afişarea în mijlocul ecranului a unui mesaj clipitor, cu litere albe pe fond albastru:

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

S-a afişat în linia _____ începînd cu coloana _____ .
Afişarea clipitoare se obţine cu instrucţiunea FLASH 1, iar efectul intermitent se anulează cu FLASH 0.

4. Ştergeţi ecranul cu CLS şi introduceţi instrucţiunile necesare pentru a se afişa clipitor, în ultimele două linii ale ecranului un mesaj în roşu şi verde.

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

"Cerneala" (INK) roşie şi verde a fost dată de instrucţiunile _____ şi _____.

5. Tastaţi următoarea linie care va afişa în linia 3, începînd din coloana 3, pe fond (hîrţie) galben şi verde:

```
CLS
PRINT AT 3,3;"Ecranul are";PAPER 6;" ␣ 22 linii";
PAPER 7;" ␣ şi ␣ ";PAPER 4 ;"32 coloane"
```

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

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

Comanda INVERSE 1 inversează culorile pentru PAPER şi INK. Reveniţi la afişarea curentă (fără CLS):

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

Am lăsat intenţionat litera "B" afişată invers pentru a sesiza diferenţa.

7. Introduceţi:

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

8. În exemplul următor realizaţi şi afişarea color a unor linii, dar reţineţi şi semnificaţia textului care are coordonatele extreme ale ecranului în afişarea grafică;

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

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

9. Introduceţi următorul program, urmărind cu atenţie comentariile:

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

Priviţi cu atenţie ecranul şi veţi observa cele patru puncte.

10. Adăugaţi 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. Continuaţi 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. Anulaţi liniile 65,70,75,80,90,95,100 şi modificaţi liniile:

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

Se trasează axa x dintr-o dată şi mai rapid decît 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. Desenaţi 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>  (tastaţi de două ori)
```

Instrucţiunea PAUSE am introdus-o pentru a observa treptat trasarea laturilor pătratelor.
Anulaţi liniile 25, 35 şi 45 şi executaţi din nou programul.
Completaţi linia 5

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

Ce s-a schimbat în privinţa culorilor?

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

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

15. Desenaţi 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 afişarea alfanumerică pentru instrucţiunea: PRINT AT linia, coloana este în colţul _ _ _ _ _ _ _ _ = _ _ _ _ _ _

-punctul de origine la afişarea grafică, pentru instrucţiunile : PLOT x, y
DRAW x, y
CIRLE x, y, r
este în colţul _ _ _ _ _ _ _ = _ _ _ _ _ _ _ _.

17. Introduceţi 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 obţine efecte neaşteptate (deosebit de utile la jocuri), generînd numere la întîmplare, între anumite limite date prin program. Acestor numere li se mai spune numere ALEATOARE (în engleză RANDOM = aleator, la întîmplare).

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

18. Schimbaţi 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
```

Observaţi că instrucţiunea DRAW permite utilizarea opţiunilor de culoare pentru PAPER şi INK, în mod asemănător cu PRINT. Executaţi de cîteva ori programul cu RUN.

20. Să analizăm acum puţin numerele aleatoare generate prin instrucţiunea RND, reluînd 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
```

Notaţi valorile afişate: _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _
Executaţi programul din nou RUN
Comparaţi cele afişate acum cu cele înscrise mai sus şi observaţi valorile diferite pentru x şi y.
Dacă doriţi numere întregi aleatoare, schimbaţi, de exemplu linia 30:

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

Notaţi: _ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _
De data aceasta y ia valori întregi.

21. Şi acum un program pentru trasarea graficului funcţiei trigonometrice SIN între 0 şi 2 PI cu două culori şi 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. Introduceţi: NEW

```
 1 REM folosirea caracterelor grafice standard ■ şi ▄
 5 BORDER 6
10 PRINT AT 18,6;INK 2; "■■■■■■■■■■■■■■■■■■■■"
20 LET l=17
30 LET y$="▄"
40 FOR c=6 TO 20
50 PRINT AT l,c;y$
60 LET l=l-1
70 NEXT c
80 FOR l=3 TO 18
90 PRINT AT l,21;INK 2;" ■ "
100 NEXT l
RUN
```

CAPITOLUL 9

Obţineţi o scară solidă. Să-i dăm o utilizare făcînd să sară pe ea o... minge reprezentată prin litere "O". Veţi învăţa astfel cum se realizează animaţia pe ecran. Adăugaţi 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 şterge 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 1,22; INK 1; "O"
280 PAUSE 2
290 PRINT AT 1,22; " "
300 NEXT l
310 NEXT k
RUN
```

Animaţia se obţine într-un ciclu FOR...NEXT, prin afişarea într-o poziţie, menţinerea imaginii pe o anumită durată şi ştergerea imaginii, urmată de afişarea în poziţia 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 1,22; "O"      -afişază "O"
280 PAUSE 2                 -menţine imaginea
290 PRINT AT 1,22; " "      -şterge pe "O"
300 NEXT l                  -poziţia următoare
```

NOŢIUNI DE BAZĂ

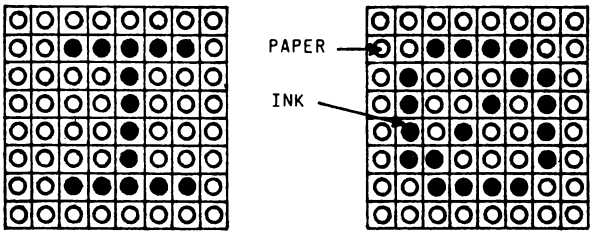
Dacă aveţi la dispoziţie un televizor color, cu CIP-ul puteţi realiza opt culori, astfel:

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

Un caracter este afişat într-un pătrat format din 8*8 puncte. De exemplu litera "I" şi cifra "O" sînt afişate 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 putînd avea valoarea de la 0 la 7.



BORDER

Instrucţiunea BORDER c colorează marginea imaginii cu culoarea dată de valoarea lui c.

BORDER 2 - margine roşie
BORDER 6 - margine galbenă

Culorile şi afişarea clipitoare (FLASH) sînt considerate ca ATRIBUTE ce pot fi asociate unui caracter.
Să afişăm 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 instrucţiunea PRINT.
Următorul program va afişa cele opt culori şi 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
```

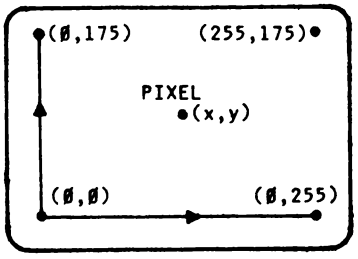
Dacă dorim ca cifrele să fie mai evidente, se poate obţine un contrast mai bun introducînd: INK 9 sau PAPER 9
Cifra 9 nu este asociată unei culori, ci are ca efect cerneala mai deschisă pe fond închis (negru, roşu, magenta) şi cerneala mai închisă pe fond mai deschis (verde, bleu, galben, alb).
Verificaţi introducînd linia:

```
15 INK 9
RUN
```

Atributele de culoare şi clipire le puteţi folosi cu mare efect, atunci cînd doriţi să scoateţi în evidenţă anumite informaţii de pe ecran.
Dacă, de exemplu aveţi ecranul plin şi afişarea este uniformă, cel care analizează cele afişate trebuie să parcurgă tot conţinutul pentru a găsi o informaţie. Afişînd însă în altă culoare şi eventual clipitor, imediat informaţia respectivă atrage atenţia privitorului.
Poziţionarea pe ecran se face cu: PRINT AT linie, colona
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;"OPRIŢI BANDA"
PRINT PAPER 1;INK 7;FLASH 1;AT 11,10;"OPRIŢI BANDA"
```

Realizarea desenelor pe ecran se obţine prin puncte.
Un punct este denumit PIXEL şi poate fi poziţionat prin program, cu două coordonate faţă de colţul din stînga - jos al ecranului (vezi figura 5.1)



PLOT

Instrucţiunea PLOT x, y stabileşte poziţia şi desenează un punct la coordonatele x, y:

x = 0 - 255
y = 0 - 175

Desenele pot fi afişate punct cu punct. De exemplu trasarea a două axe la extremităţile ecranului se poate face astfel:

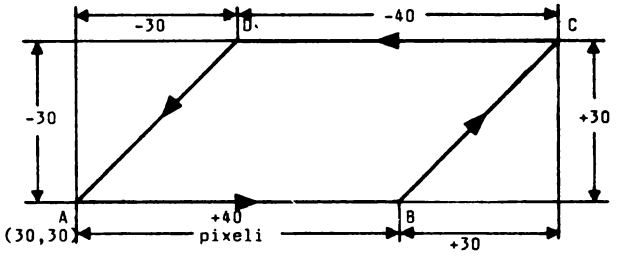
```
axa x:  FOR x=0 TO 255 : PLOT x,0: NEXT x
axa y:  FOR y=0 TO 175 : PLOT 0,y: NEXT y
```

DRAW

Instrucţiunea DRAW x,y desenează o linie de la poziţia curentă a cursorului grafic pînă la un punct de coordonate x şi y pixeli. De exemplu trasarea celor două axe, luînd ca punct de origine (10,10) se poate obţine cu:

```
NEW
10 PLOT 10, 10
20 DRAW 245, 0
30 PLOT 10,10
40 DRAW 0, 165
```

Pentru a înţelege mai bine folosirea instrucţiunii DRAW să analizăm desenarea paralelogramului din figură:



Ne propunem să începem desenul din colţul A de coordonate x = 30; y = 30 şi 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 puteţi obţine construcţii spectaculoase ca în programul următor:

```
NEW
10 BORDER 6
20 FOR n=1 100 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 instrucţiunii DRAW este: DRAW x, y, z, -z este unghiul (în radiani) pentru desenarea unei linii curbe între poziţia cursorului grafic şi poziţia 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
```

Observaţi că între cele două extremităţi date de (125, 0) şi (0, 100) se obţin curbe diferite în funcţie de valoarea lui z dată în radiani.

CIRCLE

Instrucţiunea CIRCLE x, y, r desenează un cerc de rază r cu centrul în punctul de coordonate x şi 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 întîmplare

Adesea în program (în special în jocuri) se folosesc numere diferite generate "la întîmplare". Astfel de numere ALEATOARE (în engleză : RANDOM NUMBERS). Imaginaţi-vă diferite combinaţii de numere date de aruncarea a două zaruri şi veţi realiza ce se înţelege prin numere aleatoare. Există diferite metode matematice pentru a produce numere aleatoare. CIP-ul generează astfel de numere cu funcţia RND.

RND

RND - generează numere zecimale între 0 şi 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 şi a.
Schimbaţi linia 20:

```
20 LET n=15*RND
RUN
```

Executaţi programul de cîteva ori şi veţi obţine numere între 0 şi 15.

▸ b + a* RND - generează numere zecimale între b şi (a+b)
Schimbaţi:

```
20 LET n = 2 + 15 * RND
```

Se obţin numere zecimale cuprinse între 2 şi 17.

```
20 LET n = 4.4 + 0.6 * RND
```

Se obţin numere între 4,4 şi 5

▸ INT (a*RND) -generează numere întregi între 0 şi a

```
20 LET n = INT(6*RND)
```

▸ b + INT(a*RND) -generează numere întregi între b şi (a+b)

```
20 LET n = 1 + TNT(6*RND)
RUN
```

Executaţi de cîteva ori programul şi veţi obţine numere aleatoare cuprinse între 1 şi 6, deoarece funcţia INT întregeşte numărul la valoarea întreagă inferioară.

```
INT (3,65) = 3
INT (5,999) = 5
```

RAND (RANDOMIZE)

Uneori este însă nevoie de numere aleatoare în seturi care să se repete. Această posibilitate o dă enunţul RANDOMIZE (prescurtat: RAND). Pe tastatură obţineţi (vezi anexa B):

"RAND" cu tasta T în modul K
"RND" cu tasta T în modul E

Pentru a vedea diferenţa între funcţia RND şi enunţul RAND, introduceţi 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 execuţie a programului veţi obţine trei seturi diferite de numere întregi aleatoare între 1 şi 10.
Completaţi programul cu liniile:

```
10 FOR x = 1 TO 5
20 INPUT "n= ␣ ";n
30 CLS
70 RANDOMIZE n
140 NEXT n
```

Enunţul RAND are sintaxa: RAND n unde n = 0 la 65535.
Executaţi programul dînd lui x valorile următoare şi notaţi rezultatele pentru a sesiza diferenţele şi asemănările între ele:

```
n = 1       <ENT>
n = 70      <ENT>
n = 999     <ENT>
n = 12345   <ENT>
n = 65535   <ENT>
```

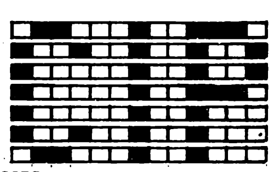
Analizînd rezultatele notate, observaţi că pentru o valoare dată lui n se obţin aceleaşi seturi de numere, aleatoare, dar diferă numerele din seturi, pentru diferite valori ale lui n.
RAND şi RAND 0 utilizează timpul trecut de la punerea în funcţiune a calculatorului, care dacă nu creşte mult între două execuţii ale lui RANDOMIZE, determină generarea unor seturi cu numere aproximativ aceleaşi.
Încercaţi execuţia programului repetînd 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
```

Întrerupeţi cînd doriţi cu BREAK.
Caracterele grafice standard se obţin în modul G (CS+9), cu tastele numerice 1 la 8 (vezi anexa B). Ele sînt utile cînd doriţi să afişaţi texte sau grafice de dimensiuni mai mari decît cele obişnuite.
Introduceţi:



```
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?

Aşa cum am arătat, un caracter este reprezentat (afişat) într-un pătrat de 8 × 8 puncte. Folosind acest mod de reprezentare utilizatorul poate să-şi definească, simboluri grafice introducînd 0 pentru PAPER şi 1pentru INK.
Dacă priviţi în Anexa 1, codurile 144 la 164 sînt 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 obţinut oricînd se trece în modul grafic şi se acţionează tasta respectivă.
Pentru exemplificare să încercăm să definim litera grecească "α" pe tasta <A>:
1. punctăm noul caracter în 8 × 8 puncte lăsînd însă la margini cîte un şir nefolosit pentru a separa caracterul de altele alăturate:

	1	2	3	4	5	6	7	8
1	O	O	O	O	O	O	O	O
2	O	O	O	O	●	●	O	O
3	O	O	O	●	O	O	●	O
4	O	O	O	●	O	O	O	O
5	O	O	●	●	O	O	O	O
6	O	●	O	●	O	O	●	O
7	O	O	●	O	●	●	O	O
8	O	O	O	O	O	O	O	O

2. se memorează fiecare din cele 8 rînduri cu enunţul BIN (provenit din BINAR) urmat de 8 cifre binare:

0 - pentru fond hîrtie (PAPER)
1 - pentru cerneală (INK)

Cele opt numere binare rezultate sînt înscrise în memoria internă a CIP-ului în opt poziţii, fiecare poziţie avînd o ADRESĂ. Adresa primului rînd este URS"A" (URS de la USER şi "A" de la tasta pe care am stabilit-o pentru noul caracter).
Pătratul 8 × 8 puncte devine:

OCTEŢI		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 biţi (cifre binare) poartă numele de octet (în engleză BYTE).
Recapitulînd puţin:

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 octeţi reprezentînd caracterul " α ".

```
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 0 1 1 0
130 DATA BIN 0 0 0 1 0 0 1 0
140 DATA BIN 0 0 0 1 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 execuţia programului vi se afişează mesajul: OK, 180:1

3.Introduceţi linia:

```
200 PRINT "aaaa        (fără <ENT>)
```

4.Intraţi în mod grafic şi apăsaţi din nou "A". Acum se afişează " £ ". Completaţi linia 200:

```
200 PRINT "aaaa ␣ £££££ "
```

Atenţie! Pentru a introduce ghilimelele de la sfîrşitul şirului, reveniţi din modul G , în modul L reapăsînd (CS + 9)

```
RUN 200
```

Se afişează aaaa £££££
De cîte ori stabiliţi modul G , prin apăsarea tastei <A> veţi obţine " £ ".

Animaţia (mişcarea) pe ecran

Efectul de mişcare se obţine prin afişarea unui desen într-o poziţie, menţinerea imaginii pe o durată scurtă, ştergerea desenului şi reafişarea lui în poziţia următoare.
Menţinerea limitată a imaginii se realizează cu comanda:

```
PAUSE n
```

Opreşte execuţia 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. Puteţi 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 "O" 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 afişarea paletei
60 PRINT AT 21, 10; "/"
70 PRINT AT n, 10; "O"
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; "O"
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 apariţii ale cifrelor 1 la 7 (utilizate pentru culori) generate de funcţia RND, dintr-un număr mare de execuţii.

Dacă de exemplu cerem prin program generarea a 700 numere întregi aleatoare, între 1 şi 7, ne-am aştepta ca să obţinem 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 şi 7 şi vor fi contorizáte (însumate) apariţiile fiecăreia dintre ele.
Rezultatele să fie afişate sub forma:

```
număr          de cîte ori
 1                 140
 2                  96
 .
 .
 .
 7                 102
```

Pentru ca exemplul să cuprindă cît mai multe din noţiunile acestui capitol şi pentru ca afişarea rezultatelor să fie cît mai expresivă, să se asocieze cifrelor 1 la 7 culorile corespunzătoare şi să se obţină sub liniile cu rezultate, un grafic vertical cu şapte coloane a căror înălţime să fie proporţională cu numărul de apariţii pentru fiecare cifră (culoare).
Pentru a însuma numărul de apariţii să se folosească un tablou unidimensional cu 7 elemente declarat astfel:

```
10 DIM c(7)
```

- Codificarea programului:

```
10 DIM c(7)
15 PRINT FLASH 1; AT 10, 0; "AŞTEPTAŢI 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 apariţiilor
40 LET c(a) = c(a) + 1
50 NEXT k
55 REM afişare 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 încercaţi înţelegerea programului răspunzînd la întrebările următoare:

ÎNTREBĂRI RECAPITULATIVE

Î 9.1 Ce instrucţiune a determinat:
a) colorarea conturului ecranului?
b) afişarea clipitoare a mesajului "AŞTEPTAŢI..." ?
c) colorarea rezultatelor orizontale?

Î 9.2 Care a fost efectul instrucţiunii INK 9?

Î 9.3 Ce linie de program a trasat graficul vertical pentru apariţiile cifrei 3?

Î 9.4 În ce linii de program trebuie să faceţi modificări pentru a genera 10000 de numere întregi aleatoare?

Î 9.5 Ce instrucţiune bănuiţi că generează sunete proporţionale ca durată şi înălţime cu numărul de apariţii ale cifrelor? (căutaţi singura instrucţiune neexplicată pînă la acest capitol).

Probleme

Realizaţi probleme pentru:

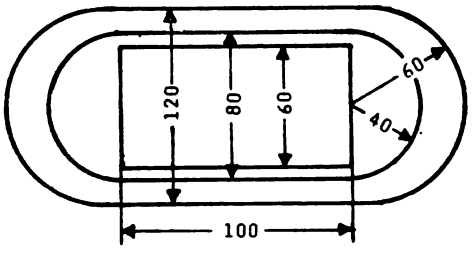
a) generarea şi afişarea a 20 de numere aleatoare între 0 şi 100

b) trasarea axelor x şi y de lungime maximă, începînd cu colţul din dreapta sus

c) umplerea ecranului cu puncte aleatoare, pe fond închis (ca un cer înstelat)

d) săritura amortizată a unei mingi (înălţimea să descrească pînă la zero)

e) desenarea unui "stadion" de forma:



f) obţinerea literei greceşti β prin apăsarea tastei "B"

RĂSPUNSURI

R 9.1 a) 120 BORDER 0
b) 15 PRINT FLASH 1; AT 10, 0;
"AŞTEPTAŢI..."; FLASH 0
c) 90 PRINT PAPER p...

R 9.2 Afişarea contrastă între INK şi PAPER

R 9.3 Linia 150

R 9.4 În linia 15 pentru a modifica timpul de aşteptare şi în linia 20 care devine: 20 FOR k = 1 TO 10000

R 9.5 Instrucţiunea BEEP (vezi anexa C) care va fi explicată în capitolul 10.

SĂ INTRODUCEM SUNETE

După cum vă amintiţi din capitolul 1, CIP-ul este înzestrat cu un difuzor şi este capabil să oroducă multe melodii pe placul dvs.

APLICAŢIE PRACTICĂ AP 10

1.Introduceţi următorul program:

```
NEW
 2 PRINT AT 10,1;FLASH;"ASCULTAŢI!"
 3 PRINT:PRINT FLASH 0;"Cunoaşteţi 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
```

Observaţi ce se întîmplă... Fără a comenta prea mult, aţi programat calculatorul să facă muzică, respectiv să interpreteze ceva foarte cunoscut, nu?

2.Tastaţi:
```
NEW
BEEP 1,0
```

Ce se întîmplă? Pentru circa 1 secundă, pe ecran nu apare nici o schimbare, în schimb, în microdifuzorul încorporat se aude un sunet în care veţi recunoaşte nota DO din octava 1 (cei cu ureche muzicală, bineînţeles).

Mai încercaţi:
```
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
```

Sesizaţi că înălţimea diferă, în funcţie de valoarea celui de al doilea argument din BEEP, iar durata sunetului este dată de primul argument.

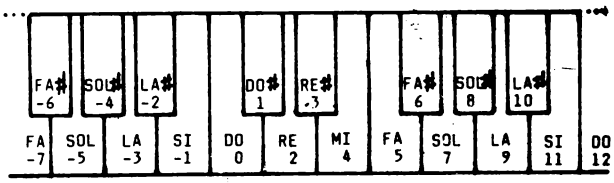
NOŢIUNI DE BAZĂ DESPRE PRODUCEREA SUNETELOR

Producerea sunetelor se face cu instrucţiunea BEEP, care are forma generală: BEEP d,i
> d (primul argument) reprezintă durata sunetului. Durata în secunde are valori permise între 0 şi 10.
> i (al doilea argument) indică înălţimea sunetului (frecvenţa), şi are valori permise între -60 şi +69.

Folosirea unor valori în afara limitelor produce o eroare care se soldează cu întreruperea execuţiei şi un mesaj de eroare.

Codificarea notelor

Corespondenţa între valorile lui i şi gama muzicală (temperată) este dată de următoarea regulă:
Notei DO din octava 1 (DO CENTRAL de pe claviatura unui plan) îi corespunde valoarea 0 mergînd în sus (DO DIEZ, RE, RE DIEZ ...) se creşte 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 aceiaşi notă cu RE BEMOL etc..., respectiv diezii şi bemolii au ca efect adunarea sau scăderea unei unităţi la valoarea corespunzătoare notei (similar ca la pian).



Exemple de programare

Să încercăm pentru început să punem calculatorul să cînte arpegiul gamei DO MAJOR. Avem nevoie de notele DO, MI, SOL, DO şi 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 aceleiaşi game, dar în altă octavă, ar trebui să adunăm sau să scădem multiplii de 12 (sîntem siguri că aţi observat că între doi de DO sînt 12 unităţi diferenţă). Respectiv, pentru octava 2, ar trebui să folosim valorile 12, 16, 19, 24.
Mai elegant însă ar fi să scriem programul astfel:
(editaţi programul anterior, linie cu linie, pentru a-l obţine 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
```

şi să-i dăm lui n valori de forma ± 12*k avînd grijă să nu ieşim din limitele permise. Ca rafinament suplimentar, propunem următorul program (procedaţi la fel, prin editarea programului anterior):

Schimbarea duratei

```
 5 REM arpegiul gamelor DO MAJOR
10 BEEP .2*m,0+n
20 BEEP .2*m,4+n
30 BEEP .2*m,7+n
40 BEEP .2*m,12+n
50 BEEP .2*m,12+n
60 BEEP .2*m,7+n
70 BEEP .2*m,4+n
80 BEEP .2*m,0+n
```

şi să-i dăm lui m diverse valori avînd deasemeni grijă să nu ieşim din gama valorilor permise.

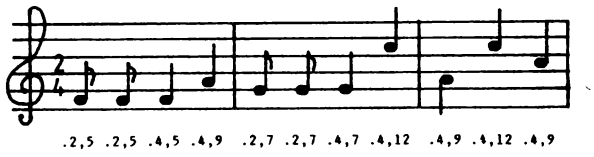
Schimbarea tempou-lui

Am realizat "parametrizarea" programului pentru diverse durate ale notelor şi diverse octave. Dacă am conveni, de exemplu că .2 secunde este durata unei optimi, dînd diverse valori (întregi sau nu) lui m, am putea interpreta efectul prin schimbarea tempo-ului , sau înlocuirea optimilor cu doimi, note întregi, şaisprezecimi etc...
O utilizare a acestui mod de programare a muzicii ar fi posibilitatea "acordării" calculatorului cu alte instrumente.
Ca soluţie de "codificare" a unei partituri, sugerăm să completaţi portativul cu o linie dedesubt şi una deasupra, şi să scrieţi cu creionul în dreptul liniilor şi spaţiilor (ţinînd cont de armura cheii) valorile corespunzătoare notelor.
Încă o treabă foarte importantă: cum codificăm pauzele? o soluţie ar fi folosirea instrucţiunii PAUSE n, ţinînd 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 şi o valoare pentru înălţime care să iasă din spectrul practic audibil (ex.69). În acest mod uniformizăm regula de durată şi 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 .4,5: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
```



Introduceţi programul şi cu RUN însărcinaţi calculatorul cu descifrarea partiturii. Melodia care se aude nu se poate să nu o recunoaşteţi. Dacă doriţi, vă puteţi folosi la introducere de "parametrizarea" tempo-ului şi 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
```

Continuaţi...
Mai facem o precizare pentru cei care doresc să facă din acest domeniu un cîmp de aplicaţii mai serioase: există programe utilitare specializate, aşa numite editoare de muzică cu care se poate ajunge la performanţe mai ridicate în componistica asistată de calculator.
Deasemeni mai precizăm că la frecvenţe joase, sunetele ies din clasa celor "muzicale" putînd fi folosite pentru efecte sonore interesante.

ÎNTREBĂRI RECAPITULATIVE ŞI EXERCIŢII

Î 10.1 Care este forma generală a instrucţiunii BASIC de prelucrare a sunetelor?

Î 10.2 Care sînt limitele permise pentru cele două argumente ale instrucţiunii BEEP?

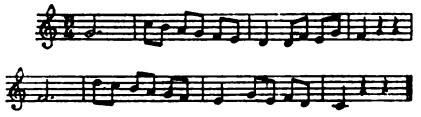
Î 10.3 Cărei note îi corespunde valoarea 0 a celui de al doilea argument al instrucţiunii BEEP?

Î 10.4 Care este regula de construire a codificării notelor, plecînd de la corespondenţa Do central=0?

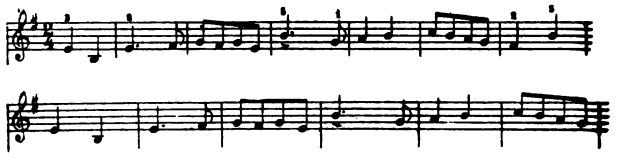
Probleme

1. Scrieţi un program care să cînte gama DO MAJOR în ambele sensuri.

2. Combinaţi BEEP cu RND pentru a genera sunete de durate şi înălţimi aleatoare, pe un timp nedefinit (întrerupeţi cu (BREAK!).
3. Codificaţi următoarea partitură:



4. Codificaţi 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 Faţă de DO central se adună 1 spre dreapta şi se scade 1 spre stînga, 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 3*d,7 : BEEP d,12 :BEEP d,11
20 BEEP d,9 :BEEP d,7 :BEEP d,5:BEEP d,4
30 BEEP 2*d,2 :BEEP d,2:BEEP d,5 :BEEP d,4
40 BEEP d,7:BEEP 2*d,5:BEEP 2*d,69:BEEP 2*d,69
50 BEEP 3*d,5:BEEP d,14:BEEP d,12:BEEP d,11
60 BEEP d,9 : BEEP d,7 :BEEP d,5
70 BEEP 2*d,4:BEEP d,7:BEEP d,4:BEEP d,5:BEEP d,2
80 BEEP 2*d,0 :BEEP 2*d,69 :BEEP 2*d,69
```

Se va alege pentru d o valoare convenabilă pentru durata unei optimi (ex: .3).

Problema 4

```
 5 LET d=...
10 BEEP 2*d,4 : BEEP 2*d,-1
20 BEEP 3*d,4:BEEP d,6:BEEP d,7:BEEP d,6:BEEP d,7
30 BEEP d,4:BEEP 3*d,11:BEEP d,7:BEEP 2*d,9
40 BEEP 2*d,11:BEEP d,12:BEEP d,11:BEEP d,9
45 BEEP d,7:BEEP 2*d,6:BEEP 2*d,11:BEEP d,9
50 BEEP d,7 :BEEP d,6 :BEEP d,7
60 BEEP 2*d,4 :BEEP 2*d,-1:BEEP 3*d,4 :BEEP d,6
65 BEEP d,7 :BEEP d,6 :BEEP d,7 :BEEP d,4
70 BEEP 3*d,11 :BEEP d,7
75 BEEP 2*d,9 : BEEP 2*d,11 :BEEP d,12 :BEEP d,11
80 BEEP d,9 :BEEP d,7
90 BEEP 2*d,7 :BEEP 2*d,6 :BEEP 4*d,4
```

Se va alege pentru d ca şi în exemplul precedent, o valoare convenabilă (ex: .3).

FUNCŢII

BASIC-S vă pune la dispoziţie anumite funcţii pe care le puteţi folosi în expresii introducînd numele funcţiei urmat de o valoare numit argument.
Efectul este că în calcularea expresiei respective numele de funcţie va fi înlocuit cu valoarea funcţiei corespunzătoare valorii argumentului.

APLICAŢIE 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) şi COS(x). Deasemeni, se observă că în linia 10 apare PI, semnificînd funcţia constantei cu acest număr, respectiv: 3,1415927, deasemeni SIN şi COS, apelul lor provocînd furnizarea unor valori numerice pe care le-am folosit în liniile 10, 30 şi 40.

NOŢIUNI DE BAZĂ

Funcţiile trigonometrice **SIN, COS, TAN, ASN, ACS, ATN:**
Forma generală a acestor funcţii este:

[n] SIN x sau [n] SIN(x)

variabilă pentru toate cele trei funcţii: SIN, COS, TAN şi inversele lor.
Semnificaţia lor este cea cunoscută din trigonometrie, argumentul funcţiilor directe este o expresie numerică în radiani. Toate aceste funcţii se pot utiliza oriunde este acceptată o expresie numerică.

Exemplificare:

```
PRINT SIN(0)        va afişa 0
PRINT SIN(PI/2)     va afişa 1
PRINT SIN PI/2      va afişa 0
PRINT ASN (1)       va afişa 1.5707963
PRINT ASN 1         va afişa 1.5707963
PRINT TAN(ATN(3))   va afişa 3
```

Se observă că folosirea parantezelor poate schimba valorile obţinute. Vă recomandăm ca atunci cînd argumentul este o expresie, aceasta să fie inclusă între paranteze. Altfel rezultatul poate să nu fie cel mai corect.

Funcţia PI

Forma generală este: PI şi are ca efect obţinerea valorii numărului (3.1415927 ...). Se poate folosi oriunde avem nevoie de o valoare mai exactă a numărului fără a-i scrie noi efectiv atîtea zecimale cîte avem nevoie. (Calculatorul îl generează cu 7 zecimale).

Funcţiile **ABS, SGN, INT, SQR, EXP, LN, BIN:**

Forma generală pentru **ABS:**

[n] ABS x sau [n] ABS (x)

valabilă pentru toate funcţiile enumerate, argumentul x fiind un număr sau o expresie numerică.

ABS - returnează valoarea absolută a expresiei argument calculată.

```
Exemple:   PRINT ABS (-10)        va afişa 10
           PRINT ABS (10)         va afişa 10
           PRINT ABS (10-25)      va afişa 15
```

SGN - returnează: 1 dacă argumentul > 0
0 dacă argumentul = 0
-1 dacă argumentul < 0

```
Exemple:    PRINT SGN (-10)        va afişa -1
            LET x=15
            PRINT SGN x            va afişa 1
            PRINT SGN (-x)         va afişa -1
            PRINT SGN (x-x)        va afişa 0
            PRINT SGN x-x          va afişa -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 retruna 3
            SQR -9      va produce un mesaj de eroare
```

EXP - calculează valoarea funcţiei exponenţiale e ↑ x.

```
Exemple:  PRINT EXP 10  va returna 22026,466, care repre-
                        zintă puterea a-10-a a numărului e.
          PRINT EXP 3.14  va retruna 23,103867 deci e la
                          puterea 3.14
```

LN - calculează valoarea funcţiei LN x

```
Exemple:    PRINT LN 10            va afişa 2.3025851
            PRINT LN(EXP(10))      va afişa 10
```

BIN xxxxxxxx - această funcţie specifică faptul că urmează un număr în reprezentarea binară pe un octet. Are ca argument un şir de 8 cifre binare.

POINT are forma: [n] POINT (x,y) şi la apel, returnează:
0 - dacă punctul de pe ecran din poziţia x,y are culoarea (INK) identică cu a fundalului (PAPER).
1 - în caz contrar.

Funcţiile: LEN, STR$, VAL, VAL$, CHR$, CODE

LEN Funcţia LEN avînd ca argument un şir sau o expresie de tip şir de caractere returnează numărul caracterelor din şir.

```
Exemple:    PRINT LEN ("ABCDE")        va afişa 5
```

**STR$** Funcţia STR$ are argument numeric şi returnează un şir de caractere care reprezintă numărul.

```
Exemple:    PRINT STR$ 10          va returna 10
            PRINT STR$ (2E3)       va returna 12000
```

VAL Funcţia VAL are efectul invers; are ca argument un şir de caractere numerice şi returnează un număr.

```
Exemple:    PRINT VAL "34.756"          va afişa 34.75
            PRINT VAL "3E4"             va afişa 30000
            PRINT VAL ("2"+"*3"))   va afişa 6
```

**VAL$** Funcţia VAL$ este similară dar are ca rezultat tot un şir.

```
Exemplu:    VAL$ """abcde"""        va afişa "abcde"
```

**CHR$** Funcţia CHR$ are ca argument un număr şi returnează caracterul care are cod acel număr (v. şi anexa A).

```
Exemplu:    PRINT CHR$(42)         va afişa *
```

CODE Funcţia CODE este inversa funcţiei CHR$. Ea are ca argument un şir şi întoarce codul primului caracter al şirului.

```
Exemplu:    PRINT CODE "*"      va afişa 42
```

**INKEY$** Funcţia INKEY$ returnează caracterul corespunzător tastei apăsate ultima dată. Dacă nu s-a apăsat nici o tastă, se returnează şirul vid.

```
Exemple:    10 PRINT INKEY$;   :PAUSE 10
            20 GO TO 10
```

va afişa caracterul de pe tasta pe care aţi apăsat-o sau un şir vid.

**SCREEN$** Această funcţie se foloseşte în cadrul lui LOAD şi SAVE şi înlocuieşte adresa de început a zonei ecran în memorie şi numărul de octeţi ai acesteia, precedate de CODE. Este utilă în LOAD şi SAVE. SCREEN$ <==> CODE 16384,6912

```
Exemplu:    LOAD "SCREEN$" este echivalent cu LOAD "CODE
            16384,6912"
```

Funcţii definite de utilizator: DEF FN şi FN

În afară de funcţiile enumerate pe care calculatorul vi le pune la dispoziţie, aveţi posibilitatea să vă definiţi funcţii proprii, corespunzătoare programelor pe care le scrieţi. Pentru aceasta se alege un nume, format dintr-o literă sau dintr-o literă urmată de $ dacă funcţia va returna ca valoare un şir de caracterre, şi o listă de argumente.
Definirea se face cu: [n] DEF FN nume(x1,x2 ...xn)= expresie unde prin expresie se înţelege o expresie numerică sau de tip şir de caractere, care poate la rîndul ei să conţină apeluri de funcţii.
Trebuie notat că argumentele funcţiei sînt în număr de maximum 26 numerice şi 26 de tip şir. De asemenea, ca notă specială, trebuie reţinut că variabilele funcţiei sînt numite "legate", în sensul că valabilitatea lor este restrînsă la linia de definire a funcţiei, neavînd nici o legătură cu o altă posibilă variabilă cu acelaşi 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+x*y+a
40 DEF FN n()=x+y+x*y+a
50 PRINT FN m(2,5)
60 PRINT FN n()
```

La execuţia liniei 50, valorile 2 şi 5 sînt trecute pe seama variabilelor x şi y din linia 30 fără a altera pe cele din linia 10. Pentru evaluarea funcţiei m, rezultînd valoarea 32 (verificaţi!), care se afişează. La evaluarea liniei 60, se apelează funcţia n definită în linia 40, x şi y fiind de data aceasta variabilele libere x şi y definite în linia 10 cu valorile 0 şi 10 deci rezultatul funcţiei, deşi are aceeaşi expresie, va fi altul - cel care se va afişa pe ecran (25).

Exerciţii recapitulative

E 11.1 Să se scrie un program de afişare a valorilor funcţiilor EXP şi LN.

E 11.2 Să se afişeze 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 şi +100 cu pasul 2.

E 11.3 Folosind funţia 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 afişeze caracterele respective.

E 11.4 Să se calculeze şi să se afişeze pentru fiecare x folosind funcţii utilizator valorile funcţiei:

$$e = \begin{cases} x+0.5 \ln x & \text{pentru } x \in (0,1] \\ \dfrac{2x + \ln x}{2} & \text{pentru } x \in (1,10] \\ \dfrac{2x\uparrow 2 + 2x + x\ln x + \ln x}{2} & \text{pentru } x > 10 \end{cases}$$

E 11.5 Afişaţi valorile funcţiei POINT pentru 10 puncte de pe ecran alese aleator.

CAPITOLUL 11

PROBLEME

P 11.1 Să se scrie un program care să calculeze rădăcinile ecuaţiei de gradul II (toate cazurile).

P 11.2 Să se calculeze şi să se afişeze pentru fiecare x valorile funcţiei:

$$f(x)=\begin{cases}\max\{1+x*x,\ 1-x,\ \cos x\} & \text{pentru } x <= 0\\ e \uparrow x & \text{pentru } 0 < x < =1\\ 0 & \text{în rest pentru intervalul -10 şi +2 cu pasul .2}\end{cases}$$

P 11.3 Desenaţi pe ecran 3 cicloide.
Indicaţie: Cicloida este definită de ecuaţiile

```
y=r-dcos
x=r+dsin
    şi se va folosi funcţia PLOT
```

Vă recomand: x între 0 şi 22 cu pasul 1/10
r=10, d=20
şi în loc de y se va calcula şi 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 ↑ x=";l$
 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)=(2*x+(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
```
şi lansaţi de mai multe ori cu RUN

P 11.1
```
 10 REM REZOLVAREA ECUAŢIEI DE GRADUL II
 20 CLS
 30 PRINT "REZOLVAREA ECUAŢIEI DE GRADUL II, DE FORMA
        ax +bx+c=0"
 40 INPUT "INTRODUCEŢI VALOAREA a:";a
 45 IF a = 0 THEN GO TO 190
 50 INPUT "INTRODUCEŢI VALOAREA b:";b
 60 INPUT "INTRODUCEŢI VALOAREA c:";c
 70 LET D=b*b-4*a*c
 80 IF D >= 0 THEN GO TO 110
 90 PRINT "ECUAŢIA 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+SQRD)/(2*a)
160 LET x2=(-b-SQRD)/(2*a)
170 PRINT "x1=";x1;"x2=";x2
180 STOP
190 PRINT "a = 0 ecuaţia 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 < cosx THEN LET f=cosx
 60 GO TO 110
 70 IF x > 1  THEN GO TO 100
 80 LET f=EXPx
 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*SINf
 30 LET y=(10-20*COSf)+10
 40 PLOT INK2, x,y
 50 NEXT f
```

CUM LUCRĂM DIRECT CU MEMORIA CIP-ULUI

Pentru acei dintre dvs. care au asimilat bine noţiunile prezentate pînă acum şi reuşesc să scrie uşor programe în BASIC-S, există o mare tentaţie de a intra tot mai adînc în tainele CIP-ului. Pentru aceasta aveţi la dispoziţie anumite funcţii / instrucţuni care vă permit să lucraţi direct cu memoria CIP-ului, să introduceţi informaţii direct în ea, să citiţi conţinutul ei şi să vedeţi cum sînt reprezentate caracterele etc.

APLICAŢIE PRACTICĂ AP12

1. Introduceţi:

```
 5 REM program pentru afişarea 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
```

Notaţi codurile pentru următoarele caractere (priviţi Anexa A)

caracter	cod
spaţiu (blank)	
1	
=	
A	
a	
RND	
SIN	
PEEK	
STOP	
RANDOMIZE	
COPY	

2. Introduceţi şi executaţi:

```
NEW
20 PRINT "x+y="
30 PRINT "adresa","conţinut"
40 PRINT "------","--------"
50 FOR a=23759 TO 23765
60 PRINT a,PEEK a:PRINT
70 NEXT a
RUN
```

Notaţi rezultatele şi folosind Anexa A scrieţi caracterele corespunzătoare codurilor:

adresa	conţinut	caracter
23759		
23765		

NOŢIUNI DE BAZĂ

Memoria internă a CIP-ului o putem imagina ca un fişet cu o mulţime de sertare numerotate unul după altul, începînd cu sertarul avînd numărul 0, urmat de sertarul numărul 1, apoi de cel cu numărul 2 ş.a.m.d. pînă la ultimul sertar de la sfîrşitul memoriei.
Un asemenea sertar poartă numele de LOCAŢIE DE MEMORIE şi numărul sau reprezintă ADRESA la care se poate memora orice caracter din setul de 255 pe care le cunoaşte CIP-ul.

Caracterele se memorează prin codurile cuprinse în Anexa A, dar de fapt în circuite ele sînt memorate sub forma binară prin combinaţii între două cifre:

0 - BIT zero
1 - BIT unu (bit=Binary digiT - cifră binară)

Exemple: cod zecimal	caracter	binar
32	spaţiu	00100000
43	+	00101011
70	F	01000110
102	f	01100110
241	LET	11110001

Observaţi că pentru a fi exprimat un caracter se folosesc 8 biţi (cifre binare) formînd un OCTET (în engleză: BYTE).

Reanalizaţi cuprinsul memoriei de la AP 12 - exerciţiul 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

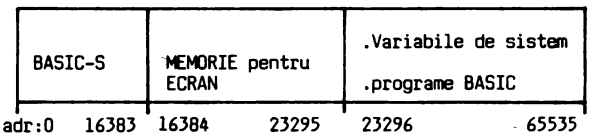
Orice octet poate fi localizat în memorie prin ADRESA sau LOCAŢIA sa.
Memoria CIP-ului poate cuprinde un număr maxim de 65536 octeţi primul avînd adresa 0, iar ultimul 65535 (pentru că numărătoarea începe cu zero).
Capacitatea memoriei se exprimă în KILOOCTEŢI / KILOBYTES 1KO = 1024 octeţi.
Memoria maximă a CIP-ului este deci de 64 KO.
Memoria calculatorului dvs. este de două feluri:

RAM (Random Acces Memory) şi
ROM (Read Only Memory)

Informaţia scrisă în ROM rămîne nealterată la decuplarea alimentării CIP-ului. În schimb, informaţia din RAM se pierde la scoaterea de sub tensiune.

Memoria cu BASIC-S

Memoria ROM este invalidată, arhitectura RAM-ului avînd următorul conţinut:



▸ Acces la o locaţie de memorie

Pentru început să vedem cum putem avea acces la memorie.
Introduceţi următorul program:

```
10 POKE 32000,128
20 PRINT PEEK 32000
25 STOP
30 POKE 32000,255
40 PRINT PEEK 3200
```

Instrucţiunea din linia 10 introduce în locaţia de memorie cu adresa 32000 numărul 128, iar instrucţiunea din linia 20 afişează conţinutul locaţiei de memorie de la adresa 32000.
Lansarea în execuţie a programului, cu comandă RUN va avea ca efect afişarea pe ecran a numărului 128.
Executînd însă programul cu RUN 30, pe ecran va fi afişată valoarea 255.
Deci s-a modificat informaţia scrisă în RAM la adresa 32000 înlocuind valoarea 128 cu 255. Funcţia POKE scrie în memorie la adresa specificată un octet de informaţie, iar funcţia PEEK extrage de la adresa specificată un octet de informaţie

Sintaxa: POKE adresa, cod
adresa=0...65535 0 <= cod <=255
PEEK adresa

RAM VIDEO

Memoria pentru ecran conţine informaţia care va fi afişată pe ecranul televizorului.
Introduceţi:
```
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 execuţie programul. Ecranul va începe să se umple cu linii negre şi în centrul lui va fi afişată adresa locaţiei de memorie ce va fi încărcată cu 255., adică un octet în care toţi biţii sînt 1. Dacă vom modifica linia 30 înlocuind astfel: 30 POKE i,15
la rularea programului pe ecran va apare o structură de linii albe şi negre, octetul 15 avînd structura 00001111.
Fiecare caracter este o matrice de 8 * 8 pixeli. Fiecare asemenea matrice are atribute de culoare care sînt poziţionate începînd de la adresa 22528 pe o lungime de 22 * 24. Fiecărui caracter îi este asociat un octet care conţine atributele.
Structura octetului de atribute poate fi determinată astfel:

atribut = INK + 8 * PAPER + 64 * BRIGHT + 128 * FLASH

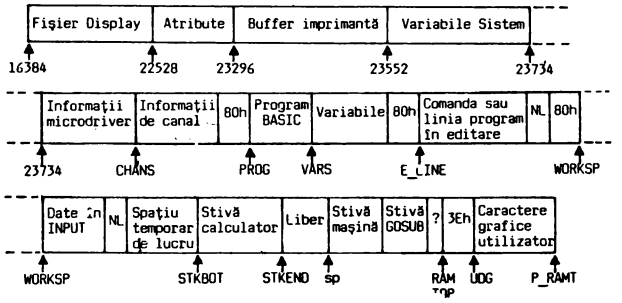
Încercaţi acum următorul program:

```
NEW
10 CLS
20 PRINT AT 10,10;"A"
30 POKE 22528+10*32+10,121
RUN
```

În linia 30 a acestui program am calculat poziţia octetului de atribute pentru caracterul tipărit prin comanda din linia 20.

VARIABILE DE SISTEM

Pentru a vorbi despre zona variabilelor de sistem să prezentăm mai în amănunt structura memoriei.



Variabilele de sistem sînt explicate în anexa F.

CLEAR

Instrucţiunea CLEAR fixează noul RAM -TOP, adică adresa maximă alocată de calculatorul dvs. pentru un program BASIC. Atenţie! La o comandă CLEAR se poate şterge conţinutul memoriei.
În plus, comanda CLEAR are următoarele efecte:
- alterează conţinuturile variabilelor de sistem;
- şterge VIDEO-RAM
- resetează poziţiile PLOT
- execută o comandă RESTORE

Sintaxa: CLEAR adresa

Exemplu:
```
10 CLEAR 40000
RUN
```

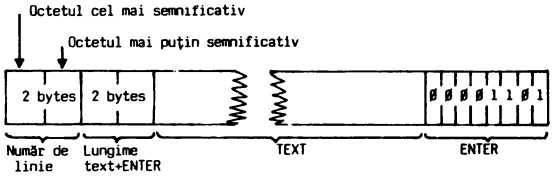
După execuţia acestei linii zona de memorie alocată pentru programul BASIC este limitată pînă la adresa 40000.

NEW

Comanda NEW şterge memoria RAM pînă la adresa RAM -TOP. Zona UDG rămîne neschimbată.

Linie BASIC

Informaţia este stocată în memorie după diferiţi algoritmi. De exemplu în continuare este prezentat modul de organizare al unei linii BASIC în memoria calculatorului.



Cod maşină

Şi acum mai mult despre programele scrise în cod maşină. Calculatorul are un procesor care execută anumite comenzi. Comenzile sînt indicate prin codurile lor. Octeţii ce reprezintă codurile se găsesc stocaţi în memoria calculatorului. De exemplu introduceţi 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 apare în stînga adresele de memorie începînd de la 32000 iar în dreapta conţinutul lor. Cei 18 octeţi între 32000 şi 32017 conţin o rutină echivalentă cu:

```
PRINT AT 10,10;1
```

Pentru a executa programul scris în cod maşină se foloseşte comanda RANDOMISE USR 32000.

RANDOMISE USR

Să încercăm acum comanda RANDOMISE USR 32000. La execuţia programului, pe ecran va apare cifra 5.
Acesta a fost un program scris în cod maşină. Programatorii experţi pot folosi un ASAMBLOR adică un program care preia mnemonicele asociate codurilor de instrucţiune şi le transformă într-un format direct executabil.

SAVE CODE

Pentru salvarea pe casetă a rutinelor scrise în cod maşină se foloseşte instrucţiunea:

```
SAVE "nume" CODE adresa,nr.octeţi
```

LOAD CODE

Pentru încărcarea programului în memorie: LOAD "nume program" CODE adresa de încărcare
Dacă doriţi verificarea unui program salvat introduceţi:

```
VERIFY "nume" CODE
```

LUCRUL CU PERIFERICE

Şi acum să vedem cum comunică un calculator cu lumea înconjurătoare.

Ce este un PORT intrare/ieşire

Microprocesorul Z80 consideră perifericele fie ca o zonă de memorie (RAM-VIDEO - pentru ecran) fie ca un port de intrare (tastatura, casetofon) sau de ieşire (BORDER, DIFUZOR, CASETOFON). Dacă la RAM-VIDEO am văzut cum puteţi avea acces, să discutăm modul de acces al CIP-ului la porturile de intrare/ieşire.
Calculatorul dvs. poate avea în configuraţie 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 şi IN

Introduceţi:
```
NEW
10 INPUT "culoarea BORDER?" ,i
20 OUT 254,i
30 GO TO 10
RUN
```

Alegînd pentru variabila i valori între 0 şi 7 vom putea modifica culoarea BORDER-ului. Observaţi 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 bîrîit la fiecare secundă. Dacă i se modifică, se va schimba corespunzător şi perioada semnalului sonor.
Instrucţiunea pentru transmiterea unei date la un port de ieşire are

SINTAXA: OUT adresa,octet adresa=adresa portului

Pentru a citi de la un port de intrare-ieşire se foloseşte instrucţiunea: IN adresa

ÎNTREBĂRI RECAPITULATIVE

Î 12.1 Care este variabila de sistem care conţine adresa ultimului OCTET DIN ZONA RAM adresabilă de procesor.

Î 12.2 Scrieţi un program în cod maşină care să încarce locaţia de memorie 17000 cu valoarea 255. Ce observaţi la executarea programului? Cum puteţi modifica atributele zonei de RAM-VIDEO care a fost modificată la execuţia programului?

Î 12.3 Salvaţi un ecran:
a) fără zone de atribute video
b) cu zone de atribute

Î 12.4 Scrieţi un program care să creeze în zona BORDER o succesiune de linii verzi şi roşii.

R RĂSPUNSURI

R 12.1 Variabila este P - RAMT şi este localizată pe 2 octeţi la adresele 23732 şi respectiv 23733.

R 12.2
```
10 REM modificare informaţie din VIDEO-RAM
20 POKE 17000,255
30 REM PROGRAM CARE URMĂREŞTE 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 APĂSĂM ORICE TASTĂ
90 NEXT i
```

Pentru a afla octetul care conţine informaţia de atribute a locaţiei la adresa 17000 din VIDEO-RAM folosim următorul program:

Locaţia căutată are adresa 22632.

R 12.3 a) Pentru zona de VIDEO-RAM fără atribute se foloseşte:
```
10 SAVE "1" CODE 16384,6144
```
b) Pentru zona de VIDEO-RAM cu atribute se foloseşte:
```
10 SAVE "2" CODE 16384,6912
```

R 12.4
```
10 REM BORDER VERDE
20 OUT 254,4
30 REM BORDER ROŞU
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 l	
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		1F	rra	rr a	
32	spaţiu	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 l,N	sra(hl)	
47	/	2F	cpl	sra a	
48	0	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	
77	M	4D	ld c,l	bit 1,l	reti
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(NN),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,e	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) caractere	99	sbc a,c	res 3,c	
154	(k) grafice	9A	sbc a,d	res 3,d	
155	(l) utilizator	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	

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	B0	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	C0	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	D0	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	F0	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	ret 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		
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+O
: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+O
OVER	(CS+SS)	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	(CS+SS)	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
	(CS+9)	CS+7
	(CS+9)	CS+6
	(CS+9)	CS+5
	(CS+9)	CS+4
	(CS+9)	CS+3
	(CS+9)	CS+2
	(CS+9)	CS+1
	(CS+9)	CS+8
!		SS+1
" (ghilimele)		SS+P
#		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+O
<		SS+R
=		SS+L
>		SS+T
?		SS+C
@ (a rond)		SS+2
	(CS+SS)	SS+Y
\ (slash inv)	(CS+SS)	SS+D
	(CS+SS)	SS+U
		SS+H
_ (subl.)		SS+O
£ (lira)		SS+X
{ (acolada)	(CS+SS)	SS+F
| (bara vert)	(CS+SS)	SS+S
} (acolada)	(CS+SS)	SS+G
~	(CS+SS)	SS+A
©	(CS+SS)	SS+P
<=		SS+Q
>=		SS+E
<>		SS+W
cursor sus	CS+7	
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*10 ↑ -39....10 ↑ 38
- Constante numerice : > format întreg ex.: 375
  > format real ex.: 375.6
  virgulă zecimală
  > format exponențial:
  ex.: 3.6245E3 (=3624,5)
  3.62E-2 (=0,0036)
  mantisa, exponent
- Constante șir 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 șir - nume variabilă urmat de $;
  ex.: A$; z1$; A$7; z1$(I,K)
- Operatori
  .aritmetici : + - * / ↑
  .relaționali : =,<,>,<=,>=,<>
  .logici : AND OR NOT
  .de concatenare șiruri: +
- Prioritatea în executarea 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-ieșire:
  > 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)
xANDy - dacă y=0 xANDy = x
y=0 xANDy = 0
x$ANDy - dacă y=0 x$ANDy = x$
y=0 x$ANDy = "" (șir nul)
ASN număr - calculează arcsinusul numărului (radiani)
ATN număr - calculează arctangenta numărului (radiani)
ATTR (linie,coloană) - dă informații despre atributele de culoare ale unui caracter
BEEP x,y - scoate o notă muzicală de înălțime y cu o durată de x secunde
BIN biți - indică un număr binar
BORDER n - colorează marginea imaginii cu culoarea dată de valoarea lui n (n=0 la 7)
BRIGHT n - produce afișarea caracterelor: normal (n=0), strălucitor (n=1)
CHR$ - dă caracterul al cărui cod este x, sub formă de șir ("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 influența variabilele stabilite prin program
CODE x$ - dă codul primului caracter din șirul x
CONT - (CONTINUE) continuă un program oprit temporar prin STOP
COPY - copiază primele 22 de linii ale ecranului pe imprimantă
COS x - calculează cosinusul unghiului x (x în radiani)
DATA c1, c2,... - introduce valori (constante numerice / șir de caractere) pentru variabilele folosite în program
Ex.:135 DATA 3.2, "ABC",43
DEF FN n(V1,V2...)=e - definește o funcție nestandard, a utilizatorului
DEF FN n$(V1$, V2$...)=e
.n este un număr dat printr-o literă
.V1, V2... sînt numere de variabile constituind parametrii funcției (maxim 26)
.e este o expresie matematică
Ex.: 10 DEF FN p(x,y)=x↑2*3
70 LET A=FN p(3,7)
DIM a(n1,n2...) - declară tablouri sau (matrici) de numere sau de șiruri
DIM a$(n1,n2...)
.a/a$ este numele tabloului
.n1,n2... sînt dimensiunile tabloului
Ex.:DIM b(14) DIM m(3,15)
DIM c$(7) DIM g$(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<=x<=255 -175<=y<=175
.z este unghiul (în radiani) pentru descrierea

```
            unei linii circulare
          Ex.:DRAW 255,0  DRAW x-45, y-80
              DRAW 60,60,PI
EXP x - calculează funcţia exponenţială la puterea x
FLASH  n  - produce  afişarea normală (n=0)  sau  clipitoare
          (n=1)
FOR  x  = n TO m - execută secvenţa de instrucţiuni  care  o
                   urmează
FOR  x = n TO m STEP y (pînă la NEXT x) de un număr  de  ori
        dat de relaţia:  (m-n)/(y-1)
          .x este numele variabilei (o literă)
          .n, m şi y sînt 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  şi  se
                    termină cu prima instrucţiune RETURN
                    După executarea subrutinei programul  se
                    reia cu instrucţiunile după GOSUB
                          Ex.: 60 GO SUB 9000
GO TO n - produce salt necondiţionat la linia n
          Ex.:IF x=0 THEN GOTO 10
IF  x THEN s - dacă expresia x este adevărată  (diferită  de
               zero) se execută  s în caz contrar se execută
               instrucţiunea care urmează după IF
               Ex.:IF a$="SF" AND b=0 PRINT b
                   IF T=0 THEN GOTO 100
IN m - citeşte din memorie octetul din portul cu adresa m
INK  n  - determină  culoarea  cu  care  vor  fi  afişate
          caracterele ce urmează (n=0 la 7)
INKEY$ - introduce caracterul primit de la tastatură
         Ex.:LET a$=INKEY$:   IF INKEY$="N" THEN GO TO 80
INPUT  "c",v1,v2...-permite  introducerea datelor  de  la  o
                    tastatură sau
INPUT LINE...
     .c   este  un  şir  de  caractere  opţionale  care
          explicitează datele ce vor fi introduse
     .v1,v2..  sînt  nume de variabile numerice sau şir  de
          caractere
     Ex.:INPUT R
         INPUT "INTRODUCEŢI PREŢUL";p
         INPUT n$
         INPUT LINE a$
INT n - dă o valoare întreagă a numărului n (<=n)
INVERSE  n - controlează  inversa afişării  caracterelor  ce
             urmează
     .dacă n=0 caracterele sînt afişate video normal
     .dacă n=1 caracterele sînt afişate video invers
LEN şir - dă numărul de caractere conţinute în şir
LET v=e - atribuie variabilei v valoarea expresiei e
     Ex.:LET i=i+1
         LET a=32
         LET N$="NUME"
         LET SUMA=4.25*A 2
LIST - listează programul, începînd cu prima linie
LIST n - listează programul începînd 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 octeţi începînd de  la
                 adresa specificată
LPRINT listă - scrie la imprimantă
MERGE  "nume" - încarcă în memorie un program de pe casetă,
                dar nu şterge în întregime pe cel vechi,  ci
                înlocuieşte numai liniile cu acelaşi număr
NEW - şterge din memorie programul şi variabilele
NEXT  v  - produce reluarea execuţiei 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  - înscrie  valoarea  octetului  n  în  portul  de
            intrare/ieşire cu adresa m
OVER n - dacă n=1,  produce supratipărirea unui nou caracter
         deasupra  celui vechi,  în poziţia 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 - opreşte execuţia programului pe o perioadă de n/50
          secunde (n=0-65535),  ecranul rămînînd neschimbat.
          Cînd se dă PAUSE 0 reluarea programului se face la
          apăsarea oricărei taste.
PEEK  adresa - citeşte din memorie şi afişează 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 poziţia de coordonate x,y
           (x=0-255 şi y=0-175) şi desemnează 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 informaţii despre punctul  de  coordonate
                (x,y);
       .valoarea  1  - dacă punctul este  vizibil  (culoarea
                       cernelii)
       .valoarea  0  - dacă  punctul  este  stins  (culoarea
                       hîrtiei)
PRINT...   - afişează  pe  ecran  constante,  variabile  sau
             expresii,  separate  în  instrucţiune  prin
             virgulă, punct şi virgulă sau apostrof.
PRINT  AT  1,c;...  - afişează începînd cu poziţia dată  de
                      coordonatele  l  şi  c  (linia=0-21  şi
                      coloana=0-31)
            Ex.:PRINT AT 11,13; "CENTRUL"
                PRINT A$(x TO 6), A$(7-n) to 6
PRINT TAB C;... - afişează începînd cu coloana C (C=0-31).
             Ex.:PRINT AT 3,1;"Capitol"; TAB24;"Pagina"
RND  - dă un număr pseudoaleator cuprins între 0 şi  1  sau
       cuprins în orice alt domeniu de valori astfel:
       x*RND              numere între 0 şi 8
       a+x*AND            numere între a şi x
       Ex.:8*RND          numere între 0 şi 8
           2.1+0.9*RND    numere între 2.1 şi 3
           5+INT(RND*20) numere întregi între 5 şi 20
RANDOMIZE   n  - generează  o  valoare  n  de  start  pentru
                 instrucţiunea RND (n=1-65535)
RANDOMIZE sau RANDOMIZE 0 - dă o valoare de start în funcţie
                            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
                instrucţiunilor 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 - restabileşte indicatorul de citire în lista de
             instrucţiuni DATA cu număr de linie n,  astfel
             încît următoarea instrucţiune 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  - instrucţiune  obligatorie  de  încheiere  a  unei
          subrutine;  provoacă  întoarcerea  execuţiei
          programului la prima instrucţiune după GOSUB n.
RUN  - lansează în execuţie programul începînd cu prima
       linie.
```

ANEXA C

RUN n - lansează în execuţie 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 execuţie începînd cu linia n.
SAVE "nume" DATA nume matrice - înregistrază pe casetă un tablou.
SAVE "nume" CODE adresă, număr - copiază pe bandă un număr de octeţi din memorie începînd de la adresa specificată.
SCREEN$ (x,y) - dă caracterul ASCII aflat în poziţie determinantă de coordonatele x (linia) si 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*PI)
SQR x - calculează radical de ordin doi din x (x > 0)
Ex.:SQR 4
SQR 0.625
SQR (n/64)
STR$ x - converteşte argumentul x într-un şir
Ex.:LET n$=STR$ 2E3
STOP - opreşte execuţia programului afişîndu-se mesajul:
STOP statement, n:m
Execuţia programului se reia cu CONTINUE
TAN x - calculează tangenta unghiului x (x în radiani)
USR adresa - lansează în execuţie o subrutină scrisă în cod maşină memorată începînd cu adresa specificată.
Ex.: INPUT "rînd"
POKE USR "a"+n,rînd
URS x$ - x este o literă între a şi u, sau un caracter grafic definit de utilizator. Se obţine adresa primului octet pentru caracterul grafic respectiv.
VAL "x" - evaluează şirul x ca o expresie numerică.
Ex:VAL "[illegible]=15
VAL$"x" - evaluează şirul x ca o expresie şir.
VERIFY "nume" - verifică dacă înregistrarea de pe banda magnetică are acelaşi conţinut cu memoria.

ANEXA D

VARIABILE DE SISTEM

Octeţii din memorie de la adresa 23552 la adresa 23733 sînt rezervaţi pentru operaţii specifice ale sistemului. Ei pot fi citiţi pentru a afla diferite lucruri despre sistem, iar cîţiva din ei pot fi şi modificaţi. Aceşti octeţi se numesc variabile de sistem şi au cîte un nume, dar nu trebuie confundaţi cu variabilele utilizate de BASIC. În cazul variabilelor formate din mai mulţi octeţi primul va fi cel mai puţin semnificativ. Variabilele de sistem sînt date în lista de mai jos. Abrevierile din coloana 1 au următoarea semnificaţie:
X - această variabilă nu poate fi modificată deoarece sistemul va funcţiona eronat
N - modificarea acestei variabile nu are un efect asupra funcţionării normale a sistemului
n - numărul de octeţi din variabilă

Tip	Adresă	Nume	Conţinut
N8	23552	KSTATE	Folosită în citirea tastaturii
N1	23560	LAST K	Reţine ultima tastă apăsată
1	23561	REPDEL	Durata (în 1/50 sec.) cît trebuie ţinută apăsată tasta pentru a se repeta, iniţial 35
1	23562	REPPER	Timpul (în 1/50 sec.) după care se repetă o tastă apăsată, iniţial 5
N2	23563	DEFADD	Adresa argumentelor funcţiilor definite de utilizator dacă una dintre ele a fost evaluată
N1	23565	K DATA	Al doilea octet pentru controlul culorii introdus de la tastatură
N2	23566	TVDATA	Controlul culorii, al lui AT şi TAB pentru TV
X38	23568	STRMS	Adresa canalului ataşat căii
2	23606	CHARS	Adresa generatorului de caractere minus 256 în mod normal în BASIC-S, dar vă puteţi genera unul şi dvs. făcînd ca CHARS să pointeze la el
1	23608	RASP	Durata sunetului de atenţionare
1	23609	PIP	Durata sunetului la apăsarea unei taste
1	23610	ERR NR	Codul de mesaj minus 1
X1	23611	FLAGS	Diferiţi indicatori de control ai sistemului BASIC
X1	23612	TVFLAG	Indicatorii asociaţi cu TV-ul
X2	23613	ERR SP	Adresa elementului din stiva maşinii 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 instrucţiunii în linie la care se sare
2	23621	PPC	Numărul liniei pentru instrucţiunea în execuţie
1	23623	SUBPPC	Numărul instrucţiunii din linie în execuţie
1	23624	BORDER	Culoarea border-ului înmulţită cu 8; conţine 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 informaţiei curente folosită pentru intrare sau ieşire
X2	23635	PROG	Adresa programului BASIC
X2	23637	NXTLIN	Adresa următoarei linii de program
X2	23639	DATADD	Adresa ultimului element din lista DATA
X2	23641	E LINE	Adresa comenzii 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 spaţiului de lucru temporar
X2	23651	STKBOT	Adresa inferioară a stivei calculator

ANEXA D

X2	23653	STKEND	Adresa de început a spaţiului liber
N1	23655	BREG	Registrul B al calculatorului
N2	23656	MEM	Adresa spaţiului folosit pentru memoria calculatorului
1	23658	FLAGS2	Alţi 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	OLDPPC	Numărul liniei la care sare CONTINUE
1	23664	OSPPC	Numărul din linie la care sare CONTINUE
N1	23665	FLAGX	Diverşi indicatori
N2	23666	STRLEN	Lungimea asigurată şirului
N2	23688	T ADDR	Adresa următorului element din tabela sintaxă
2	23670	SEED	Variabila pentru RND
3	23672	FRANES	Controlul de cadre pe 3 octeţi, cel mai puţin 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 poziţiei de scriere pe ecran
1	23680	PR CC	Octetul mai puţin semnificativ al adresei pentru noua poziţie la care se imprimă prin LPRINT
1	23681		Nefolosit
2	23682	ECHO E	Numărul coloanei şi al liniei
2	23684	DF CC	Adresa de afişare pe ecran prin PRINT
2	23686	DFCCL	Acelaşi 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	Alţi 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

I.P."Filaret" cda. 613/1992