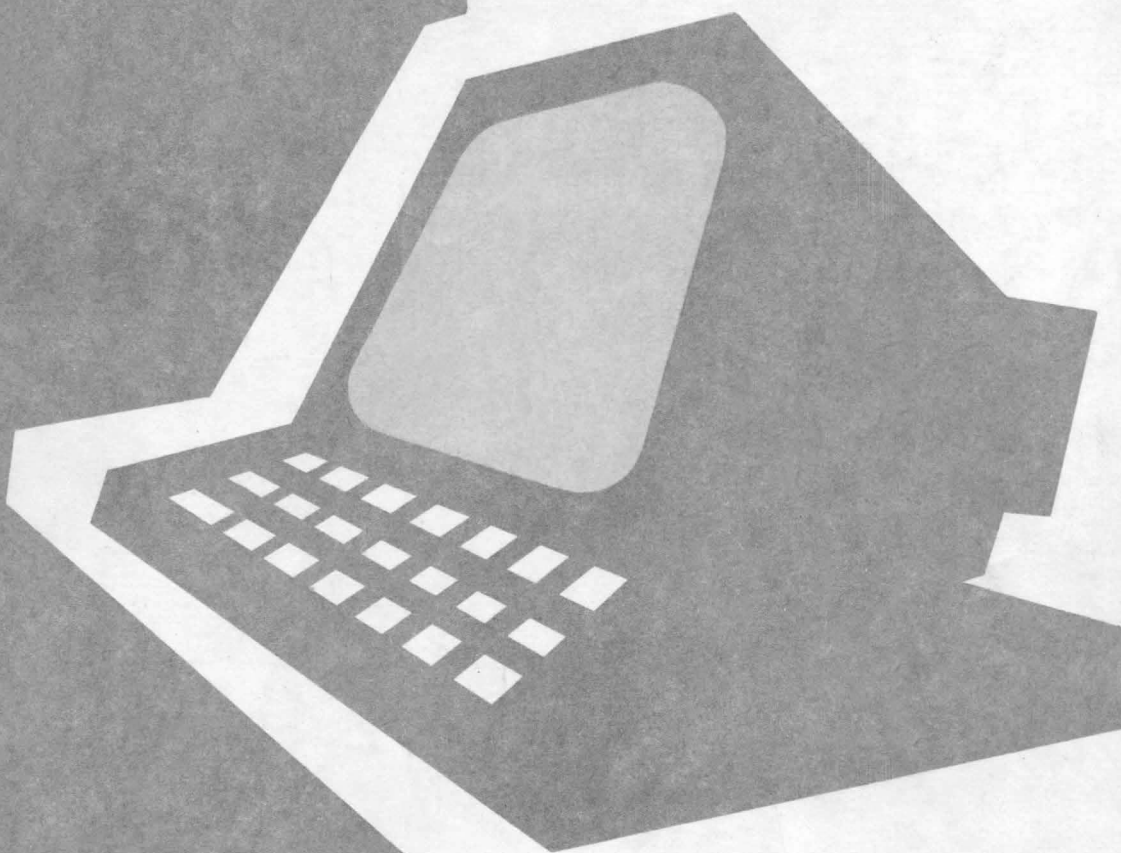


PALATUL PIONIERILOR ȘI ȘOIMILOR PATRIEI



BULETIN METODIC INFORMATICĂ

București-1989

CUPRINS

BULETIN METODIC

INFORMATICĂ

	pag.
Cuștă introductivă	5
Capitolul I Algoritmi	9
Capitolul II Programarea calculatoarelor	10
Capitolul III Limbajul BASIC	33
Capitolul IV Aplicații	88
Capitolul V Plasarea activității în cercurile de informatică	130
Capitolul VI Câteva modalități de integrare a calculatului ca mijloc de învățământ	151
Bibliografie	157

COLECTIVUL DE ELABORARE :

Ioan Albu (Cap. I, II, III și VI)

Crășnea Călinescu și

Ioan Diamandi (Cap. IV)

Constantin Hărăbor (Cap. V)

PALATUL PIONIERILOR ȘI ȘOIMILOR PATRIEI

BULETIN METODIC INFORMATICĂ

COLECTIVUL DE ELABORARE :

Ioan Albu (Cap. I, II, III și VI),
Cryseea Călinescu și
Ioan Diamandi (Cap. IV)
Constantin Hărăbor (Cap. V)

COPERTA — Alexandru Leu

CUPRINS

	<u>Pag.</u>
Cuvînt introductiv	5
Capitolul I	
Algoritmi	9
Capitolul II	
Programarea calculatoarelor	19
Capitolul III	
Limbajul BASIC	33
Capitolul IV	
Aplicații	83
Capitolul V	
Planificarea activității în cercurile de informatică	139
Capitolul VI	
Cîteva modalități de integrare a calculatorului ca mijloc de învățămînt	153
Bibliografie	157

În cadrul activității de perfecționare a conținutului învățămîntului, a nomenclaturilor de meserii și specializări, se urmărește în permanență ca școala să angajeze cunoștințe generale mai largi în domeniul fiecărei profesii, astfel încît tînrul să poată presta munci diferite în cadrul aceleiași profesii. Realizarea unei pregătiri multilaterale, integrarea învățămîntului cu cercetarea științifică, producția și practica

CUPRINS

5	Cușnt introductiv
9	Capitolul I Algoritmi
19	Capitolul II Programarea calculatoarelor
33	Capitolul III Limbajul BASIC
83	Capitolul IV Aplicații
139	Capitolul V Planificarea activității în cercurile de informatică
153	Capitolul VI Câteva modalități de integrare a calculatorului ca mijloc de învățământ
157	Bibliografie

COLECTIVUL DE ELABORARE :

Ivan Albu (Cap. I, II, III și VI),
 Cryseea Călinescu și
 Ioan Diamand (Cap. IV)
 Constantin Hărăbor (Cap. V)

COPERTA — Alexandru Leu

CUVÎNT INTRODUCATIV

Dezvoltarea în ritm înalt a bazei tehnico-materiale a societății socialiste multilateral dezvoltate, prin aplicarea celor mai noi cuceriri ale științei și tehnicii pe plan mondial, promovarea celor mai avansate tehnologii în producție, implementarea pe scară largă a informaticii în economia națională, realizarea în cele mai bune condițiuni a mărețelor obiective de dezvoltare economico-socială a țării în lumina documentelor Congresului al XIII-lea și Conferinței Naționale ale partidului impun cu necesitate o nouă viziune în legătură cu pregătirea forței de muncă, a cadrelor de specialiști, încă de pe băncile școlii generale, în strânsă legătură cu cerințele producției de bunuri materiale și spirituale din țara noastră.

Concepția Partidului Comunist Român cu privire la locul și rolul forței calificate de muncă și creație în făurirea noii orînduiri se întemeiază pe înțelegerea, în toată complexitatea sa, a interdependențelor dintre educație și societate, pe necesitatea valorificării plene a potențialului creator al celei mai însemnate avuții naționale — omul. În actualul stadiu de dezvoltare al societății noastre, mersul înainte depinde în mod hotărâtor și nemijlocit de calitatea și competența oamenilor, de capacitatea lor de a stăpîni mijloacele de producție mereu mai perfecționate, de receptivitatea la tot ce este nou și înaintat, de puterea de a ține pasul cu progresul mondial.

Așa cum sublinia tovarășul Nicolae Ceaușescu, secretarul general al partidului și la Congresul educației politice și culturii socialiste, sarcina formării omului nou revine, în primul rînd școlii, cadrul cel mai potrivit pentru pregătirea, în mod organizat și sistematic, a forței de muncă, pentru instruirea și educarea cetățenască, moral-politică și patriotică a tinerei generații.

În cadrul activității de perfecționare a conținutului învățămîntului, a nomenclatoarelor de meserii și specializări, se urmărește în permanență ca școala să asigure cunoștințe generale mai largi în domeniul fiecărei profesii, astfel încît tînărul să poată presta munci diferite în cadrul aceleiași profesii. Realizarea unei pregătiri multilaterale, integrarea învățămîntului cu cercetarea științifică, producția și practica

social-politică constituie elementele definitorii ale concepției revoluționare a partidului nostru cu privire la școala românească, care trebuie să asigure formarea unor specialiști cu înaltă calificare, care să îndeplinească cele mai complexe sarcini, atât în producția materială, cât și în sfera activității spirituale.

Din această perspectivă, și organizațiilor revoluționare de copii și tineret, celorlalți factori educaționali le revin responsabilități deosebit de importante în ceea ce privește formarea omului nou, constructor de nădejde al societății de azi și de mâine.

Pe această linie se înscriu măsurile privind îmbunătățirea activității ideologice și politico-educative în cadrul organizațiilor de copii și de tineret, eforturile Consiliului Național al Organizației Pionierilor de a extinde și diversifica activitatea cercurilor științifice și tehnico-aplicative din școli și case ale pionierilor și șoimilor patriei, de a asigura, de la o etapă la alta, un conținut superior tuturor activităților menite să contribuie la educarea comunistă a celor mai tinere generații.

În ultimii patru ani, s-a manifestat o preocupare deosebită pentru crearea bazei tehnico-materiale, pregătirea cadrelor și asigurarea funcționalității cercurilor de informatică din întreaga țară. În sprijinul acestor acțiuni vine industria românească de tehnică de calcul, creație a epocii de aur pe care o trăim — Epoca Nicolae Ceaușescu — care, răspunzând necesităților timpului prezent și, mai ales, celor ale viitorului apropiat și îndepărtat, realizează o gamă largă de calculatoare destinate nu atât specialiștilor, informaticienilor, cât mai ales cercurilor largi de utilizatori, printre care se numără și copiii.

Practica de până acum, concluziile desprinse în urma desfășurării concursului republican de informatică, evidențiază faptul că nivelul de pregătire a copiilor care activează în cercurile de informatică este strâns legat de pregătirea psihopedagogică și de specialitate a conducătorilor de cercuri.

Pentru a realiza în bune condițiuni activitatea de cerc, cadrele didactice sau instructorii îndrumători trebuie să stăpânească, în afara specialității lor, un volum apreciabil de cunoștințe în domeniul informaticii, limbajele de programare specifice micro-calculatoarelor, o varietate de tehnici de transpunere pe ecran a modelelor și desenelor, o paletă largă de metode și mijloace de lucru cu copiii, în concordanță cu orientările metodologice din pedagogia modernă.

Lucrarea de față a izvorât din necesitatea de a pune la dispoziția conducătorilor de cercuri de informatică și a cadrelor didactice care intenționează să utilizeze calculatorul electronic în procesul instructiv-educativ din școli, un material cu caracter metodic-aplicativ, accesibil și eficient atât în procesul de inițiere în domeniul informaticii, cât și al perfecționării pregătirii cadrelor. Lucrarea este concepută, atât din punct de vedere al conținutului, cât și al modului de tratare a problemelor, astfel încât să constituie un manual la îndemâna oricărui utilizator de calculator personal, adult sau copil.

Elaborarea lucrării s-a realizat prin valorificarea experienței dobândite în cadrul cercului de informatică de la Palatul Pionierilor și Șoimilor Patriei, condus de specialiști care lucrează la Institutul de Tehnică de Calcul și Informatică București, precum și a unor materiale de specia-

litate apărute în domeniul informaticii, a manualelor de utilizare a calculatoarelor de producție românească ; s-a acordat o atenție deosebită concluziilor desprinse în urma taberelor și concursurilor cu profil de informatică.

În prima parte a lucrării, la un nivel suficient de accesibil, sînt tratate problemele legate de conceptul de algoritm, proprietățile algoritmilor și structura acestora, precum și metodele de reprezentare a algoritmilor, insistîndu-se pe reprezentarea lor grafică, ce s-a dovedit deosebit de eficientă în cadrul cercurilor de informatică.

Începînd cu capitolul al II-lea, lucrarea abordează problemele programării propriu-zise a calculatoarelor. În scopul înțelegerii cu ușurință a principiilor de programare, a limbajului de programare BASIC, autorii s-au străduit să construiască, la nivelul de înțelegere al copiilor din ciclul gimnazial, un limbaj ipotetic de programare, care utilizează instrucțiuni și comenzi în limba română.

În capitolele al III-lea și al IV-lea sînt prezentate pe larg limbajul de programare BASIC și un grupaj de aplicații relativ simple, care, transpuse pe calculator ca exerciții, aduc un plus de informații în legătură cu utilizarea limbajului BASIC în rezolvarea unei game largi de probleme ce pot constitui puncte de plecare sau chiar subrutine pentru rezolvarea unor probleme complexe din activitatea social-economică.

În capitolul al V-lea, pe baza recomandărilor Consiliului Național al Organizației Pionierilor, prezentăm planuri și programe tematice orientative pentru cercurile de informatică, ce pot fi îmbogățite și îmbunătățite pe baza experienței proprii a fiecărui conducător de cerc.

Capitolul al VI-lea din lucrare se referă la cîteva modalități de integrare a calculatorului ca mijloc de învățămînt în procesul instructiv-educativ din școli și din case ale pionierilor și șoimilor patriei, care, sperăm că vor contribui la lărgirea sferei de aplicabilitate a calculatoarelor aflate în dotarea caselor pionierilor și șoimilor patriei și a unităților de învățămînt.

Fără a epuiza întreaga problemă legată de activitatea cercurilor de informatică, apreciem că prezentul buletin periodic constituie un sprijin real pentru conducătorii cercurilor de profil, fiind, în același timp, un instrument util oricărui cadru didactic, indiferent de specialitate, pentru a utiliza calculatorul electronic în procesul instructiv-educativ, ca mijloc modern și eficient de predare-învățare.

Lucrarea se înscrie ca o contribuție la eforturile mari care se fac din partea conducerii superioare de partid și de stat, a Consiliului Național al Organizației Pionierilor, a cadrelor din sistemul Organizației Pionierilor pentru perfecționarea continuă a întregii activități de educație comunistă multilaterală a tinerei generații din patria noastră.

AUTORII

Capitolul I

ALGORITMI

1.1. CONCEPTUL DE ALGORITM

Noțiunea de algoritm este o noțiune matematică foarte veche. Cuvântul „algoritm“ este de origine arabă.

Noțiunea de algoritm nu are o definiție matematică, fiind o noțiune primară. În aceeași situație se află și alte noțiuni din matematică, cum ar fi noțiunea de mulțime.

De obicei, prin algoritm se înțelege o *secvență finită și ordonată de operații*, care pornind de la o mulțime finită de *date* (inițiale), conduce (prin aplicarea în mod mecanic și uneori repetată a operațiilor) la o mulțime finită de *rezultate*.

Multe procese naturale pot fi descrise cu ajutorul algoritmilor și, de asemenea, aproape toate activitățile umane. Modul de rezolvare a unei probleme este un algoritm, căci pornind de la date (prezentate în enunțul problemei) și aplicând acestora un număr finit de operații ordonate (transformări, raționamente etc.), ajungem la soluția problemei, adică la rezultate.

Un algoritm este compus din unul sau mai mulți pași, un pas reprezentând efectuarea unei singure operații din șirul celor care alcătuiesc algoritmul.

1.2. EXEMPLE DE ALGORITMI

1.2.1. Algoritmul împărțirii întregi a două numere naturale

Se știe că împărțirea întreagă a două numere constă în efectuarea unor scăderi succesive, pînă cînd descăzutul devine mai mic decît scăzătorul.

Pentru fiecare scădere care se efectuează, descăzutul este rezultatul scăderii precedente, iar scăzătorul este împărțitorul. Rezultatul ultimei scăderi efectuate este tocmai restul împărțirii celor două numere, iar numărul de scăderi efectuate reprezintă cîtuș împărțirii.

Pașii acestui algoritm sînt constituiți de operațiile de scădere și operațiile de comparare a descăzutului cu scăzătorul. Șirul acestor operații este finit, deoarece descăzutul se micșorează cu fiecare nouă scădere, în timp ce scăzătorul rămîne neschimbat.

Să luăm spre exemplificare aflarea restului împărțirii numerelor 29 și 7. Pașii algoritmului care conduce la aflarea cîtului și restului împărțirii sînt următorii :

Numărul pasului	Operația	Descrierea pasului	Numărul scăderii
1	scădere	$29 - 7 = 22$	1
2	comparare	$22 < 7$ NU	—
3	scădere	$22 - 7 = 15$	2
4	comparare	$15 < 7$ NU	—
5	scădere	$15 - 7 = 8$	3
6	comparare	$8 < 7$ NU	—
7	scădere	$8 - 7 = 1$	4
8	comparare	$1 < 7$ DA	—

Numărul de scăderi efectuate este 4; rezultatul ultimei scăderi este 1, deci cîtul împărțirii numărului 29 la 7 este 4, iar restul este 1.

1.2.2. Algoritmul lui Euclid

Acest algoritm se folosește pentru obținerea celui mai mare divizor comun a două numere naturale.

Notînd cele două numere naturale cu m și n , vom presupune că $m > n$.

Algoritmul constă din efectuarea unui șir de împărțiri întregi pînă cînd se obține restul 0. Pentru fiecare împărțire care se efectuează, împărțitorul este restul împărțirii precedente, iar deîmpărțitul este împărțitorul din împărțirea precedentă. Împărțitorul din ultima împărțire efectuată constituie cel mai mare divizor comun al celor două numere.

Pașii acestui algoritm sînt constituiți de operațiile de împărțire și de verificare a anulării restului.

Deoarece restul unei împărțiri este mai mic decît împărțitorul, șirul de resturi al împărțirilor succesive este strict descrescător, astfel că numărul de împărțiri din algoritm este finit.

De exemplu, pentru numerele 150 și 42, pașii algoritmului care conduc la aflarea celui mai mare divizor comun sînt :

Numărul pasului	Operația	Descrierea pasului	Observații
1	împărțire	$150 : 42 = 3$ rest 24	
2	verificare	$24 = 0$	NU
3	împărțire	$42 : 24 = 1$ rest 18	
4	verificare	$18 = 0$	NU
5	împărțire	$24 : 18 = 1$ rest 6	6 este
6	verificare	$6 = 0$	NU c.m.m.d.c.
7	împărțire	$18 : 6 = 3$ rest 0	(ultimul
8	verificare	$0 = 0$	DA rest diferit de zero)

Acest algoritm poate fi redactat pentru numerele m și n astfel :

P_1 — Atribuie lui x valoarea m și lui y valoarea n .

P_2 — Calculează restul împărțirii întregi a lui x prin y și atribuie valoarea r variabilei z .

P_3 — Compară z cu 0. Dacă $z = 0$, treci la P_5 .

P_4 — Atribuie lui x valoarea y și apoi lui y valoarea lui z . Treci la P_2 .

P_5 — Atribuie lui d (c.m.m.d.c.) valoarea lui y .

Cu această convenție, algoritmul de aflare a c.m.m.d.c. al numerelor 150 și 42 poate fi scris :

	x	y	z
	150	42	24
	42	24	18
	24	18	6
	18	6	0

1.3. PROPRIETĂȚILE ALGORITMILOR

Orice algoritm trebuie să satisfacă, în principiu, următoarele cerințe :

a) **Generalitatea.** Un algoritm este util dacă rezolvă nu numai o problemă particulară, concretă, ci o întreagă clasă de probleme asemănătoare. Aceasta înseamnă că un algoritm trebuie să se poată aplica la o mulțime de sisteme de date inițiale.

Această mulțime poartă numele de domeniu de aplicabilitate al algoritmului. De exemplu, algoritmul lui Euclid se poate aplica la orice pereche de numere naturale. Vom spune deci că domeniul de aplicabili-

tate al algoritmului lui Euclid este mulțimea perechilor de numere naturale. Această proprietate este cunoscută și sub numele de universalitate.

b) *Finitudinea*, adică algoritmul să se termine după un număr finit de operații, cu obținerea rezultatului. Această proprietate se mai numește și *eficacitate*.

c) *Simplitate și claritatea*. Un algoritm trebuie să fie descris cât mai simplu și ușor de înțeles; pentru aceasta se impune utilizarea unor convenții (standarde) universal acceptate, precise și clare.

De asemenea, un algoritm trebuie să fie caracterizat printr-o descriere riguroasă, fără ambiguități a tuturor acțiunilor care urmează să se execute. Cu alte cuvinte, un algoritm, datorită caracterului său de automatism, trebuie să precizeze în mod univoc toate etapele de calcul pe care le va urma executantul algoritmului (omul sau mașina). De aceea, această proprietate se mai numește și *unicitate*.

d) *Corectitudinea*, adică să rezolve corect orice problemă din clasa de probleme la care se referă.

1.4. STRUCTURA ALGORITMILOR

Acțiunile componente ale unui algoritm se efectuează asupra unor date inițiale sau asupra unor rezultate intermediare ale operațiilor anterioare. Atât datele cât și rezultatele intermediare apar ca valori ale unor variabile. O variabilă are, în cadrul unui algoritm, o semnificație deosebită de aceea din matematică. Astfel, în timp ce în matematică o variabilă reprezintă o nedeterminată cu care se pot face operații matematice fără a fi cunoscută valoarea sa, într-un algoritm variabilele sînt utilizate pentru a denumi date sau rezultate intermediare. Deci, o variabilă este destinată să aibă o anumită valoare. Această variabilă poate fi totuși schimbată pe parcursul algoritmului. Este cazul așa-numitelor variabile de lucru. O variabilă de lucru este folosită pentru reținerea unui rezultat intermediar și poate fi refolosită pentru reținerea altui rezultat intermediar, atunci cînd rezultatul anterior nu mai este necesar pentru alte calcule.

Acțiunile unui algoritm se realizează sub forma unor operații, care constituie pașii acestuia. Operațiile care pot apărea într-un algoritm sînt de două categorii: *operații de calcul* și *operații de decizie*.

1.4.1. Operațiile de calcul

O operație de calcul constă în efectuarea calculelor indicate de o expresie simbolică, înlocuind fiecare variabilă cu valoarea sa. Rezultatul acestor calcule, adică valoarea expresiei respective, este reținut sub forma valorii unei variabile. Se spune, de obicei, că valoarea expresiei este atribuită variabilei respective.

Notăția utilizată pentru operațiile de calcul poate fi dedusă din următoarele exemple:

$$x \leftarrow y^2 + 2$$

$$S \leftarrow \text{Lungimea} \times \text{Lățimea}$$

$$M \leftarrow \frac{x_1 + x_2 + x_3}{3}$$

$$v \leftarrow \frac{S}{t}$$

Simbolul „ \leftarrow “ se citește „ia valoarea“ și este întrebuințat pentru a marca atribuirea unei valori variabilei indicate.

Într-o operație de calcul se poate atribui o nouă valoare unei variabile a cărei valoare este utilizată în cadrul calculului respectiv.

De exemplu, operația de calcul

$$y \leftarrow y^2$$

atribuie variabilei y o nouă valoare, egală cu vechea valoare ridicată la pătrat. La fel și operația $A \leftarrow A + 1$ indică faptul că variabilei A i se atribuie o valoare egală cu vechea valoare plus 1. Astfel, dacă înainte de efectuarea operației variabila A avea valoarea 10, după efectuarea operației

$$A \leftarrow A + 1$$

variabila A va avea valoarea $10 + 1 = 11$

1.4.2. Operațiile de decizie

În structura unui algoritm un rol deosebit de important îl au operațiile de decizie.

În general, prin operație de decizie se înțelege determinarea valorii logice de adevăr a unei propoziții. Propozițiile analizate de operațiile de decizie sînt propoziții enunțiative, care nu pot fi decît adevărate sau false.

De obicei, aceste propoziții enunță că un obiect are o anumită proprietate (de exemplu : „valoarea variabilei V este pozitivă“, „variabila x are ca valoare un număr întreg“, „variabila y are ca valoare un număr divizibil cu 2“ etc.).

De cele mai multe ori, propozițiile asupra cărora se aplică operația de decizie se referă la o relație între două obiecte ($x = y$, x/y etc.).

Rezultatul unei operații de decizie îl constituie valoarea „adevărat“ sau „fals“ a propoziției analizate.

În cadrul acestei operații se calculează valorile diferitelor expresii care constituie obiectele relațiilor respective, ținînd cont de valorile variabilelor care apar în aceste expresii. Astfel, propoziția

$$x + 2 > y - 1$$

are valoarea logică „adevărat“ dacă, de exemplu, variabila x are valoarea 7 iar variabila y are valoarea 2 și valoarea logică de „fals“ dacă variabilele x și y au valorile 3 și respectiv 8.

O importanță deosebită în descrierea unui algoritm o are și specificarea succesiunii de efectuare a operațiilor componente. Înlanțuirea pașilor unui algoritm poate fi indicată implicit sau explicit. Astfel, succesiunea implicită de efectuare a pașilor unui algoritm este dată de ordinea de prezentare a acestor pași în cadrul algoritmului.

Specificarea explicită a succesiunii de efectuare a unor pași apare în cazul operațiilor de decizie, unde trebuie precizat (explicit) care pas urmează să fie executat dacă rezultatul operației de decizie este „adevărat” și care pas dacă rezultatul operației este „fals”. Deoarece acești doi pași următori sînt obligatoriu diferiți, operațiile de decizie reprezintă ramificații în succesiunea de pași ai unui algoritm. Putem exemplifica cele două tipuri de operații în cadrul algoritmului împărțirii întregi (cu rest).

Să notăm variabilele acestui algoritm astfel :

D — deîmpărțitul

I — împărțitorul

C — cîțul

R — restul

Presupunînd că variabilele D și I au deja valorile corespunzătoare datelor algoritmului, pașii acestuia sînt :

P₁ C ← 0

P₂ R ← D

P₃ R < I $\left\{ \begin{array}{l} \text{da} \text{ — treci la } P_7 \\ \text{nu} \text{ — treci la } P_4 \end{array} \right.$

P₄ R ← R - I

P₅ C ← C + 1

P₆ R < I $\left\{ \begin{array}{l} \text{da} \text{ — treci la } P_7 \\ \text{nu} \text{ — treci la } P_4 \end{array} \right.$

P₇ Terminarea algoritmului.

Se observă că P₁, P₂, P₄ și P₅ specifică operațiile de calcul, în timp ce P₃ și P₆ reprezintă operații de decizie. Este posibil ca să se specifice explicit următorul pas și în cazul operațiilor de calcul. Astfel, în exemplul anterior, observînd identitatea dintre P₃ și P₆, se poate elimina P₆, specificînd la P₅ că urmează P₃.

Rezultă astfel următorul algoritm :

$$P_1 \quad C \leftarrow 0$$

$$P_2 \quad R \leftarrow D$$

$$P_3 \quad R < I \begin{cases} \text{da} & \text{— treci la } P_6 \\ \text{nu} & \text{— treci la } P_4 \end{cases}$$

$$P_4 \quad R \leftarrow R - I$$

$$P_5 \quad C \leftarrow C + 1 \text{ și treci la } P_3$$

$$P_6 \quad \text{Terminarea algoritmului.}$$

Observație : În P_3 se putea specifica numai unul dintre pași (P_3 dacă $R < I$ treci la P_6 , înțelegându-se că dacă $R \geq I$ urmează P_4).

1.5. CLASIFICAREA ALGORITMILOR

În funcție de structura lor, algoritmi pot fi împărțiți în mai multe clase.

1.5.1. Algoritmii liniari

Algoritmii liniari sînt acei algoritmi care sînt alcătuiți numai din operații de calcul. Absența operațiilor de decizie din cadrul algoritmilor liniari are ca efect execuția pașilor acestor algoritmi într-o singură succesiune (secvențial). În această categorie intră, de exemplu, algoritmul de calcul a valorii unei expresii, cum ar fi algoritmul pentru calculul valorii polinomului $ax^2 + bx + c$, prezentat mai jos

$$P_1 \quad v \leftarrow a$$

$$P_2 \quad v \leftarrow v \times x + b$$

$$P_3 \quad v \leftarrow v \times x + c$$

$$P_4 \quad \text{Terminarea algoritmului.}$$

1.5.2. Algoritmii cu ramificații

Algoritmii cu ramificații reprezintă acei algoritmi care cuprind și operații de decizie printre operațiile de calcul. În acest caz, în funcție de valorile variabilelor și de rezultatele operațiilor de decizie, pentru un algoritm cu ramificații există mai multe posibilități în ceea ce privește ordinea de execuție a pașilor săi :

a) *Algoritmii aciclici* sînt acea categorie de algoritmi cu ramificații pentru care în cadrul execuției nu se poate efectua de mai multe ori un același pas.

b) *Algoritmii ciclici* sînt algoritmi pentru care există posibilitatea repetării execuției unuia sau mai multor pași. Succesiunea de pași care

poate fi executată în mod repetat, poartă denumirea de ciclu. Numărul de repetiții ale unui ciclu poate fi fix sau variabil.

Un algoritm ciclic poate avea mai multe cicluri, acestea putând fi fi incluse unul în altul.

1.6. METODE DE REPREZENTARE A ALGORITMILOR

Experiența construirii algoritmilor, precum și rezultatele generale cunoscute despre proprietățile acestora, sugerează o serie întreagă de reguli metodologice de elaborare a algoritmilor.

Aproape toate metodologiile de construire a algoritmilor pornesc de la faptul că orice algoritm se compune din combinarea unui număr finit de algoritmi elementari sau *structuri de bază*.

Identificarea structurilor care alcătuiesc algoritmul de rezolvare a unei probleme se poate face pe baza metodei „top-down“ care constă, pe scurt, din următoarele :

- se precizează mai întâi intrările (datele) și ieșirile (rezultatele) ;
- se identifică clase sau grupe de operații ce trebuiesc executate conform enunțului problemei ;
- se rafinează operațiile deja identificate, precizându-se detalii cu privire la noile operații ; detalierea se face pînă cînd s-au identificat operațiile (structurile) elementare ale algoritmului.

Deci elaborarea algoritmului se realizează ierarhic, pornindu-se de la *general* la *particular* prin rafinări (expandări) descendente (de unde și denumirea de „top-down“).

Algoritmii pot fi reprezentați în diverse moduri echivalente și anume :

1. prin limbaj *convențional* (pseudocod) ;
2. prin scheme logice ;
3. prin limbaj algoritmic ;
4. prin tabele de decizie ;
5. prin diagrame de structură.

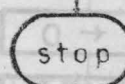
Fiecare reprezentare are avantajele și dezavantajele ei : limbajul pseudocod se remarcă prin simplitatea și naturalitatea sa ; limbajul algoritmic se remarcă prin precizie și facilitează transcrierea algoritmului într-un limbaj de programare evoluat ; tabela de decizie este un mijloc compact (prescurtat) de reprezentare a unor clase particulare de algoritmi ; diagramele de structură permit reprezentarea elementelor esențiale ale algoritmilor și descrierea compactă a acestora și ele se folosesc, de regulă, în descrierea algoritmilor complecși (cu număr mare de instrucțiuni) ; schema logică se remarcă prin aceea că este un mod sugestiv (intuitiv) de reprezentare a algoritmului.

În cele ce urmează ne vom ocupa de reprezentarea algoritmilor prin scheme logice.

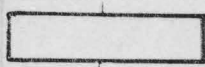
O schemă logică este alcătuită din blocuri între care se stabilesc legături orientate (săgeți). Blocurile au diferite forme grafice, în funcție de acțiunile (pașii) pe care le reprezintă.

În reprezentarea algoritmilor cu ajutorul schemelor logice se folosesc următoarele simboluri :

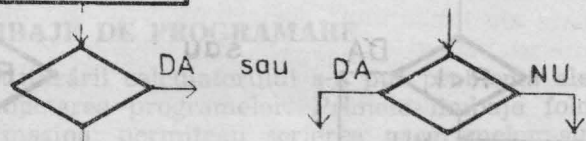
a) Bloc terminal



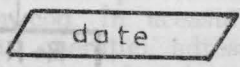
b) Bloc de calcul



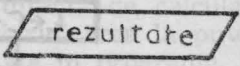
c) Bloc de decizie



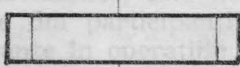
d) Bloc de intrare



e) Bloc de ieșire



f) Bloc de procedură

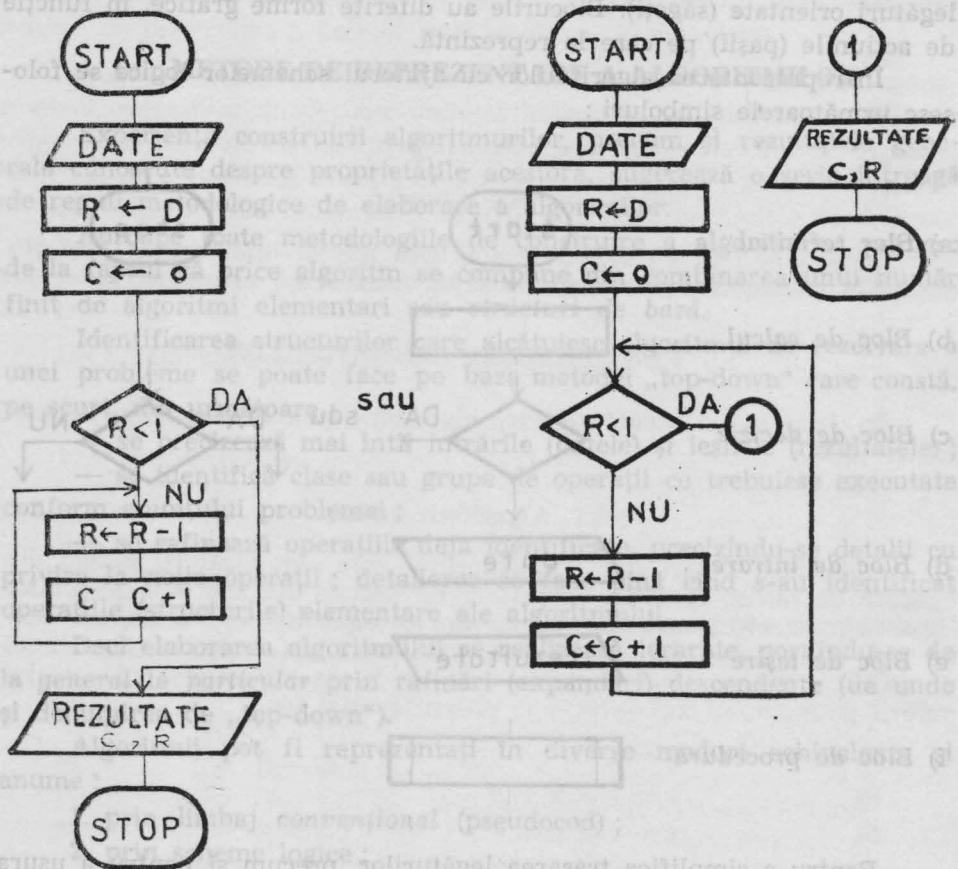


Pentru a simplifica trasarea legăturilor, precum și pentru a ușura urmărirea pașilor unui algoritm, se obișnuiește ca liniile de legătură să fie parcurse de sus în jos sau de la stînga la dreapta. Liniile de legătură care necesită să fie parcurse în alt mod trebuie desenate sub formă de săgeți. Este însă recomandabil ca toate liniile să fie trasate sub formă de săgeți.

Pentru a nu diminua claritatea unei scheme logice, se recomandă evitarea intersectării liniilor de legătură. Dacă totuși unele intersectări nu pot fi eliminate, există un simbol special, denumit conector, care permite întreruperea unei linii de legătură.

Un conector este alcătuit dintr-un cerculeț în care se înscrie o literă sau o cifră. Se presupune că între două simboluri conector care conțin aceeași literă sau aceeași cifră există o linie de legătură (netrasată).

Exemplu : Schema logică a algoritmului de aflare a restului împărțirii întregi



Fașta a simplifica țesarea legăturilor, precum și pentru utilizarea următorilor pași ai algoritmului, se recomandă să se facă legătura de la partea de sus în jos sau de la stânga la dreapta. Pentru a evita să se facă legătura de la partea de jos în sus sau de la dreapta la stânga, este necesar să se facă legătura de la partea de sus în jos sau de la stânga la dreapta. Este important să se facă legătura de la partea de sus în jos sau de la stânga la dreapta, deoarece în acest fel se evită să se facă legătura de la partea de jos în sus sau de la dreapta la stânga. Pentru a simplifica țesarea legăturilor, precum și pentru utilizarea următorilor pași ai algoritmului, se recomandă să se facă legătura de la partea de sus în jos sau de la stânga la dreapta. Este important să se facă legătura de la partea de sus în jos sau de la stânga la dreapta, deoarece în acest fel se evită să se facă legătura de la partea de jos în sus sau de la dreapta la stânga. Pentru a simplifica țesarea legăturilor, precum și pentru utilizarea următorilor pași ai algoritmului, se recomandă să se facă legătura de la partea de sus în jos sau de la stânga la dreapta. Este important să se facă legătura de la partea de sus în jos sau de la stânga la dreapta, deoarece în acest fel se evită să se facă legătura de la partea de jos în sus sau de la dreapta la stânga.

PROGRAMAREA CALCULATOARELOR

2.1. LIMBAJE DE PROGRAMARE

Încă de la începutul utilizării calculatorului s-a pus problema alegerii limbajului pentru redactarea programelor. Primele limbaje folosite, numite limbaje cod mașină, permiteau scrierea programelor sub forma unei mulțimi de comenzi pentru executarea programelor, fiecărei operații asociindu-se un cod. În această situație, fiecare instrucțiune trebuie să conțină : codul operației, adresa primului participant, adresa celui de al doilea și adresa rezultatului. Deoarece operațiile aritmetice care pot fi executate „direct“ de calculator sînt : adunarea, scăderea, înmulțirea, împărțirea și ridicarea la putere, toate operațiile mai complicate vor trebui detaliate în programe. De exemplu, pentru a calcula valoarea unei expresii aritmetice, este necesar să se descompună acest calcul în operații a cîte doi participanți, reținîndu-se rezultatele parțiale care devin participante în operațiile următoare, etc. Dacă adăugăm și faptul că utilizatorul trebuie să știe tot timpul adresele datelor de intrare, ale rezultatelor intermediare și ale celor finale, avem o imagine nu prea încurajatoare asupra programării calculatoarelor în cod mașină, operația fiind laborioasă, consumînd mult timp și energie umană.

Pentru a se depăși acest inconvenient, au fost elaborate niște limbaje evolute — *limbajele de aplicații* — care permit redactarea programelor într-o formă mult mai apropiată de limbile naturale, dar, spre deosebire de acestea, supuse unor reguli stricte. O instrucțiune scrisă într-un limbaj de aplicații este însă echivalentă cu una sau mai multe instrucțiuni în limbaj cod mașină.

Instrucțiunile scrise într-un limbaj evoluat nu pot fi executate direct de calculator. Oricare ar fi limbajul de aplicații în care se redactează un program, textul acestuia fiind numit și *text sursă*, pentru a putea fi executat, va trebui transformat într-un text echivalent — text obiect — în limbajul specific calculatorului care va executa prelucrarea. De regulă, calculatoarele sînt înzestrate cu programe care „traduc“ fiecare instrucțiune din limbajul evoluat într-un grup de instrucțiuni cod mașină, executabile de calculator. Un asemenea program se numește

program de compilare sau *compiler*. Deoarece programul ce se execută este creat automat (prin compilare) de către calculator, acest sistem de lucru cu calculatorul mai poartă numele de programare automată, iar limbajele de aplicații se mai numesc și *limbaje de programare automată*. Există mai multe limbaje de programare automată, care răspund, fiecare, cerințelor unui anumit tip general de utilizări. De exemplu, limbajul ALGOL este destinat în special prezentării metodelor de rezolvare a problemelor complexe; limbajul COBOL este destinat rezolvării problemelor cu caracter economic.

Limbajul FORTRAN este destinat rezolvării problemelor cu caracter științific și tehnic, iar limbajul BASIC este cel mai răspândit limbaj de nivel înalt din lumea microcalculatoarelor, folosit de începători, fiind utilizat într-o gamă largă de aplicații practice (cercetare, medicină, industrie, educație etc.).

Și în cazul limbajelor de programare, ca și în cazul limbajelor naturale, definirea se face pe părți componente. Vom numi „elemente componente ale limbajului” acele construcții din limbaj caracterizate de anumite particularități comune de scriere și semnificație. Deci, orice element component al unui limbaj de programare trebuie definit la două niveluri: *nivel sintactic* (reguli de formare a programelor corecte) și *nivel semantic* (regulile ce determină semnificația programelor corecte — semnificație înțeleasă ca produs asupra funcționării unui calculator).

2.2. PROGRAME ȘI SUBPROGRAME

În rezolvarea unei probleme complexe apar multe „subprobleme” care se repetă de mai multe ori în cadrul programului respectiv, ca de exemplu extragerea rădăcinii pătrate din mai multe numere sau aflarea valorilor unei funcții pentru mai multe valori etc. Fiecare din aceste subprobleme se rezolvă printr-un șir de instrucțiuni; scrierea acestui șir de fiecare dată când e nevoie duce la repetarea unei părți din munca utilizatorului și la lungirea exagerată a programului. Din acest motiv, grupul de instrucțiuni care rezolvă o subproblemă anumită poate fi scris o singură dată, în afara programului propriu-zis și memorat într-o zonă care va primi un nume. În momentul în care în program este necesară rezolvarea unei subprobleme din categoria respectivă, se *apelează setul* de instrucțiuni din zona corespunzătoare, adică se „iese” din programul principal, rezolvându-se subproblema, după care se continuă executarea programului principal. Un asemenea grup de instrucțiuni cu individualitate (denumire) proprie, scris o singură dată, în afara programului principal, care poate fi apelat ori de câte ori este necesar, poartă numele de *subprogram*.

Pentru a „funcționa”, un subprogram are nevoie de date de intrare; în urma execuției sale, el oferă date de ieșire (rezultate). Zonele de memorie în care subprogramul își găsește datele de intrare sau depune rezultatele sale sînt niște variabile numite *argumente formale*.

Un subprogram începe, de obicei, cu „descrierea” sa, deci cu inserarea denumirii sale și a denumirii argumentelor formale. Apelarea subprogramului se face cu ajutorul unei *instrucțiuni de apelare*, care tre-

buie să indice numele subprogramului apelat, precum și numele *argumentelor actuale*, adică ale variabilelor în care se găsesc datele de prelucrat și ale variabilelor în care se cere depunerea rezultatelor.

La apelare, calculatorul ia conținutul argumentelor actuale, îl trece în argumentele formale, execută programul și, în final, trece conținutul argumentelor formale în argumente actuale.

Modul de lucru cu subprograme este foarte eficient; cu timpul, fiecare utilizator își crează o bibliotecă de subprograme pe care le va folosi în cea mai mare parte a programelor sale. Mai mult, firmele producătoare de calculatoare au realizat biblioteci de *subprograme standard* (funcții standard) destinate rezolvării unor probleme care apar, de regulă, în cele mai multe programe și care se găsesc tot timpul în memorie, putând fi apelate de orice program. Spre deosebire de acestea, subprogramele scrise de utilizatori (numite subprograme utilizator) se introduc în memorie odată cu programul principal și se „șterg“ din memorie după execuția programului.

2.3. COMPONENTA UNUI PROGRAM SCRIS INTR-UN LIMBAJ DE APLICAȚII

Pentru ca un program să poată fi înțeles și folosit de către diferiți utilizatori la diferite momente, în redactarea programului se folosesc o serie de texte care dau explicații cu privire la program, la părțile sale, la variabilele utilizate etc. Aceste texte explicative nu se adresează calculatorului, deci nu sînt instrucțiuni, ci se adresează utilizatorilor; ele se numesc *comentarii* și sînt ignorate de calculator, care face doar oficiul de a le „transcrie“ pentru a putea fi la dispoziția utilizatorilor.

Celelalte texte din program se adresează calculatorului și se numesc *instrucțiuni*: ele trebuie să respecte niște reguli stricte de scriere (sintaxă), cuprinzînd o *etichetă* numerică (neobligatorie) și un text al instrucțiunii.

Instrucțiunile utilizate în elaborarea unui program pot fi:

a) *de descriere* sau definire (a fișierelor, a variabilelor, a înregistrărilor din fișiere, a subprogramelor etc.), care servesc la organizarea activității sistemului și care sînt instrucțiuni „neexecutabile“; tot instrucțiuni neexecutabile sînt și declarațiile referitoare la tipul și caracterul variabilelor;

b) *instrucțiuni executabile (operaționale)* care cuprind instrucțiunile de *intrare/ieșire* (au ca efect „citirea“ unor date și înscierea lor în memorie, scrierea unor date sau rezultate pe suporturi externe) și instrucțiuni de calcul, numite instrucțiuni de *atribuire* (dat fiind faptul că rezultatul calculului se atribuie întotdeauna unei variabile);

c) *instrucțiuni de control*, care pot fi:

— instrucțiuni de *apelare a subprogramelor*, care au ca efect „transferul“ controlului execuției de la programul principal la subprogram;

— instrucțiuni de *salt condiționat*, care realizează transferul de control al execuției în funcție de îndeplinirea unei condiții (de tip logic sau aritmetic);

— instrucțiuni de *salt necondiționat* (care realizează transferul execuției la orice punct din program dorit de utilizator) ;

— instrucțiuni de *ciclare* (care obligă la repetarea de mai multe ori a unui grup de instrucțiuni, în funcție de îndeplinirea unei condiții) ;

— instrucțiuni de *oprire* a execuției programului, situate de regulă, la sfârșitul acestuia. Dacă este vorba de un program principal, instrucțiunea de oprire duce la terminarea execuției ; dacă este vorba de un subprogram, ea are ca efect *returnarea* controlului execuției la instrucțiunea din programul apelator, imediat următoare instrucțiunii de apelare.

În concluzie, textele care compun un program într-un limbaj de aplicații se pot clasifica astfel :

texte	{	comentarii	{	a fișierelor		
		de descriere		a variabilelor		
	{	instrucțiuni	{	a înregistrărilor		
				a subprogramelor		
				de apelare a subrutinelor		
{	de control	{	de salt {	necondiționat	{	logic
			condiționat	aritmetic		
{	executabile (operaționale)	{	de ciclare (repetare)			
			de oprire			
			de intrare/ieșire	de atribuire (calcul)		

2.4. UN LIMBAJ IPOTETIC DE PROGRAMARE

În cele ce urmează vom încerca să construim un limbaj ipotetic de programare care să ajute la înțelegerea cu ușurință a limbajului de nivel înalt BASIC.

2.4.1. Instrucțiuni de bază ale limbajului

1. Să ne imaginăm că putem comanda un calculator cu comenzi scrise de tipul : citește !, scrie !, calculează ! etc. Sub această formă comenzile sînt însă incomplete. În cazul comenzii „citește“, de exemplu, calculatorul știe de unde să citească (de pe suportul extern) și ce să citească (valoarea din celula curentă a suportului extern), dar nu va ști ce să facă cu valoarea citită, în care celulă a memoriei să o plaseze. De aceea, textul comenzii trebuie completat cu numele celulei memoriei în care se depune valoarea, care am convenit că se numește „variabilă“ :

citește x

Deoarece în locul lui x poate fi orice nume de variabilă, vom defini scrierea corectă a comenzii, astfel :

citește <variabilă>

Trebuie să se rețină următoarele convenții în această definiție :

a) S-a ales numele generic „variabilă“ pentru a desemna orice celulă din memoria calculatorului ;

b) S-a folosit scrierea cuvântului „variabilă“ în paranteze unghiulare (variabilă) pentru a marca faptul că el desemnează elementele unei mulțimi (mulțimea numerelor celulelor memoriei).

O asemenea comandă reprezentată simbolic în forma de mai sus se numește *instrucțiune de citire*.

Dacă a, b, alfa, omega sînt variabile, atunci :

citește a

citește b

citește alfa

citește omega

sînt instrucțiuni de citire.

2. În mod asemănător vom defini instrucțiunea de scriere, pe care o vom nota astfel :

scrie (variabilă)

Deci „*scrie p*“ sau „*scrie nume*“ vor fi instrucțiuni de scriere, efectul asupra calculatorului fiind cel descris cînd am vorbit de unitatea de control.

3. Pentru a efectua o operație de atribuire, vom defini următoarea instrucțiune :

atribuie (variabilă) ← (expresie)

În definiția dată, variabila reprezintă numele celei din memorie în care se depune rezultatul calculului descris prin (expresie). Sintaxa pentru (expresie) este cea a expresiilor algebrice în care apar variabile, constante și evaluări de funcții elementare.

De exemplu :

atribuie x ← 7

atribuie v ← $\frac{s}{t}$

atribuie u ← 2x - 3

atribuie y ← x² - 4

atribuie s ← b × i/2

atribuie t ← tg²u - 1

sînt instrucțiuni de atribuire.

Putem folosi instrucțiunea de atribuire și pentru valori logice.

De exemplu :

atribuie x ← adevărat

atribuie z ← a v b (dacă a și b conțin valori logice)

În general, expresiile folosite în continuare pot descrie calcule atît cu valori numerice cît și logice.

4. Operația de oprire a calculatorului o vom scrie :

stop

5. Începutul oricărui program îl vom marca cu instrucțiunea :

start

În marea majoritate a limbajelor de programare, ordinea normală de execuție a operațiilor este dată chiar de ordinea în care se scriu instrucțiunile. Un grup de instrucțiuni scrise una după alta se numește secvență de instrucțiuni. De exemplu :

start

citește a

citește b

atribuie $c \leftarrow a + b$

scrie c

stop

este o secvență de instrucțiuni. O asemenea secvență de instrucțiuni o vom nota în continuare (secvență). O secvență poate cuprinde oricîte instrucțiuni (în particular și una singură). Notăția folosită va ajuta mult în reprezentarea algoritmilor în limbajul nostru. De multe ori într-o secvență apar instrucțiuni de bază de același tip, consecutive. În aceste situații vom conveni să scriem comanda o singură dată, după care vom înlănțui părțile lor variabile exact în ordinea în care au apărut în secvență. De exemplu, dacă într-o secvență apar instrucțiunile :

citește a

citește b

vom scrie prescurtat : *citește a, b* (în această ordine).

La fel pentru secvența :

scrie nume

scrie alfa

scrie adresă

vom putea scrie : *scrie nume, alfa, adresă*

În cazul atribuirilor, vom folosi semnul & („și“ comercial) pentru a lega atribuirile succesive. De exemplu, secvența :

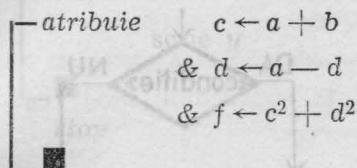
atribuie $c \leftarrow a + b$

atribuie $d \leftarrow a - b$

atribuie $f \leftarrow c^2 + d^2$

o putem scrie compact astfel :

atribuie $c \leftarrow a + b$ & $d \leftarrow a - b$ & $f \leftarrow c^2 + d^2$ sau, dacă expresiile sînt lungi :



Am marcat cu ■ plasat sub atribuie sfîrșitul secvenței celor trei atribuiri. Arcul folosit între atribuie și ■ are menirea de a evidenția unitatea celor trei instrucțiuni, faptul că ele formează o secvență de instrucțiuni de același tip.

2.4.2. Instrucțiuni de control

Instrucțiunea condițională

Să presupunem că vrem să rezolvăm cu calculatorul următoarea problemă : Fiind date două numere, să se scrie numărul cu valoarea mai mare. Logica rezolvării acestei probleme impune din partea calculatorului următoarele operații : citirea numerelor și plasarea lor în două celule de memorie, x și y ; compararea celor două valori din memorie și determinarea numărului cu valoarea cea mai mare ; scrierea valorii găsite. În cuvinte, raționamentul este următorul :

„dacă valoarea din x este mai mare decît valoarea din y , atunci scrie x , altfel (adică dacă valoarea din x este mai mică sau egală cu cea din y) scrie y “.

Forma generală pentru un asemenea raționament este :

„dacă <condiție> atunci

execută *ceva*

altfel

execută *altceva*“.

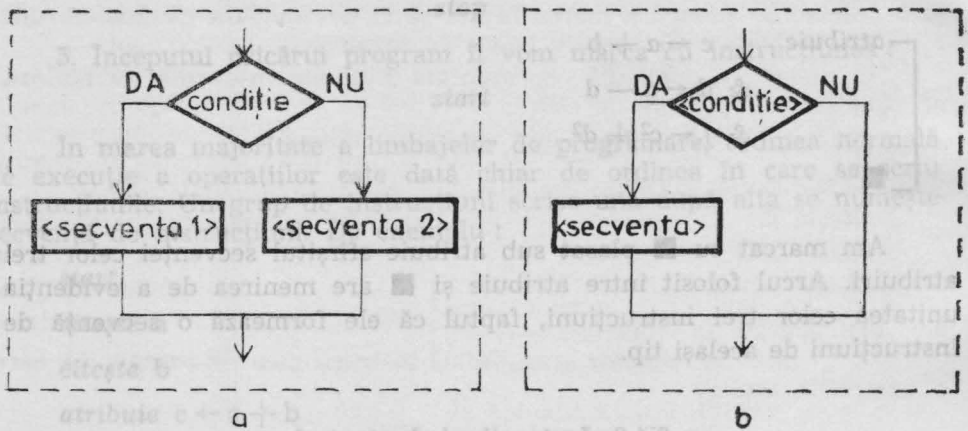
Uneori întîlnim și o formă mult mai simplă :

„Dacă <condiție> atunci

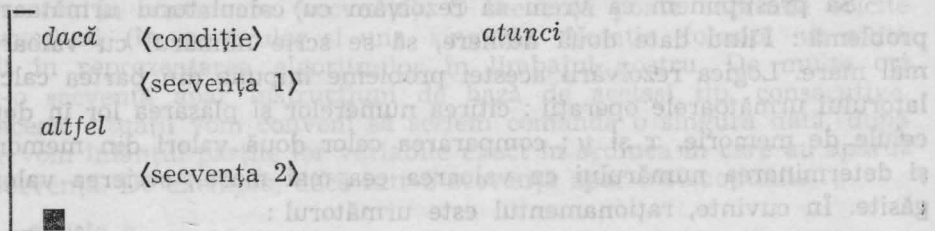
execută *ceva*“

în care a doua alternativă este vidă

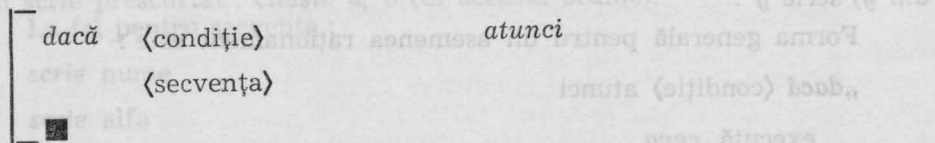
În termenii schemelor logice, acest raționament se poate reprezenta ca în figură :



Cu ajutorul instrucțiunii condiționale această operație poate fi descrisă sub forma :

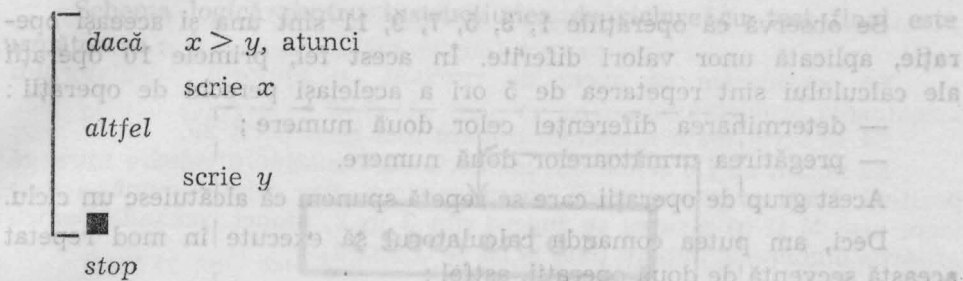


iar varianta cu a doua alternativă nulă ca efect, astfel :



În descrierile de mai sus <condiție> desemnează o expresie logică, deci o expresie care prin evaluare produce ca rezultat „adevărat” sau „fals”, cum ar fi : $a > b$, $x = 0$ $\alpha \leq 4$ etc. <secvență>, <secvență 1>, <secvență 2> desemnează secvențe de instrucțiuni de orice fel, ce pot include și instrucțiuni condiționale.

Secvența de instrucțiuni pentru rezolvarea problemei date va fi :
 citește x, y



Observăm că instrucțiunea condițională stabilește o altă ordine de execuție a operațiilor, decât cea a scrierii acestora. O asemenea instrucțiune se numește *instrucțiune de control*.

Instrucțiuni de ciclare cu condiție

Să considerăm metoda de determinare a restului împărțirii a două numere întregi prin scăderi succesive, pe care o vom aplica pentru două numere pozitive, 77 și 12. Dintr-un calcul elementar reiese că restul acestei împărțiri este 5. Acest rest îl putem determina prin scăderi succesive în următorii pași :

1. Se scade al doilea număr din primul determinînd diferența : $77 - 12 = 65$.
2. Se iau diferența obținută și al doilea număr : 65 și 12.
3. Se scade al doilea număr din diferență, determinînd o nouă diferență : $65 - 12 = 53$.
4. Se iau diferența obținută și al doilea număr.
5. Se scade al doilea număr din diferență, determinînd o nouă diferență : $53 - 12 = 41$.
6. Se iau diferența și al doilea număr : 41 și 12.
7. Se scade al doilea număr din diferență, obținînd o nouă diferență : $41 - 12 = 29$.
8. Se iau noua diferență și al doilea număr : 29 și 12.
9. Se scade al doilea număr din diferență, obținînd o nouă diferență : $29 - 12 = 17$.
10. Se iau noua diferență și al doilea număr : 17 și 12.
11. Se scade al doilea număr din diferență, obținîndu-se o nouă diferență : $17 - 12 = 5$, care este mai mică decât 12.
12. Se ia ca rezultat (restul împărțirii întregi) ultima diferență $r = 5$.

Se observă că operațiile 1, 3, 5, 7, 9, 11 sînt una și aceeași operație, aplicată unor valori diferite. În acest fel, primele 10 operații ale calculului sînt repetarea de 5 ori a aceleiași perechi de operații :

- determinarea diferenței celor două numere ;
- pregătirea următoarelor două numere.

Acest grup de operații care se repetă spunem că alcătuiesc un ciclu.

Deci, am putea comanda calculatorul să execute în mod repetat această secvență de două operații, astfel :

„ciclează“

- determinarea diferenței celor două numere ;
- pregătirea următoarelor două numere.

Această repetare s-a încheiat în exemplul nostru cînd diferența a devenit mai mică decît al doilea număr. Acest lucru poate fi specificat în comanda execuției astfel :

„ciclează“

- determinarea diferenței celor două numere ;
- pregătirea următoarelor două numere,

„pînă cînd“ diferența este mai mică decît împărțitorul.

Utilizînd instrucțiunile de bază prezentate anterior, putem scrie :

„ciclează“

atribuie $d \leftarrow x - y$

atribuie $x \leftarrow d$

„pînă cînd“ $d < y$

Am introdus astfel o nouă instrucțiune structurată, *instrucțiunea de ciclare cu test final*, care determină repetarea sub controlul unei condiții a execuției unei secvențe de instrucțiuni.

Scrierea acestei instrucțiuni este de forma :

ciclează

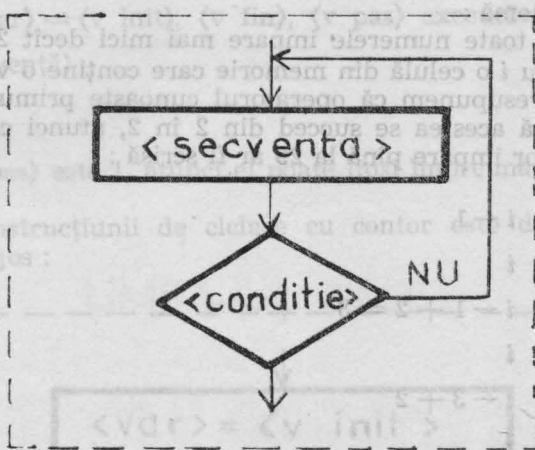
(secvență)

pînă cînd

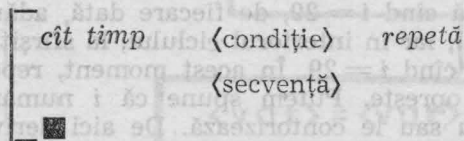
(condiție)

Operația descrisă de instrucțiune se numește *ciclu*, iar secvența de instrucțiuni formează *corpul ciclului*.

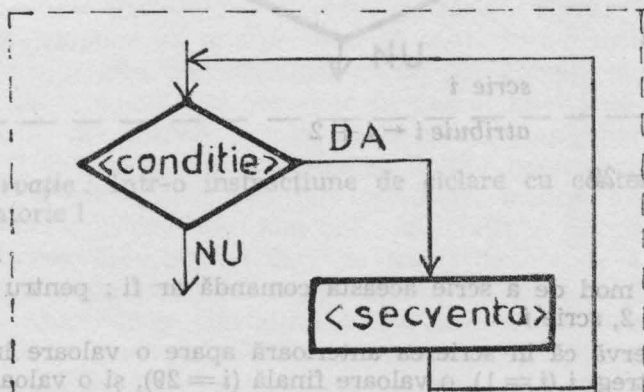
Schema logică pentru instrucțiunea de ciclare cu test final este următoarea :



În unele situații se folosește instrucțiunea de ciclare cu test inițial, care se scrie astfel :



Semnificația instrucțiunii cu test inițial este descrisă în schema logică de mai jos :



Instrucțiunea de ciclare cu contor

Să considerăm că avem de rezolvat cu ajutorul calculatorului următoarea problemă :

Să se scrie toate numerele impare mai mici decât 29.

Să notăm cu i o celulă din memorie care conține o valoare întreagă pozitivă. Dacă presupunem că operatorul cunoaște primul număr impar (care este 1) și că acestea se succed din 2 în 2, atunci comanda pentru afișarea numerelor impare pînă la 29 ar fi scrisă :

atribuie $i \leftarrow 1$

scrie i

atribuie $i \leftarrow 1 + 2 = 3$

scrie i

atribuie $i \leftarrow 3 + 2$

scrie i

.....

Procedeul poate fi continuat cu aceeași secvență de instrucțiuni pînă cînd $i = 29$. Exemplul de mai sus este tot o instrucțiune de ciclare ; (secvența) *scrie i* este repetată pînă cînd $i = 29$, de fiecare dată, adăugîndu-se 1 înaintea începerii ciclului, iar în interiorul ciclului, la sfîrșitul acestuia, valoarea lui i crește pînă cînd $i = 29$. În acest moment, repetarea instrucțiunilor din ciclu se oprește. Putem spune că i numără astfel execuțiile secvenței din ciclu sau le contorizează. De aici derivă denumirea de *contor*, dată unei asemenea variabile.

Secvența de instrucțiuni din exemplul de mai sus poate fi scrisă prescurtat astfel :

atribuie $i \leftarrow 1$

ciclează

scrie i

atribuie $i \leftarrow i + 2$

pînă $i = 29$

Un alt mod de a scrie această comandă ar fi : pentru $i = 1$ pînă la 29, cu pas 2, scrie i .

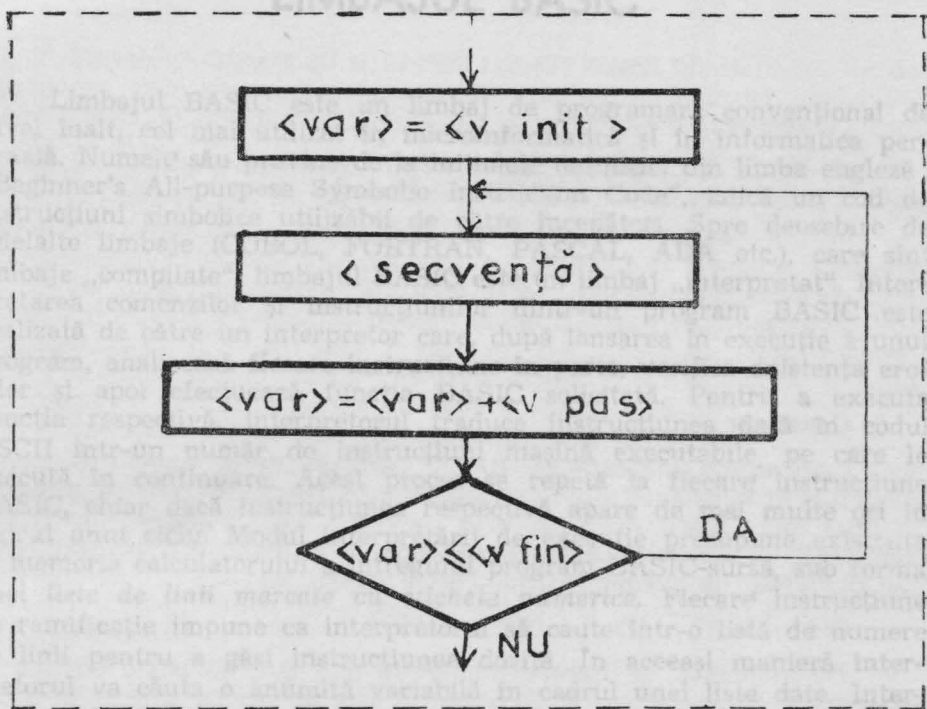
Se observă că în scrierea anterioară apare o valoare inițială dată variabilei întregi i ($i = 1$), o valoare finală ($i = 29$), și o valoare atribuită variabilei întregi „pas“ (pas = 2). Dacă vom nota cu $\langle var \rangle$ variabila întreagă numită „contor“, cu $\langle v \text{ init} \rangle$ valoarea inițială a contorului $\langle var \rangle$,

cu $\langle v \text{ fin} \rangle$ valoarea finală a contorului $\langle var \rangle$ cu $\langle v \text{ pas} \rangle$ valoarea pasului, atunci instrucțiunea de ciclare cu contor se va scrie :

pentru $\langle var \rangle = \langle v \text{ init} \rangle, \langle v \text{ fin} \rangle, \langle v \text{ pas} \rangle$ execută
 $\langle \text{secvență} \rangle$

Dacă $\langle v \text{ pas} \rangle$ este 1, atunci el poate lipsi împreună cu virgula care-l precede.

Efectul instrucțiunii de ciclare cu contor este descris în schema logică de mai jos :



Observație : Într-o instrucțiune de ciclare cu contor inițializarea este obligatorie !

LIMBAJUL BASIC

Limbajul BASIC este un limbaj de programare convențional de nivel înalt, cel mai utilizat în microinformatică și în informatica personală. Numele său provine de la inițialele definiției din limba engleză : „Beginner's All-purpose Symbolic Instruction Code“, adică un cod de instrucțiuni simbolice utilizabil de către începători. Spre deosebire de celelalte limbaje (COBOL, FORTRAN, PASCAL, ADA etc.), care sînt limbaje „compilate“, limbajul BASIC este un limbaj „interpretat“. Interpretarea comenzilor și instrucțiunilor dintr-un program BASIC este realizată de către un interpretor care, după lansarea în execuție a unui program, analizează fiecare instrucțiune în parte, verifică existența erorilor și apoi efectuează funcția BASIC solicitată. Pentru a executa funcția respectivă, interpretorul traduce instrucțiunea dată în codul ASCII într-un număr de instrucțiuni mașină executabile, pe care le execută în continuare. Acest proces se repetă la fiecare instrucțiune BASIC, chiar dacă instrucțiunea respectivă apare de mai multe ori în cadrul unui ciclu. Modul interpretării de execuție presupune existența în memoria calculatorului a întregului program BASIC-sursă, sub forma unei *liste de linii marcate cu etichete numerice*. Fiecare instrucțiune de ramificație impune ca interpretorul să caute într-o listă de numere de linii pentru a găsi instrucțiunea dorită. În aceeași manieră interpretorul va căuta o anumită variabilă în cadrul unei liste de date. Interpretorul BASIC are marele avantaj că în situația apariției unor erori, acestea pot fi eliminate prin modificarea liniilor respective din programul sursă cu ajutorul unor facilități de editare încorporate, după care se va trece direct la execuție. Mai mult decît atît, o instrucțiune poate cere să furnizeze programului date de tratat înainte de a merge mai departe. Utilizatorul, interogat de program, prin intermediul terminalului, va introduce de la claviatură aceste date, după care se va continua execuția programului. Caracterul interactiv sau „conversațional“ al limbajului BASIC face ca utilizarea calculatorului să devină extrem de atractivă, iar punerea la punct a programelor să se facă cu destulă ușurință.

Limbajul BASIC are marele avantaj de a fi foarte ușor de învățat și manipulat și, în același timp, foarte răspândit în lume. Se constată că majoritatea programelor publicate în diverse cărți, lucrări de informatică și publicații sînt scrise în acest limbaj.

Cu toate acestea, limbajul de programare BASIC are unele dezavantaje, cum sînt :

- timpul de execuție relativ mare, datorită faptului că este un limbaj interpretat și nu compilat ;
- nu se pretează întotdeauna la tehnicile programării structurate ;
- prezența în memoria calculatorului a interpretorului conduce la un volum de memorie ocupată.

În cele ce urmează vom prezenta elementele de bază legate de utilizarea limbajului BASIC extins.

ELEMENTELE LIMBAJULUI BASIC

Caractere

Cuvintele folosite în limbajul BASIC formează vocabularul limbajului. Ele se scriu după reguli precise date de sintaxa limbajului.

Caracterele întrebuințate în BASIC pentru alcătuirea cuvintelor sînt :

- literele alfabetului : A, B, C, ... Z, a, b, c, ..., z ;
- cifrele : 0, 1, 2, ..., 9 ;
- caractere speciale : +, -, *, /, =, ", >, <, ., ↑, (,), ;, :, \$, #, '.

Constante

Constantele utilizate în BASIC sînt de două tipuri : constante numerice și constante șir de caractere sau texte. În cele ce urmează, în lipsa menționării explicite, prin constantă vom înțelege o constantă numerică. Constantele numerice utilizate în limbajul BASIC sînt reale. Ele pot avea, de exemplu, următoarele exprimări :

725 ; 131.47 ; +5 ; - 0.78 ; - .3245.

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

Astfel, următoarele constante sînt corect scrise :

7.25 E2 = 725, 13147 E - 2 = 131.47, .5E + 1 = 5 ; - 78E - 2 = - 0.78.

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

M	S	E	
23	24	31	
0			

unde : M este mantisa normalizată $0.5 \leq M < 1$.

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

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

Conform cu această reprezentare internă, cel mai mic număr manipulat va fi (în modul) : $2.71051E-20$, iar cel mai mare : $9.22337E-18$. Din reprezentare se vede că sînt păstrate aproximativ 7 cifre semnificative.

Constantele șir de caractere reprezintă o secvență de simboluri alfanumerice, cuprinsă între ghilimele (""). De exemplu : „ELEVII CLASEI V-a“ ; „INTRODUCETI NUMĂRUL“ ; „1 724 KILOMETRI“ ; „XOO“ = „375F50“ ; „!!! = ?“ ; ” ” (șirul vid) etc.

Variabile

Variabilele utilizate în BASIC pot fi, ca și constantele, de două tipuri : variabile numerice (pe care le vom numi, pe scurt, variabile) și variabile șir. O variabilă numerică este reprezentată printr-o literă, urmată eventual de o cifră. Astfel, A , X , Y 1, V 5 sînt variabile numerice, pe cînd notațiile 3 A sau 2 XX nu sînt recunoscute ca variabile. Există și interpretoare BASIC care admit ca identificator (variabilă numerică) orice șir de caractere (litere sau cifre) care începe cu o literă, cum este cazul calculatorului HC-85. Variabilele pot fi simple sau indexate. Variabilele indexate reprezintă elementele unui tablou (vector sau matrice). Identificatorul (numele) tabloului trebuie să fie compus dintr-o singură literă. Indicele poate fi o constantă, o variabilă sau o expresie care urmează să fie evaluată. Dacă în urma evaluării expresiei nu se obține o valoare întregă, se va reține partea întreagă a valorii obținute.

Exemple : $X(1)$; $V(2)$; $V[ABS(R)]$; $A(I-3, J-K)$.

Variabilele șir reprezintă șiruri de caractere alfanumerice. Numele unei variabile șir este format dintr-o literă urmată de caracterul \$ (dolar). Exemple : $A\$$; $B\$$; $S\$$.

Fiind dat un șir de caractere, un subșir al lui constă în cîteva caractere consecutive conținute în el, luate în secvență. De exemplu : „LIMBAJUL BASIC“ este un subșir al subșirului „LIMBAJUL BASIC DE PROGRAMARE“.

În unele aplicații este necesară utilizarea unor subșiruri dintr-un șir desemnat de o variabilă șir. Pentru specificarea subșirurilor unei variabile șir se folosește notația :

$$x1 \text{ TO } x2$$

asociată numelui variabilei șir; $x1$ și $x2$ sînt numere întregi nenegative ce reprezintă ordinul caracterului de început, respectiv de sfîrșit, din subșir. Dacă $x1 > x2$, rezultatul este șirul vid („ “). Dacă nu se precizează începutul și/sau sfîrșitul subșirului, se iau implicit 1, respectiv lungimea șirului.

Observație : Fie $A\$ = „abcdef“$

$A\$(2TO5) = „bcde“$

$A\$(TO5) = „abcdef“$ ($1TO5$) = „abcdef“

$A\$(2TO) = „abcdef“$ (2TO6) = „bcdef“
 $A\$(TO) = „abcdef“$ (1TO6) = „abcdef“
 $A\$(3) = „abcdef“$ (3TO3) = „c“
 $A\$(3TO9)$ — dă mesaj de eroare, deoarece șirul are numai 6 caractere

Operatori

Operatorii utilizați în limbajul BASIC sînt de trei tipuri : operatori aritmetici, operatori relaționali și operatori logici. Operatorii aritmetici, în ordinea priorității în evaluare, sînt :

„↑“ — ridicare la putere
 „*“ — înmulțire
 „/“ — împărțire
 „+“ — adunare
 „-“ — scădere

Cînd se dorește schimbarea priorității de evaluare sau cînd există dubii, este bine să se utilizeze parantezele ; operațiile din interiorul parantezelor vor fi executate înaintea celor din exterior.

Operatorii relaționali sînt utilizați în unele instrucțiuni pentru a determina relația dintre valorile a două expresii :

„=“ — egalitate
 „>“ — mai mare
 „<“ — mai mic
 „≥“ — mai mare sau egal
 „≤“ — mai mic sau egal
 „<>“ — diferit (neegalitate)

Operatorii logici folosiți sînt : „OR“, „AND“, „NOT“, aceștia fiind operanzi de tip boolean (SI, SAU, SAU — EXCLUSIV, NU).

Expresii :

În alcătuirea expresiilor în limbajul BASIC pot fi utilizate următoarele funcții matematice :

$SIN(X)$ — sinus de x , unde x este un unghi exprimat în radiani ;
 $COS(X)$ — cosinus de X , unde X este exprimat în radiani ;
 $TAN(X)$ — tangentă de X , unde X este exprimat în radiani ;
 $ATN(X)$ — arctangentă de X . Rezultatul este exprimat în radiani ;
 ($-PI/2 < ATN(X) < PI/2$)

$LOG(X)$ — logaritm natural de X , X fiind un număr pozitiv (la unele interpretoare $LN(X)$) ;

$EXP(X)$ — calculează $e \rightarrow x$;

$SQR(X)$ — calculează rădăcina pătrată din x (x nenegativ) ;

$ABS(X)$ — calculează valoarea absolută a lui X ;

$INT(X)$ — calculează cel mai mare întreg $\leq X$;

RND(X) — calculează un număr aleator în intervalul (0, 1);
SGN(X) — returnează semnul lui X (1 dacă $x > 0$, 0 dacă $x = 0$;
-1 dacă $x < 0$).

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

VAL (\$) — calculează valoarea numerică a șirului tratat ca o expresie aritmetică. De exemplu: **VAL („125.6—25“)** = 100.6

VAL („2 * 3“) = 6

VAL („2“ + „* 3“) = 6

LEN (A\$) — calculează lungimea șirului specificat. De exemplu: **LEN („ELEVI SILITORI“)** = 14 (se calculează și blancurile);

STR\$ (expresie) — calculează valoarea expresiei, iar rezultatul formează un șir de caractere. Exemplu: **STR\$ (17.5—A)** = „7.5“ dacă A are valoarea 10;

CHR\$ (expresie) — calculează (determină) caracterul care are codul ASCII egal cu valoarea expresiei. De exemplu: **CHR\$ (65)** = „A“.

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

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

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

GET (X) — citește un octet de la portul cu numărul X ($0 < X < 255$). Valoarea funcției va fi un număr întreg, $0 < GET (X) < 255$. Funcția **GET** poate fi folosită în orice expresie.

PUT (X) — poate apărea numai în membrul stîng al unei instrucțiuni de atribuire. Execuția funcției constă în transmiterea la portul cu numărul X, a valorii expresiei din membrul drept al instrucțiunii de atribuire. Exemple: **10 PUT (127) = A + 17** (expresia din membrul drept este evaluată, convertită în întreg, iar cei mai puțin semnificativi 8 biți sînt trimiși la portul 127);

20 PUT (26) = GET (255) (se va citi un octet de la portul 255 și va fi trimis la portul 26).

Expresii :

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

Expresiile șir pot conține ca operanzi: constante șir, variabile șir sau subsiruri, precum și funcțiile ce au ca valori șiruri de caractere: **STR \$**, **CHR \$** și **INKEY \$**. Ca operator, în formarea expresiilor șir poate fi utilizat operatorul de concatenare, notat cu „+“. De exemplu: „2356“ + „ABCD“ = „2356ABCD“, sau **CHR \$ (65) + STR \$ (15 + 2) + „ELEV“** = „A17ELEV“.

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

Exemple : $(12 - 2 \uparrow 3) * [A(K-1) - B/LOG(K + 5)]$, sau

$$X = (A * Y + B * Z + C) / (D * Y + E * Z + F)$$

INSTRUCȚIUNI ȘI COMENZI

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

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

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

Pentru mai buna înțelegere a modului de utilizare a instrucțiunilor și comenzilor limbajului BASIC, vom prezenta folosirea acestora mai întâi în modul de lucru „imediat” și apoi în modul „program”.

Modul de lucru imediat permite utilizarea calculatorului pe post de calculator de birou pentru operații aritmetice. Rezolvarea anumitor probleme se reduce la simpla calculare a unor expresii pentru care nu este necesară scrierea unui program complet și apoi lansarea lui în execuție. De asemenea, modul „imediat” poate fi extrem de util în faza de punere la punct a programului.

PRINT. Să presupunem că dorim să efectuăm înmulțirea $105 * 25$ cu ajutorul calculatorului. Pentru aceasta va trebui ca la claviatura acestuia să tastăm :

*PRINT 105 * 25*

Comanda *PRINT* înseamnă imprimă sau afișează. Ea este utilizată, atât pentru afișarea pe ecran (display), cît și pentru imprimarea pe hîrtie, dacă sistemul folosit dispune de o imprimantă (în acest caz comanda se va scrie *LPRINT 105 * 25*). Această comandă introdusă la tastatură va fi citită de calculator numai după introducerea unui caracter terminator. Astfel de caractere sînt : *CARRIAGE RETURN (CR)* sau *ENTER*. După introducerea caracterului terminator, pe ecran va apare rezultatul : 2625, precum și semnalul că calculatorul este pregătit să primească o nouă comandă (*OK, READY, K, etc.*).

Să considerăm un alt exemplu de utilizare a instrucțiunii *PRINT*. Să presupunem că dorim să se afișeze pe ecran următorul mesaj „Acesta este un calculator de tip HC-85”. Pentru aceasta, vom introduce la tastatură următoarea comandă :

PRINT „ACESTA ESTE UN CALCULATOR DE TIP HC-85”.

După apăsarea tastei terminator (CR sau ENTER), pe ecran va apare mesajul :

ACESTA ESTE UN CALCULATOR DE TIP HC—85

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

Alte exemple ale instrucțiunii PRINT executate în mod imediat :

PRINT SQR (1225) — afișează rezultatul 35 ;

PRINT RND (X) — afișează un număr aleator subunitar ;

PRINT SIN (2 * PI) — Afișează valoarea $\frac{2}{\pi}$;

PRINT „VALOAREA RADICALULUI ESTE“ — afișează textul dintre ghilimele.

Formatul instrucțiunii PRINT este : nr. linie PRINT listă sau nr. linie PRINT unde lista poate conține : constante, variabile simple sau indexate, expresii sau șiruri. Dacă lista lipsește, se va trece la începutul liniei următoare pe display.

INSTRUCȚIUNEA DE ATRIBUIRE LET

În BASIC, la fel ca și în celelalte limbaje evolute, se definesc constante și variabile. O instrucțiune de atribuire este o instrucțiune prin care se atribuie unei variabile o anumită valoare, care este fie o constantă, fie valoarea unei alte variabile, fie valoarea unei expresii calculabile în care intervin constante și variabile. Din punct de vedere fizic această atribuire se traduce prin depunerea, la o anumită adresă din memorie, a unei valori numerice, a unei constante sau a valorii unei variabile. În cazul unei instrucțiuni de atribuire între două variabile, are loc un transfer al conținutului unei adrese în altă adresă.

Fie o variabilă V. Acestei variabile i se poate atribui o valoare numerică utilizând instrucțiunea LET astfel :

LET V = 375 (CR)

Dacă dorim să verificăm că atribuirea s-a făcut, dăm comanda PRINT V (CR) și pe ecran va apare rezultatul : 375.

În numeroase implementări ale limbajului BASIC ordinul LET poate fi omis, el fiind opțional. Așa stau lucrurile în cazul microcalculatoarelor românești FELIX-M-18, FELIX-M-118, minicalculatorul INDEPENDENT I-100, I-102 F, TPD-JUNIOR. La aceste sisteme, LET V = 375 este echivalent cu V = 375.

În cazul calculatoarelor compatibile SPECTRUM, ZX-81, ordinul LET este obligatoriu (HC-85, TIM-S), instrucțiunea V = 375 producând o eroare.

Deci formatul instrucțiunii este : nr. linie LET variabilă = expresie.

Setul de instrucțiuni al modului de lucru imediat cuprinde în afară de LET și PRINT următoarele instrucțiuni : INPUT, DIM, END, CALL, MAT. Acestea vor fi prezentate mai târziu.

COMENZILE INTERPRETORULUI BASIC

Aceste comenzi sînt extrem de necesare, deoarece fîră ele nu s-ar putea face o serie de operații de manevră, ca de exemplu : afișarea sau modificarea unui program, încărcarea unui program de pe casetă sau disketă în memorie sau salvarea pe un suport magnetic etc. Aceste comenzi pot fi privite din punct de vedere al tratării ca instrucțiuni ale modului „imediat“, datorită faptului că sînt executate imediat după introducere.

Comenzi de editare

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

Comanda LIST

Comanda LIST servește la afișarea pe terminalul utilizatorului a programului sau a unei părți a programului aflat în memorie.

Să presupunem că am introdus în memoria unui calculator următoarea secvență de instrucțiuni, numerotate (un program) :

1Ø LET A = 2Ø

2Ø LET B = 15

3Ø LET C = (A + B)/(A - B)

4Ø LET X = C ↑ (A - B)

5Ø PRINT X

introdus comanda LIST 2Ø, pe ecran ar fi apărut toate liniile cu număr va apare afișat pe ecran programul introdus în memorie. Dacă s-ar fi introdus comanda LIST 2 Ø, pe ecran ar fi apărut toate liniile cu număr mai mare sau egal cu 2Ø. De asemenea, comanda „LIST 2Ø, 4Ø“ va afișa toate liniile cu număr cuprins între 2Ø și 4Ø. Pentru a tipări la imprimantă un program aflat în memorie, comanda va fi de forma : LLIST.

Deci formatul comenzii de listare este :

LIST N 1, N 2

unde N 1 și N 2 sînt numere de linie. Execuția comenzii constă în listarea programului existent în memorie în ordinea crescătoare a numerelor de linie. Parametrii N 1 și N 2 sînt opționali. În cazul în care se specifică un singur număr de linie, se vor lista instrucțiunile ce au numărul de linie mai mare sau egal cu numărul specificat în comanda LIST. Cînd se specifică ambii parametri, se vor lista instrucțiunile care au numărul de linie cuprins între N 1 și N 2 inclusiv.

Observație : Execuția comenzii poate fi oprită acționînd tasta CTRL simultan cu C sau tasta BREAK.

Comanda DELETE

Această comandă există numai la anumite variante ale limbajului BASIC, de exemplu la BASIC-AMS implementat pe minicalculatoarele INDEPENDENT și CORAL. Ea servește la eliminarea dintr-un program a unei secvențe de instrucțiuni. Sintaxa comenzii este aceeași ca cea a comenzii LIST.

Comanda NEW

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

Comanda SCRATH (SCR)

Comanda SCR are același rol ca și NEW. La unele calculatoare, această comandă este însoțită de două tipuri de opțiuni: P, dacă este vorba de ștergerea programului (inclusiv a eventualelor subrutine în limbaj mașină), sau D, dacă este vorba de ștergerea datelor (FELIX-M-18, FELIX-M-118). Absența opțiunii semnifică ștergerea deopotrivă a programelor și datelor.

Comanda RENAME

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

Formatul comenzii: RENAME nume nou.

Modificarea unui program

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

Ștergerea liniilor unui program se poate face utilizând comanda DELETE prezentată mai sus. O altă metodă este introducerea numărului de linie, urmat de caracterul terminator CR.

Pentru inserarea de noi linii în program este necesar ca această eventualitate să fie prevăzută de la introducerea variantei inițiale, numărarea fiind făcută cu o anumită rație, de exemplu 10. Numerele noilor linii vor trebui să se încadreze între numerele liniilor programului inițial.

Modificarea unei linii se poate face prin reintroducerea ei.

Exemplu: Să presupunem că avem programul:

```
10 LET X = 10
```

```
20 LET Y = 3
```

```
30 LET Z = X ↑ 2 + Y
```



```
40 PRINT „Z =“  
50 PRINT Z
```

Dacă dorim să ștergem linia 40, atunci introducem numărul de linie, urmat de CR.

Dacă vrem să mai introducem o linie în program, atunci 25 LET T = 4 (urmat de CR).

Dacă vrem să modificăm linia 30, atunci o reintroducem modificată.

```
30 LET Z = X ↑ 2 + T ↑ Y
```

Listînd, noul program va fi :

```
10 LET X = 10
```

```
20 LET Y = 2
```

```
25 LET T = 4
```

```
30 LET Z = X ↑ 2 + T ↑ Y
```

```
50 PRINT Z
```

Comanda EDIT

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

EDIT nr. linie

La această comandă, pe ecran va apare o copie a liniei respective și numărul liniei situat pe linia următoare. Acționînd tasta de spațiu, se vor putea vizualiza caracterele succesive ale liniei. Pentru a însera caractere noi, va trebui să vizualizăm pînă la ultimul caracter care nu se schimbă. Aici vom introduce I (inserare) urmat de caracterele dorite. Exemplu : Presupunem că dorim să modificăm linia : 30 PRINT „ELEVII OBȚIN NOTE BUNE“ prin inserarea cuvîntului „HARNICI“. Cu ajutorul comenzii EDIT 30 și a tastei de blank vom opri cursorul după cuvîntul ELEVII, după care vom introduce comanda I urmată de caracterele HARNICI. După execuția acestor comenzi, linia 30 va deveni :

```
30 PRINT „ELEVII HARNICI OBȚIN NOTE BUNE“
```

Suprimarea unor caractere se face cu comanda D. Modul de utilizare este același, cu diferența că se introduce numărul de caractere de suprimat, pornind numărătoarea de la caracterul curent, de exemplu :

```
30 PRINT „ELEVII HARNICI OBȚIN NOTE BUNE“
```

Pentru a elimina caracterele „HARNICI“, cu ajutorul comenzii EDIT 30 și a tastei de spațiu vom vizualiza pînă la ultimul caracter al cuvîntului ELEVII, după care vom introduce comanda 8 D urmată de CR sau ENTER. Noua linie va deveni :

```
30 PRINT „ELEVII OBȚIN NOTE BUNE“
```


Observație : Trebuie numărate și blancurile care se suprimă. Anularea unei corecții în curs se face cu comanda Q. Programul de editare comportă și alte ordine :

- X — pentru adăugarea caracterelor la sfârșitul unei linii ;
- H — pentru înlocuirea de caractere la sfârșitul unei linii ;
- S — pentru căutarea unui anumit caracter.

COMENZI DE EXECUȚIE ȘI DE LUCRU CU PERIFERICE

În această categorie intră comenzile de lansarea programului, de pregătire a relansării și de continuare a unui program întrerupt, precum și pentru operațiile de transfer a programelor între memoria internă și cea externă (benzi, casete, diskete etc.).

RUN

Un program BASIC odată introdus în memoria internă nu poate fi lansat în execuție decît printr-o comandă RUN. La unele calculatoare, comanda admite ca parametru numărul instrucțiunii cu care va începe execuția. La altele (pe care este implementat BASIC-AMS), comanda admite drept parametru numele programului dorit a fi executat, care va fi, în prealabil, încărcat de pe un suport extern în memoria centrală. În acest caz numele este un specificator de fișier în sens AMS.

CONT (CONTINUE)

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

HELP

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

SAVE

Această comandă permite obținerea unei copii a programului aflat în memorie pe un anumit periferic. Sintaxa comenzii variază în funcție de interpretor. Astfel, pentru calculatoarele care au implementat BASIC-118, sintaxa este :

SAVE N 1, N 2

N 1 și N 2 fiind numerele de linie ce delimitează secvența care urmează a fi copiată.

Calculatoarele compatibile SINCLAIR SPECTRUM (HC-85, TIM-S, COBRA) admit ca sintaxă a acestei comenzi SAVE „Nume program“.

Calculatoarele care au interpretor BASIC-AMS, admit următoarea sintaxă pentru comanda de salvare : SAVE DX : nume fișier.

LOAD

Această comandă, complementară lui SAVE, este utilizată pentru a transfera în memoria internă un program înregistrat pe un dispozitiv periferic extern. Fără această operație prealabilă nu este posibilă execuția nici unui program. Sintaxa comenzii diferă de la un interpretor la altul (LOAD „ “ — HC-85 ; LOAD „Nume fișier“, LOAD „Nume program“).

MODUL DE LUCRU PROGRAM

Modul program este modul de lucru ce va fi utilizat pentru scrierea programelor. În modul program se introduc secvențe de instrucțiuni, fiecare instrucțiune fiind numerotată. Simplul fapt că o linie de comandă are etichetă este un indiciu pentru calculator că va intra în modul de lucru program. Așa cum am văzut, pentru o eventuală inserare de instrucțiuni este necesar ca numerotarea liniilor de program să se facă crescător cu un anumit pas (cel mai utilizat este pasul 10).

Instrucțiunile STOP și END

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

Formatul instrucțiunii STOP este :

nr. linie STOP

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

STOP AT NN

unde NN este numărul de linie al instrucțiunii STOP care a produs oprirea execuției. Într-un program pot fi utilizate mai multe instrucțiuni STOP, în funcție de necesități.

Formatul instrucțiunii END este :

nr. linie END

Spre deosebire de instrucțiunea STOP, într-un program trebuie să existe o singură instrucțiune END și să fie ultima instrucțiune din program. Este necesară utilizarea acestei instrucțiuni pentru ca interpretorul să poată determina sfârșitul programului.

Instrucțiunea INPUT

Nu toate datele pe care le utilizează un program în cadrul prelucrărilor pe care le face pot fi cunoscute în faza de introducere a sa. Apare uneori necesitatea de a furniza date programului în mod interactiv, în timpul execuției sale, prin dialog cu acesta. Pentru a se realiza această comunicare, programul afișează diverse mesaje pe ecran (cu ajutorul instrucțiunii PRINT) și citește răspunsurile la aceste mesaje cu ajutorul instrucțiunii INPUT. Atunci când se execută instrucțiunea INPUT sistemul va tipări la consolă unul din semnele : „ : “ sau „ ? “, ceea ce înseamnă că așteaptă date.

Exemplu :

```
10 PRINT „CALCULUL ARIEI PĂTRATULUI DE LATURA L“
20 PRINT „INTRODUCEȚI LATURA PĂTRATULUI“
30 INPUT L
40 LET A = L ^ 2
50 PRINT „ARIA PĂTRATULUI ESTE“ ; A
60 END
```

În linia 30 se remarcă instrucțiunea INPUT. La introducerea comenzii de lansare RUN, pe ecran va apare unul din semnele care indică așteptarea datelor. Dacă la consolă se introduce 15, atunci pe ecran va apare rezultatul : ARIA PĂTRATULUI ESTE 225. Deci instrucțiunea INPUT are formatul :

nr. linie INPUT listă de variabile

Lista de variabile poate conține variabile simple, variabile indexate sau variabile șir, separate prin virgule. Instrucțiunea INPUT este folosită pentru atribuirea de valori variabilelor din listă, valorile fiind introduse de la tastatură (consolă) în timpul execuției programului.

Dacă s-au introdus mai puține date decât numărul variabilelor din instrucțiunea INPUT, sistemul va tipări din nou unul din semnele care indică solicitarea de date, până ce toate variabilele vor primi valori. Dacă se introduc valori mai multe decât numărul variabilelor, constantele suplimentare sînt ignorate de sistem. Dacă este comisă o eroare în timpul introducerii de date, sistemul va tipări un mesaj de eroare, ceea ce obligă utilizatorul să introducă din nou linia de date.

În cazul folosirii variabilelor șir, trebuie remarcat că din linia cu constante șir corespunzătoare, introdusă la consolă, sistemul elimină spațiile (blancurile), necuprinse între ghilimele. Deci datele șir de caractere care sînt compuse din mai multe cuvinte trebuie cuprinse între ghilimele.

Exemplu : 10 INPUT A \$, B \$, M.

Dacă datele sînt BAIA MARE, ORADEA, 300, atunci ele trebuie introduse la consolă astfel : "BAIA MARE", ORADEA, 300. Ghilimelele au fost folosite pentru a păstra spațiul dintre cuvintele BAIA și MARE.

În programul scris anterior să remarcăm utilizarea în linia 50 a caracterului " ; ". În BASIC, acest caracter are un rol foarte important,

semnificând continuarea afișării pe aceeași linie fără a efectua CR. Exemplul care urmează poate fi edificator în acest sens. Să considerăm următoarele secvențe de program.

```
10 PRINT "7 × 8 ="
20 PRINT 7 * 8
30 END
```

Execuția acestui program va afișa pe ecran :

```
7 × 8 =
56
```

Pentru a ameliora forma de prezentare a rezultatului, programul se va modifica astfel :

```
10 PRINT "7 × 8 =" ; 7 * 8
20 END
```

După execuția programului, rezultatul va fi :

```
7 × 8 = 56
```

DECLARAREA ȘI CITIREA DATELOR

Așa cum s-a mai arătat, un program este o secvență de instrucțiuni capabilă să realizeze o serie de operații de prelucrare asupra unor date.

Instrucțiunile DATA și READ. Putem introduce date în cadrul unui program declarându-le cu ajutorul instrucției DATA. De exemplu, instrucțiunea :

10 DATA 1, 3, 5, 7, 9 — introduce o listă de valori care sînt primele 5 numere impare. Programul va dispune de datele introduse prin instrucțiunea DATA cu ajutorul unei instrucțiuni complementare READ.

Prezentăm în continuare cîteva exemple de utilizare a instrucțiilor DATA și READ :

Exemplul 1. 10 READ a, b, c
20 PRINT a, b, c
30 DATA 3, 5, 7, 8
40 END

Rezultatul programului va fi :

```
3 5 7
```

Exemplul 2. 10 READ A\$
20 PRINT "DATA ESTE" ; A\$
30 DATA "8 IULIE 1987"
40 END

Rezultatul programului va fi :

```
DATA ESTE 8 IULIE 1987
```


Sintaxa instrucțiunilor DATA și READ este :

nr. linie READ listă de variabile

nr. linie DATA listă de constante

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

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

Utilizarea instrucțiunilor DATA și READ comportă următoarele trei reguli :

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

2. Instrucțiunile DATA pot apare în orice linie din program.

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

Instrucțiunea RESTORE este folosită pentru a inițializa indicatorul din blocul DATA pe prima instrucțiune din program. În acest fel se poate realiza reutilizarea datelor.

Exemplu :

```
10 READ a, b
20 PRINT a b
30 RESTORE
40 READ x, y, z
50 PRINT x, y, z
60 DATA 3, 4, 5, 6
70 STOP
```

Rezultatele furnizate de acest program vor fi :

```
3 4      (a = 3, b = 4)
3 4 5    (x = 3, y = 4, z = 5)
```

Formatul instrucțiunii este :

nr. linie RESTORE n

Ea face ca instrucțiunea READ următoare să citească datele de la o instrucțiune DATA aflată în linia "n" sau după aceasta. Dacă "n" lipsește, se ia valoarea implicită 1.

Instrucțiunea REM. Este indicat ca programele să conțină comentarii explicative care să le facă mai ușor lizibile și mai clare. Aceste linii de comentariu cu caracter pur explicativ sînt introduse de instrucțiunea REM. Deci o instrucțiune REM este o notă pe care cel care realizează programul o plasează în interiorul acestuia pentru a furniza explicații unui eventual cititor.

Formatul instrucțiunii este :

nr. linie REM comentariu

Observație : Deoarece unele sisteme elimină spațiile din orice șir de caractere, este bine ca un comentariu să fie cuprins între ghilimele.

Exemplu : 100 REM "RUTINA DE AFLARE A REUNIUNII A 3 MULȚIMI".

Instrucțiunea DIM

În cazul listelor sau a tablourilor trebuie întotdeauna să rezervăm spațiu în memorie pentru elementele acestora. În acest scop se folosește instrucțiunea DIM, care are următorul format :

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

unde : — tablou 1, tablou 2 ... sînt identificatori (nume) de tablouri (vectori, matrici sau variabile șir) ;

— dimens reprezintă expresii ale căror valori definesc dimensiunile tablourilor.

Instrucțiunea DIM are următorul efect :

— rezervă spațiul necesar tabloului definit ;

— inițializarea elementelor tabloului cu 0 ;

— șterge orice tablou care are același nume cu variabilele definite prin instrucțiunea curentă.

Observație : În general pot coexista un tablou și o variabilă simplă cu același nume, fără să apară confuzii :

Exemple : a) 110 DIM A (50).

Instrucțiunea 110 declară vectorul A ca fiind un vector cu 50 de elemente.

b) 10 DIM B (5,4).

Instrucțiunea 10 declară un tablou B de tip matrice, avînd $5 \times 4 = 20$ elemente (5 linii și 4 coloane). Iată o reprezentare grafică a acestei matrici :

COLOANE :

	1	2	3	4	
linii :	1	A (1,1)	A (1,2)	A (1,3)	A (1,4)
	2	A (2,1)	A (2,2)	A (2,3)	A (2,4)
	3	A (3,1)	A (3,2)	A (3,3)	A (3,4)
	4	A (4,1)	A (4,2)	A (4,3)	A (4,4)
	5	A (5,1)	A (5,2)	A (5,3)	A (5,4)

Această matrice, sub forma ei canonică, poate fi scrisă A (i, j) cu $1 \leq i \leq 5$, $1 \leq j \leq 4$.

c) 50 DIM A \$(30)

Instrucțiunea din linia 50 declară o variabilă șir cu lungimea de 30 de caractere.

d) 150 DIM B\$(15, 20)

Această instrucțiune declară un tablou B\$ format din 15 variabile șir a câte 20 de caractere fiecare.

Observație : Tablourile numerice și cele șir pot fi declarate intercalat, cu aceeași instrucțiune DIM.

Exemplu : 10 DIM A (10, 10), B\$(100), C(16).

OPERAȚII PE ȘIRURI DE CARACTERE

În afară de valorile numerice pe care le prelucrează, interpretorul BASIC poate manipula șiruri de caractere.

Concatenarea șirurilor de caractere înseamnă de fapt adunarea sau juxtapunerea lor.

Exemplu : 10 A\$ = "ALEXE"

20 B\$ = "IOAN"

30 PRINT A\$ + B\$

40 END

Execuția acestui program va da :

ALEXEIOAN

Introducerea șirurilor se poate face în mod interactiv, în timpul execuției unui program, în aceeași manieră ca la introducerea numerelor.

Exemplu : 10 PRINT "CUM VĂ NUMIȚI ?"

20 INPUT A\$

30 PRINT "NUMELE DUMNEAVOASTRĂ ESTE" ; A\$

40 END

RUN

CUM VĂ NUMIȚI ?

VASILE

NUMELE DUMNEAVOASTRĂ ESTE VASILE

La unele interpretoare BASIC extragerea subșirurilor se poate face cu una din instrucțiunile LEFT\$, MID\$, RIGHT\$. Formatul acestor instrucțiuni și semnificația lor este :

Nr. linie LEFT\$(A\$, N) care calculează subșirul lui A\$ compus din ultimele caractere începînd cu al N-lea.

Nr. linie MID\$(A\$, N1, N2), care calculează subșirul lui A\$ compus din N2 caractere, începînd cu al N1-lea caracter din A\$.

Exemplu : 10 A\$ = "ȘCOALA NOASTRĂ ESTE FRUNTAȘĂ"

20 B\$ = LEFT\$(A\$, 14)

30 C\$ = RIGHT\$(A\$, 20)

40 D\$ = MID\$(A\$, 15, 4)

50 PRINT B\$

60 PRINT C\$

70 PRINT D\$

RUN

ȘCOALA NOASTRĂ

FRUNTAȘĂ

ESTE.

Calculatorul poate găsi cu ușurință poziția unui caracter din cadrul unui șir de caractere. Pentru aceasta se folosește instrucțiunea INSTR, care are următorul format :

Nr. linie INSTR (N, A\$, "C"), unde N reprezintă punctul de plecare din șir, iar C reprezintă caracterul căutat.

Exemplu : 10 A\$ = "ȘTEFAN CEL MARE"

20 B = LEN (A\$)

30 C = INSTR (1, A\$, "F")

40 D = INSTR (7, A\$, "A")

50 PRINT B

60 PRINT C

70 PRINT D

80 END

RUN

15

4

7

INSTRUCȚIUNI DE CONTROL

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

Instrucțiunile care modifică execuția secvențială a programului sînt instrucțiunile de control.

În limbajul BASIC, instrucțiunile de control sînt următoarele :

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

— instrucțiunile IF și ON pentru transfer condiționat ;

— instrucțiunea de ciclare FOR.

Instrucțiunea de transfer necondiționat (salt) GOTO

Există numeroase cazuri cînd este necesară reluarea repetată a execuției unui program, pentru diverse valori ale uneia sau mai multor variabile. Să presupunem că dorim să scriem un program care să calculeze diferite puteri ale numărului 2, exponentul fiind introdus la consolă. Să considerăm programul :

```
10 PRINT "SĂ SE CALCULEZE 3 LA PUTEREA"  
20 INPUT Z  
30 PRINT "REZULTATUL ESTE" ; 3 ↑ Z  
40 END
```

Execuția acestui program cere introducerea valorii lui Z. După introducere, rezultatul va fi afișat și programul se oprește. Pentru a calcula alte valori ale puterii lui 2 va trebui de fiecare dată să listăm programul și să introducem RUN. Pentru a înlătura acest inconvenient, vom introduce o instrucțiune suplimentară, care să transfere controlul la linia 10 a programului.

```
10 PRINT "SĂ SE CALCULEZE 3 LA PUTEREA"  
20 INPUT Z  
30 PRINT "REZULTATUL ESTE" ; 3 ↑ Z  
35 GOTO 10  
40 END
```

În felul acesta, ori de cîte ori execuția programului va ajunge în linia 35, se va efectua, în mod invariabil un așa-zis salt necondiționat care va face ca următoarea instrucțiune de executat să fie cea din linia 10. În felul acesta programul va cere mereu să introducem valori ale variabilei Z la consolă și va calcula valoarea $3 \uparrow Z$ (spunem că programul ciclează la infinit). Pentru a opri acest ciclu infinit, putem folosi una din comenzile BREAK, PAUSE, ARRET sau CTRL-C. Reluarea unui program stopat prin BREAK (din punctul în care a fost întrerupt) se face cu comanda CONT (CONTINUE). Formatul instrucțiunii de salt necondiționat este :

Nr. linie GOTO n

unde n este numărul liniei la care se va face transferul controlului. Saltul se poate face la orice linie din program.

Exemplu : 10 LET A = SQR ($x \uparrow 2 + y \uparrow 2$)

20.....

50 GOTO 100

80 GOTO 10

100 PRINT A

Instrucțiunea de salt condiționat IF... THEN

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

```
10 PRINT "TABLA ÎNMULȚIRII CU 9"  
20 LET A = 0  
30 PRINT "9 ori"; A; "="; 9 * A  
40 LET A = A + 1  
50 IF A <= 9 THEN 30  
60 PRINT "SFÎRȘIT PROGRAM"  
70 END
```

Instrucțiunea din linia 40 ($LET A = A + 1$) semnifică faptul că variabilei A i se atribuie o valoare egală cu vechea sa valoare plus 1. Astfel, dacă inițial A avea valoarea 0, după execuția acestei instrucțiuni A va avea valoarea $0 + 1 = 1$.

Instrucțiunea din linia 50 are următoarea semnificație : dacă (*IF*) A este mai mic sau egal cu 9, atunci (*THEN*) controlul este transferat la linia 30, iar dacă A depășește 9, controlul este dat liniei 60. Este vorba deci de un salt condiționat de valoarea variabilei A . Rezultatul execuției programului va fi :

```
RUN  
TABLA ÎNMULȚIRII CU 9  
9 ori 0 = 0  
9 ori 1 = 9  
9 ori 2 = 18  
9 ori 3 = 27  
9 ori 4 = 36  
9 ori 5 = 45  
9 ori 6 = 54  
9 ori 7 = 63  
9 ori 8 = 72  
9 ori 9 = 81  
SFÎRȘIT PROGRAM
```

Formatul instrucțiunii IF... THEN este :

Nr. linie IF exp. 1 relație exp. 2 THEN N,

unde : exp. 1, exp. 2 sînt expresii numerice sau șiruri,

N este un număr de linie

relația poate fi : =, >, <, ≤, ≥, <>

Dacă relația dintre exp. 1 și exp. 2 este adevărată, controlul va trece la instrucțiunea cu numărul N ; dacă nu este adevărată, execuția va continua cu instrucțiunea care urmează după IF.

În cazul comparației șirurilor de caractere, ambele expresii trebuie să fie de tip șir. Șirurile sînt egale dacă au aceeași lungime și conțin aceleași caractere. Un șir se consideră mai mic decît altul dacă

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

Exemplu :

"ABC" este mai mic decât "BCD"

Exemple de utilizare a instrucțiunilor IF...THEN

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

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

b) Programul de mai jos scrie pe ecran textul "ABC" și îl șterge la acționarea alternativă a unor taste

```
10 IF INKEY$ = " " THEN 10
20 PRINT "ABC"
30 IF INKEY$ = " " THEN 30
40 PRINT " "
50 GOTO 10
```

c) Acest program este de fapt un joc :

```
10 REM "GHICIȚI NUMĂRUL"
20 INPUT a
30 CLS (această comandă șterge ecranul)
40 INPUT "GHICIȚI NUMĂRUL", b
50 IF b = a, THEN 80
60 IF b < a THEN 100
70 IF b > a THEN 120
80 PRINT "REZULTAT CORECT"
90 STOP
100 PRINT "PREA MIC ! MAI ÎNCEARCĂ O DATĂ !"
110 GOTO 30
120 PRINT "PREA MARE ! MAI ÎNCEARCĂ O DATĂ !"
130 GOTO 30
140 END
```

Instrucțiunea de salt multiplu ON

În cursul derulării unui program sînt dese acele situații cînd este necesară efectuarea unui salt la o adresă care depinde de valoarea calculată a unei variabile. O asemenea situație reiese din exemplul de mai jos :

```
10 PRINT "INTRODUCEȚI UN NUMĂR ÎNTRE 1 ȘI 3"
20 INPUT N
30 IF N=1 THEN 80
40 IF N=2 THEN 100
50 IF N=3 THEN 120
60 PRINT "EROARE ! INTRODUCEȚI DIN NOU NUMĂRUL"
70 GOTO 20
80 PRINT "AȚI INTRODUS 1"
90 GOTO 130
100 PRINT "AȚI INTRODUS 2"
110 GOTO 130
120 PRINT „AȚI INTRODUS 3”
130 END
```

Putem simplifica acest program concatenînd cele trei condiții de ramificație în una singură, un salt multiplu, executat printr-un ON... GOTO, astfel :

```
10 PRINT "INTRODUCEȚI UN NUMĂR ÎNTRE 1 și 3"
20 INPUT N
30 ON N GOTO 60,80,100
40 PRINT "EROARE ! INTRODUCEȚI DIN NOU NUMĂRUL"
50 GOTO 20
60 PRINT "AȚI INTRODUS 1"
70 GOTO 110
80 PRINT "AȚI INTRODUS 2"
90 GOTO 110
100 PRINT "AȚI INTRODUS 3"
110 END
```

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

Formatul instrucțiunii ON...GOTO este :

Nr. linie ON expresie GOTO listă,
unde listă reprezintă un șir de numere de linie. Saltul se execută la cea de a i -a adresă din listă dacă valoarea expresiei este i . În cazul în care valoarea expresiei nu este număr întreg, pentru efectuarea saltului este considerată numai partea întregă.

Pentru execuția instrucțiunii ON este necesar ca valoarea expresiei să fie mai mică sau cel mult egală cu numărul elementelor din listă. Dacă acest lucru nu se întîmplă, execuția continuă cu instrucțiunea care urmează după ON.

Exemplu : 10 ON M-5 GOTO 100, 75, 90, 950, 1 000.

Pentru ca această instrucțiune să fie executată, trebuie ca M-5 să aibă o valoare între 1 și 5, altfel instrucțiunea nu are nici un efect.

Instrucțiunile de ciclare FOR și NEXT

Instrucțiunile de ciclare sînt folosite pentru execuția repetată a unor instrucțiuni din program (numite și cicluri sau bucle în program). Pentru acest lucru se puteau folosi instrucțiunile de transfer. Introducerea unor instrucțiuni speciale de ciclare s-a făcut pentru a simplifica munca de programare.

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

Pentru a înțelege mai bine utilizarea acestor instrucțiuni, să reluăm programul care afișa tabla înmulțirii cu 9.

```
10 PRINT "TABLA ÎNMULȚIRII CU 9"  
20 LET A = 0  
30 PRINT "9 ori" ; A ; "=" ; 9 * A  
40 LET A = A + 1  
50 IF A ≤ 9 THEN 30  
60 PRINT "SFÎRȘIT PROGRAM"  
70 END
```

Rezultatul execuției acestui program este echivalent cu execuția următorului program :

```
10 PRINT "TABLA ÎNMULȚIRII CU 9"  
20 FOR A = 0 TO 9  
30 PRINT "9 ori" ; A ; "=" ; 9 * A  
40 NEXT A  
50 PRINT "SFÎRȘIT PROGRAM"  
60 END
```

FOR A = 0 TO 9 înseamnă "EXECUTĂ PENTRU A CU VALORI DE LA 0 LA 9"

În mod automat calculatorul va începe execuția secvenței, atribuindu-i lui A limita inferioară (0) și va incrementa valoarea cu 1, pînă cînd A va deveni egal cu 9. Incrementarea variabilei A este indicată prin instrucțiunea "NEXT A", adică, "URMĂTOAREA VALOARE A LUI A". Calculatorul va determina noua valoare a lui A și va reveni la linia 20. Ciclul se va derula pînă cînd A va atinge valoarea 9 inclusiv. În acest moment va avea loc așa-zisa „ieșire din ciclu“, iar controlul va fi transferat la instrucțiunea 50. Ciclurile FOR-TO pot utiliza un pas diferit de 1 (pozitiv sau negativ).

Pentru a programa un ciclu cu pas diferit de 1 este suficientă utilizarea ordinului complet FOR...TO...STEP, unde STEP înseamnă "pas".

Exemple : a) 10 FOR i=1 TO 15 STEP 3

20 PRINT "i ↑ 2=" ; i ↑ 2

30 NEXT i

40 END

RUN

1 ↑ 2 = 1

4 ↑ 2 = 16

7 ↑ 2 = 49

11 ↑ 2 = 121

14 ↑ 2 = 196

b) 10 FOR N=9 TO 0 STEP -3

20 PRINT N ; "X7=" ; N*7

30 NEXT N

40 END

RUN

9 × 7 = 63

6 × 7 = 42

3 × 7 = 21

0 × 7 = 0

Formatul instrucțiunilor de ciclare este :

nr. linie FOR variabilă = exp. 1 TO exp. 2 STEP exp. 3 ;

....

(instrucțiuni ce formează bucla)

....

nr. linie NEXT variabilă.

unde :

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

— exp. 1, exp. 2, exp. 3 sînt expresii. Semnificația acestor expresii este următoarea :

exp. 1 — este valoarea inițială care se atribuie variabilei din instrucțiunea FOR ;

exp. 2 — este valoarea finală a variabilei ;

exp. 3 — este incrementul (pasul) care se adună la valoarea variabilei din instrucțiunea FOR, de fiecare dată, cînd se execută instrucțiunea din buclă.

Dacă exp. 3 nu se specifică, pasul se consideră 1. Execuția instrucțiunii NEXT constă din adunarea valorii pasului (exp. 3) la valoarea variabilei FOR și testul dacă noua valoare a variabilei nu depășește valoarea limită (exp. 2). Dacă valoarea limită este depășită, execuția buclei ia sfîrșit, următoarea instrucțiune fiind cea de după NEXT.

Exemple : a) Calculul sumei a N numere reale :

```
10 PRINT "N = "  
20 INPUT N  
30 LET S = 0  
40 FOR I = 1 TO N  
50 INPUT A (I)  
60 LET S = S + A (I)  
70 NEXT I  
80 PRINT "SUMA ESTE" ; S  
90 END
```

b) Program care afișează numărul cu valoarea maximă dintr-un șir de N numere.

```
10 PRINT "N = "  
20 INPUT N  
30 INPUT M  
40 FOR I = 2 TO N  
50 INPUT A (I)  
60 IF M >= A (I) THEN 80  
70 LET M = A (I)  
80 NEXT I  
90 PRINT "MAXIMUL ESTE" ; M  
100 END
```

c) Program care ordonează crescător (sortează) elementele unui vector $V(1) \leq V(2) \leq V(3) \leq \dots \leq V(N)$

```
10 PRINT "NUMĂRUL ELEMENTELOR ESTE"  
20 INPUT N  
30 FOR I = TO N  
40 READ A (I)  
50 NEXT I  
60 FOR I = 1 TO N - 1  
70 FOR J = 1 TO N - I  
80 IF A (J) > A (J + 1) THEN 120  
90 LET T = A (J)  
100 LET A (J) = A (J + 1)  
110 LET A (J + 1) = T  
120 NEXT J  
130 NEXT I  
140 REM "AFIȘAREA REZULTATULUI"  
150 FOR I = 1 TO N  
160 PRINT A (I)  
170 NEXT I  
180 END
```


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

```
10 FOR X =  
  [ 50 FOR Y =  
    [ 100 FOR Z =  
      .....  
      200 NEXT Z  
    250 NEXT Y  
  500 NEXT X
```

Exemplu :

```
100 FOR I = 1 TO 5  
110 FOR J = 1 TO 10  
120 READ A(I, J)  
130 NEXT J  
140 NEXT I (instrucțiunea 120 se va executa de  $5 \times 10 = 50$  ori).
```

UTILIZAREA SUBRUTINELOR

Cind este necesară efectuarea de mai multe ori într-un program a aceluiași calcule (instrucțiuni) pentru date eventual diferite, se poate folosi conceptul de subrutină (sau subprogram). Utilizarea subrutinelor conduce la o diminuare a dimensiunii programului, deoarece prelucrările se descriu o singură dată (în cadrul subrutinei) și pot fi executate de câte ori este nevoie prin apelarea subrutinei, eventual cu alte date. Folosirea funcțiilor SIN, COS, LOG, GET, PUT etc. în alcătuirea expresiilor constituie un exemplu în acest sens, deoarece funcțiile sînt definite o singură dată (în interpretor) și sînt apelate cu diferite argumente de către utilizator.

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

Instrucțiunile GOSUB și RETURN

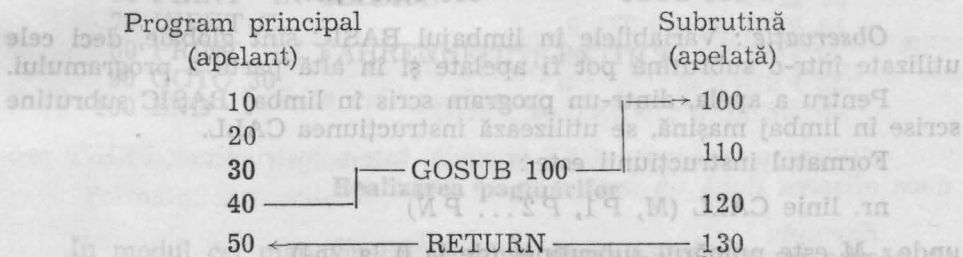
Ținînd cont de faptul că o subrutină este un program de sine stătător, programul care apelează o subrutină îl vom numi program principal. Să considerăm un exemplu simplu :

```
10 REM "ACESTA ESTE PROGRAMUL PRINCIPAL"  
20 PRINT "INTRODUCEȚI A ȘI B"  
30 INPUT A, B  
40 GOSUB 100  
50 GOTO 20  
60 END  
Subrutina este :  
100 REM "ACESTA ESTE SUBRUTINA"  
110 LET S = A + B  
120 PRINT "SUMA ESTE" ; S  
130 RETURN
```

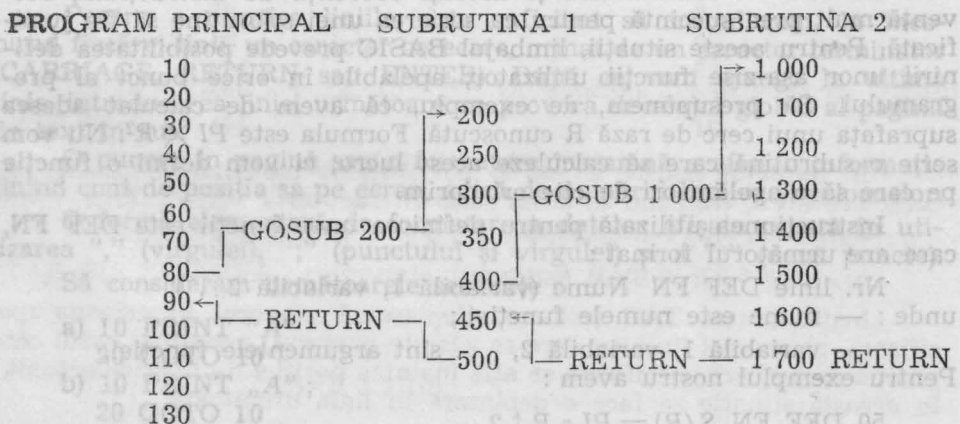
Se observă în instrucțiunea 40 faptul că s-a apelat cu comanda **GOSUB** subrutina, indicându-se numărul primei linii a acesteia. Subrutina, prin instrucțiunea 130 efectuează un **RETURN** prin care înfrumusețează programul principal în legătură cu terminarea misiunii sale.

- Deci : — apelul subrutinei se face prin ordinul **GOSUB** ;
 — revenirea din subrutină se face prin **RETURN**.

Relația dintre programul principal și subrutină poate fi prezentată intuitiv astfel :



Atunci când problemele pe care le avem de rezolvat cu calculatorul se dovedesc a fi complexe, putem realiza mai multe subrutine diferite. Este posibil ca o subrutină să apeleze la rândul ei o altă subrutină, care să apeleze o altă subrutină etc. Această tehnică se numește apel de subrutine în cascadă. Dăm în continuare o schemă care ilustrează această tehnică, pentru două subrutine.



Observație : Subrutinele trebuie să fie juxtapuse. Ele nu pot fi imbricate unele și altele, acest lucru putând genera erori.

Formatul instrucțiunilor :

nr. linie **GOSUB N**

Execuția instrucțiunii constă în transferul controlului în instrucțiunea cu numărul **N** (de regulă prima instrucțiune a unei subrutine).

nr. linie **RETURN**

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

```
Exemplu : 100 LET X = 6      500 Y = 3 * X
          110 GOSUB 500      510 LET Z = 1.2 EXP. (Y)
          120 X = 7         520 LET Y = SQR(Z + 2)
          130 GOSUB 500      530 IF Y < 100 THEN 550
          140 X = 11        540 RETURN
          150 GOSUB 500      550 PRINT X, Y
          160 STOP          560 RETURN
```

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

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

Formatul instrucțiunii este :
nr. linie CALL (M, P 1, P 2 ... P N)

unde : M este numărul subrutinei (de la 0 la 254)

P 1, P 2, ... P N sînt parametri (constante, variabile sau expresii).

Definirea funcțiilor utilizator

De multe ori se întîmplă ca în mai multe puncte ale unui program să trebuiască să folosim una și aceeași secvență de instrucțiuni, secvență mult prea succintă pentru ca scrierea unei subrutine să fie justificată. Pentru aceste situații, limbajul BASIC prevede posibilitatea definirii unor așa-zise funcții utilizator, apelabile în orice punct al programului. Să presupunem, de exemplu, că avem de calculat adesea suprafața unui cerc de rază R cunoscută. Formula este $PI \times R^2$. Nu vom scrie o subrutină care să calculeze acest lucru, ci vom defini o funcție pe care să o apelăm ori de cîte ori dorim.

Instrucțiunea utilizată pentru definirea unor funcții este DEF FN, care are următorul format :

Nr. linie DEF FN Nume (variabilă 1, variabilă 2 ...)

unde : — nume este numele funcției ;

— variabilă 1, variabilă 2, ... sînt argumentele funcției.

Pentru exemplul nostru avem :

```
50 DEF FN S(R) = PI * R ^ 2,
```

S fiind numele funcției iar R argumentul.

Exemple de programe care utilizează funcții definite :

```
a) 10 REM "CALCULUL SUPRAFETEI CERCULUI"
    20 DEF FN S(R) = PI * R ^ 2
    30 PRINT "INTRODUCETI VALOAREA RAZEI"
    40 INPUT R
    50 LET X = FN S(R)
    60 PRINT "SUPRAFAȚA CERCULUI ESTE"; X
```

Se observă că funcția este definită în instrucțiunea 20 iar valoarea acestei funcții pentru un anumit R este calculată în instrucțiunea 50.

b) Conversia din grade Celsius în grade Fahrenheit

```
10 REM "CONVERSIE CELSIUS — FAHRENHEIT"  
20 DEF FN T(X) = (9/5) * X + 32  
30 PRINT "DORIȚI O CONVERSIE ?"  
40 INPUT A$  
50 IF A$ = "NU" THEN 100  
60 PRINT "IN CELSIUS"  
70 INPUT C  
80 PRINT "IN FAHRENHEIT"; FN T(C)  
90 GOTO 30  
100 END
```

Realizarea paginării

În modul cel mai general, ecranul display-ului are un număr de 24 linii și 80 de coloane. La majoritatea minicalculatoarelor de producție românească (PRAE, a MIC, HC-85, TIM-S, TPD JUNIOR) ecranul are numai 32 coloane. Pentru a afla numărul caracterelor afișabile pe ecran este suficient să introducem de la tastatură diferite caractere pînă la umplerea totală a unei linii și apoi să le numărăm. Vom constata că după ce o linie se umple, cursorul ecranului trece automat la începutul liniei următoare.

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

O punere în pagină sau o formatare înseamnă a afișa o informație ținînd cont de poziția sa pe ecran, adică de numărul liniilor și coloanelor.

O formă elementară de formatare a datelor afișate constă în utilizarea ",," (virgulei), ":", (punctului și virgulei) și a ":", (două puncte).

Să considerăm următoarele secvențe :

- a) 10 PRINT "A"
20 GOTO 10
- b) 10 PRINT "A," ;
20 GOTO 10

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

Cel de-al doilea program (b) va umple ecranul cu caractere A. Singura diferență între cele două secvențe este utilizarea caracterului ",," la sfîrșitul liniei 10 din al doilea program. Acest caracter produce continuarea afișării, fără a sări la o linie nouă. Numai cînd linia curentă este plină, se trece automat la linia următoare.

Deci caracterul ";" semnifică continuarea afișării pe aceeași linie fără a efectua CR.

Utilizând caracterul ":" (două puncte) drept operator de concatenare, putem grupa două sau mai multe instrucțiuni pe aceeași linie.

Exemple :

- a) 10 FOR $x = 1$ TO 20 : PRINT x : NEXT x : END
- b) 10 DATA 1, 3, 5, 7, 9, 11, 13, 15
20 FOR $N = 1$ TO 6 : READ $A(N)$; PRINT $A(N)$: NEXT N
30 END
- c) 10 FOR $m = 0$ TO 6
20 FOR $n = 0$ TO m STEP 1/2
30 PRINT m ; ":" ; n ; " " ;
40 NEXT n ; NEXT m : END

Utilizarea caracterului ";" (virgulă) într-o instrucțiune PRINT produce afișarea după un spațiu de 15 caractere.

Exemplu :

- ```
10 FOR $n = 1$ TO 21 STEP 4
20 PRINT n , $n \uparrow 2$
30 NEXT n
40 END
```

Această secvență va produce afișarea următorului tabel :

|    | 15 caractere |     |
|----|--------------|-----|
| 1  |              | 1   |
| 5  |              | 25  |
| 9  |              | 81  |
| 13 |              | 169 |
| 17 |              | 289 |
| 21 |              | 441 |

În limbajul BASIC există și posibilitatea tabulării la numărul de caractere dorit, utilizând în acest sens instrucțiunea PRINT TAB.

Formatul instrucțiunii este :

nr. linie PRINT TAB (coloana)

Instrucțiunea TAB (coloana) deplasează cursorul în coloana specificată, pe aceeași linie pe care se găsește cursorul, exceptând cazul când poziția de tipărire specificată se află înaintea poziției de tipărire actuală ; în această situație se face o deplasare în linia următoare.

Exemplu :

- ```
10 PRINT "X" ; TAB (8) "X  $\uparrow$  2" ; TAB (18) ; "X  $\uparrow$  3" ;  
TAB (25) ; "X  $\uparrow$  4"  
20 FOR  $X = 1$  TO 5  
30 PRINT  $X$  ; TAB (8) ;  $X \uparrow 2$  ; TAB (18) ;  $X \uparrow 3$  ;  
TAB (25) ;  $X \uparrow 4$   
40 NEXT  $X$   
50 END  
RUN
```


X	X ↑ 2	X ↑ 3	X ↑ 4
1	1	1	1
2	4	8	16
3	9	27	81
4	16	64	256
5	25	125	625
(col 1)	(col 8)	(col 18)	(col 25)

Observație : Elementele de tipărire care urmează instrucțiunilor TAB sînt de obicei terminate cu caracterul ”;”. Dacă s-ar folosi ”;” sau nimic, cursorul, după ce este poziționat, se deplasează.

La realizarea paginărilor se poate folosi și instrucțiunea AT.

Formatul instrucțiunii :

nr. linie AT linie, coloană

Efectul acestei instrucțiuni este deplasarea cursorului (locul unde va fi tipărit următorul element) la linia și la coloana specificată.

Ca și în cazul instrucțiunii TAB, elementele de tipărire care urmează instrucțiunii AT sînt de obicei terminate cu caracterul ”;”. Tipărind cu AT într-o poziție deja scrisă, ultima tipărire o anulează pe prima.

Exemple :

a) 10 PRINT AT 12, 16 ; ”*” va imprima un asterisc în centrul ecranului ;

b) PRINT AT 3, 8 ; ”DRAGĂ PRIETENE”, va tipări pe linia 3 la mijlocul ei, textul dintre ghilimele.

c) PRINT TAB 30 ; 1 ; TAB (12) ; INDEX” ; AT 3, 1 ; „CAPITOL” ; TAB 24 ; ”PAGINA” reprezintă începutul paginii unei cărți ;

```
d) 10 CLS
    20 FOR X = 1 TO 30
    30 PRINT TAB (X ↑ 2/10) ; ”*”
    40 NEXT X
    50 END
```

Acest program va afișa punctele unei curbe.

Instrucțiunea PRINT USING

Instrucțiunea PRINT USING este folosită pentru a determina diferite formate de afișaj sau tipărite a numerelor sau a șirurilor de caractere.

În cazul numerelor, instrucțiunea PRINT USING stabilește numărul de cifre posibile înaintea virgulei, poziția virgulei și numărul de zecimale. Instrucțiunea are formatul :

nr. linie PRINT USING "###...#.##...#" ; N

unde : — caracterele "#" (diez) dinaintea caracterului "." semnifică cifrele posibile dinaintea virgulei ;

— "." punctul zecimal (virgula) ;

— N reprezintă numărul.

Exemplu :

```
10 LET A = 7238.65
```

```
20 PRINT USING "#####.##" ; A
```

```
30 END
```

Execuția acestei secvențe va afișa : 7 238.6500

În cazul în care numărul pe care dorim să-l formatăm depășește partea întreagă a formatului, în fața numărului afișat va apare caracterul "%", fapt ce indică depășirea de capacitate.

Exemplu :

```
10 LET X = 2 375
```

```
20 PRINT USING "###.##" ; X
```

```
30 END
```

```
RUN
```

```
% 2 375.00
```

Pentru a constata diferența dintre instrucțiunile PRINT și PRINT USING, să considerăm următorul exemplu :

```
10 FOR N = 1 TO 5
```

```
20 READ X(N)
```

```
30 PRINT X(N) : NEXT N
```

```
40 PRINT ; PRINT
```

```
50 RESTORE
```

```
60 FOR N = 1 TO 5
```

```
70 READ X(N)
```

```
80 PRINT USING "###.##" ; NEXT N
```

```
90 DATA 1, 3.14, 565.3, 4 571.287, 13
```

```
100 END
```

Execuția programului va da :

```
1
```

```
3.14
```

```
565.3
```

```
4 571.287
```

```
13
```

```

1.00
3.14
565.30
% 4 571.29
13.00

```

Se observă că așezarea numerelor în urma executării instrucțiunii PRINT USING este alta decât la instrucțiunea PRINT. De asemenea, PRINT USING a realizat o rotunjire a părții fracționare a numărului 4 571.287 la 4 571.29.

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

Exemplu :

```

10 LET X = 15.73
20 PRINT USING "##.##"; X
30 PRINT USING "###.##"; X
RUN
15.73
15.73

```

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

Exemplu :

```

10 LET A = 32 157.35
20 PRINT USING "#####.###"; A
RUN
32,157.350

```

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

Exemplu :

```

10 LET X = 25
20 PRINT USING "#####"; X
30 PRINT USING "$$#####"; X
RUN
*****25
$ 25.00

```

Patru semne de exclamare așezate la sfârșitul formatului vor determina afișarea numărului în forma științifică, exprimat prin intermediul puterilor lui 10.

Exemplu :

```
10 LET X = 25
20 PRINT USING "##!!!" ; X
30 PRINT USING ".###!!!" ; X
40 PRINT USING "+.##!!!" ; X
RUN
2.5 E + 1
.250 E + 2
+.25 E + 2
```

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

Exemplu :

```
10 READ A$, A
20 PRINT USING "/ /"SÎNT DE"#### KG"; A$ ; A
30 IF A$ = "SF" THEN END
40 GOTO 10
50 DATA STOCURILE, 320, IEȘIRILE, 35
60 DATA COMENZILE, 178, SF
RUN
STOCURILE SÎNT DE 320 KG
IEȘIRILE SÎNT DE 35 KG
COMENZILE SÎNT DE 178 KG
```

Utilizarea instrucțiunii PRINT USING este deosebit de eficientă pentru editarea tabelelor, facturilor, comenzilor, formularelor sau a tuturor documentelor care necesită o prezentare standardizată.

Instrucțiuni complementare

AUTO. La unele interpretoare BASIC există facilități de numerotare automată a liniilor. În acest sens se folosește instrucțiunea AUTO, care are următorul format :

nr. linie AUTO N1, N

unde :

N1 este numărul liniei de la care dorim să facem numerotarea ;
N este incrementul.

Dacă incrementul N lipsește, atunci numerotarea se face automat din zece în zece. Dacă N1 lipsește, atunci numerotarea se realizează începînd cu linia 10.

SPACE \$ este o comandă care realizează spațierea cu un caracter ” ”.

Exemplu :

```
10 LET A$ = BUNĂ
20 LET B$ = ZIUA !
30 FOR X = 1 TO 5
40 PRINT A$ ; SPACE $ (X) ; B$
50 NEXT X
60 END
RUN
BUNĂ ZIUA !
BUNĂ ZIUA !
BUNĂ ZIUA !
BUNĂ ZIUA !
BUNĂ ZIUA !
```

PEEK este o instrucțiune care comandă citirea unei anume locații de memorie.

Formatul instrucțiunii este :

nr. linie PEEK (adresă)

unde adresă reprezintă numărul unei locații de memorie internă.

POKE servește la scrierea în memorie, la o adresă dată, a unei valori numerice.

Formatul instrucțiunii este :

nr. linie POKE (ADRESA, VALOARE)

Exemplu : 10 POKE (1 325,70). Executarea acestei instrucțiuni va conduce la depunerea valorii 70 în locația de memorie cu adresa 1 325. Același efect îl are și secvența :

```
10 LET A = 1 325
20 LET B = 70
30 POKE (A, B)
```

Instrucțiuni de prelucrare grafică

Soluțiile multor probleme constau din șiruri lungi de numere a căror interpretare este uneori dificilă.

Reprezentarea sub formă grafică a acestor șiruri de valori numerice facilitează aprecierea cantitativă și calitativă a soluțiilor.

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

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

Facilitățile grafice ale limbajului BASIC au o largă arie de utilizare ; știință și tehnologie, cartografie, proiectare asistată de calculator, construcție și fabricație asistată de calculator, instruirea asistată, simulare și animație (jocuri), conducere automată a proceselor tehnologice, publicistică, artă și comerț etc.

Una din soluțiile de realizare a graficelor în limbajul BASIC poate consta în scrierea unor subrutine în limbajul de asamblare, care utilizează un anumit display, și utilizarea lor în BASIC cu ajutorul instrucțiunii CALL. Această soluție este puțin flexibilă (nu este independentă de tipul perifericului grafic) și destul de greoaie. De aceea au fost introduse instrucțiuni speciale pentru prelucrări grafice.

MOVE. Această instrucțiune este folosită pentru poziționarea cursorului display-ului grafic în punctul de coordonate X, Y. De menționat este că instrucțiunea MOVE execută numai poziționarea în punctul de coordonate (X, Y) fără a marca punctul respectiv.

Formatul instrucțiunii este :

nr. linie MOVE X, Y

unde : — X, Y pot fi constante, variabile sau expresii.

PLOT este instrucțiunea care afișează punctul de coordonate X, Y.

Formatul instrucțiunii este :

nr. linie PLOT X, Y

unde X, Y sînt coordonatele punctului (pentru ecranul cu 24×32 caractere, $0 \leq X \leq 255$ și $0 \leq Y \leq 175$).

DRAW este utilizată pentru a trage o linie între punctul în care se află la întîlnirea instrucțiunii și punctul de coordonate X, Y specificate în instrucțiune.

Formatul instrucțiunii este :

nr. linie DRAW X, Y

Următorul exemplu realizează reprezentarea grafică a datelor conținute în vectorul V :

```
10 DIM V (20)
```

```
20 INPUT V (I)
```

```
30 MOVE 1, V (1)
```

```
40 FOR I = 1 TO 20
```

```
50 DRAW I, V (I)
```

```
50 NEXT I
```

```
70 END
```

Instrucțiunea 30 execută o poziționare în punctul de coordonate [1, V (1)], apoi în ciclul FOR se unesc prin linii elementele vectorului V. Se observă că instrucțiunile MOVE și DRAW sînt suficiente pentru a face reprezentări grafice.

Observație : Originea suprafeței grafice se consideră a fi punctul din stînga jos [de coordonate (0,0)].

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

Instrucțiunile **WINDOW** și **VIEWPORT**

Orice dispozitiv grafic se caracterizează printr-un număr de puncte distincte ce pot fi reprezentate, totalitatea acestor puncte alcătuind „spațiul grafic“.

Atributele principale ale spațiului grafic sînt : precizia (numărul de puncte din care este alcătuit spațiul grafic) și rezoluția (distanța dintre două puncte ale spațiului grafic). Instrucțiunile de prelucrare grafică reperă puncte ale spațiului grafic prin coordonatele lor (X, Y), exprimate în unități de măsură naturale, corespunzătoare mărimilor reprezentate grafic (metri, kg, volți etc.). Totalitatea acestor valori alcătuiesc „spațiul virtual” sau „spațiul utilizator”. Spațiul virtual este practic limitat doar de precizia aritmeticii calculatorului.

Spațiul virtual va putea fi reprezentat la o anumită scară, în cadrul spațiului grafic.

Instrucțiunea **WINDOW** permite utilizatorului să definească limitele spațiului virtual.

Formatul instrucțiunii este :

nr. linie **WINDOW** A, B, C, D

unde : — A, B, C, D sînt variabile, constante sau expresii care reprezintă limitele spațiului utilizator în ordinea :

A = limita stîngă

C = limita inferioară

B = limita dreaptă

D = limita superioară

Cele patru limite definesc un spațiu dreptunghiular. Orice punct de coordonate X, Y pentru care : $A \leq X \leq B$ și $C \leq Y \leq D$ va fi reprezentat grafic. Punctele care cad în afara dreptunghiului nu vor avea imagine pe suprafața display-ului.

Din linia trasă cu instrucțiunile :

10 **MOVE** X 1, Y 1

20 **DRAW** X 2, Y 2

va fi reprezentată numai porțiunea interioară dreptunghiului (ferestrei) definite mai sus de limitele A, B, C, D. Laturile ferestrei (dreptunghiului) declarate în instrucțiunea **WINDOW** trebuie să fie cît mai apropiate de domeniul de valori de reprezentat, pentru ca graficul obținut să fie cît mai fin.

Pînă acum s-a considerat că spațiul utilizator, definit prin instrucțiunea WINDOW va fi reprezentat pe toată suprafața display-ului. În unele aplicații se va dori scrierea unor comentarii alături de grafic sau realizarea mai multor grafice pe aceeași suprafață.

Pentru a descrie porțiunea din suprafața display-ului pe care va fi realizat graficul (va fi proiectat spațiul utilizator) se folosește instrucțiunea :

nr. linie VIEWPORT A, B, C, D

unde : — A, B, C, D sînt variabile, constante sau expresii și reprezintă limitele zonei din suprafața display-ului, în aceeași ordine ca pentru WINDOW. Spre deosebire de WINDOW, unitățile în care se exprimă liniile A, B, C, D sînt unități fizice. Alegerea unității fizice în care să se exprime limitele trebuie să satisfacă cerințele de independență față de tipul perifericului grafic (unele dispozitive grafice au suprafața pătrată, altele dreptunghiulară).

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

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

În aceste condiții instrucțiunea :

10 VIEWPORT 0, 100, 0, 100

va specifica suprafața celui mai mare pătrat înscris în suprafața grafică. Această instrucțiune se execută implicit la inițializarea sistemului.

Tot implicit se execută și instrucțiunea :

10 WINDOW 0, 100, 0, 100

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

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

Exemplu :

```
100 MOVE 85,50
```

```
110 FOR J = 0 TO 2 * PI STEP PI/10
```

```
120 DRAW 35 * cos (J) + 50, 35 * sin (J) + 50
```

```
130 NEXT J
```

```
140 END
```

Acest program va trasa un cerc pe orice display, cu centrul în punctul de coordonate (50, 50), iar raza de 35.

Dacă se va dori ca reprezentarea grafică să se facă numai pe o porțiune din suprafața display-ului, va trebui utilizată instrucțiunea VIEWPORT. De pildă, pentru colțul din dreapta sus se va folosi instrucțiunea :

```
10 VIEWPORT 75, 100, 75, 100
```

Instrucțiunea INIT are următorul format :

nr. linie INIT argument

Instrucțiunea INIT este utilizată pentru ștergerea ecranului display-ului grafic și trecerea lui în mod pagină (argument P) sau în mod defilare (argument S). INIT P trebuie să fie prima instrucțiune grafică dintr-un program. Această instrucțiune poziționează cursorul în punctul (0, 0), realizând inițializările grafice implicite, și anume :

WINDOW 0, 100, 0, 100 și

VIEWPORT 0, 100, 0, 100

Instrucțiunea SCALE. În unele aplicații, realizarea de hărți, diagrame, desene etc., este utilă indicarea explicită a scării de reprezentare dorite.

Formatul instrucțiunii folosite în acest caz este :

nr. linie SCALE S 1, S 2

unde : — S 1, S 2 sînt constante, variabile sau expresii care reprezintă factorii de scară pe orizontală și pe verticală. Factorul de scară indică numărul unităților utilizator reprezentate pe o unitate grafică :

$$Si = \frac{\text{unități utilizator}}{\text{unități grafice (UG)}}$$

În cazul în care se dorește efectuarea de măsurători pe grafic, va trebui să se știe cîți milimetri (de exemplu) reprezintă o unitate grafică (UG) pentru perifericul respectiv, știind că 1 UG este 1/100 din latura pătratului maxim inscriptibil în suprafața display-ului.

Originea sistemului de coordonate se consideră în punctul în care se află spotul (punctul grafic) la execuția instrucțiunii SCALE. Se poate remarca faptul că instrucțiunea WINDOW este echivalentă cu SCALE, avînd în plus specificarea originii. Coordonatele originii vor lua locul parametrilor A, C din instrucțiunea WINDOW, iar factorii de scară vor determina limitele B și D din WINDOW.

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

Exemplu :

10 INIT P

20 MOVE 10, 40

30 LABEL SCL (4) "PALATUL PIONIERILOR"

Această secvență de program va edita, începînd cu punctul (10, 40), textul "PALATUL PIONIERILOR", utilizînd scara de reprezentare 4.

Instrucțiunea ROTATE are ca efect numai asupra instrucțiunilor RMOVE și RDRAW, realizând rotația de un anumit unghi a vectorilor generați de RDRAW sau a poziționării realizate cu RMOVE.

Formatul instrucțiunii este :

nr. linie ROTATE U

unde : — U este o constantă, variabilă sau expresie. Dacă instrucțiunea ROTATE este utilizată înainte de apelul unei rutine care generează o figură cu ajutorul instrucțiunilor RMOVE și RDRAW, figura generată va apărea rotită cu unghi specificat (în radiani). Rotația figurii respective va fi realizată față de punctul în care s-a început generarea figurii, punctul unde se află spotul la începutul secvenței de program (rutinei) ce desenează figura.

Instrucțiunile MOVE și DRAW nu sînt afectate de ROTATE, deoarece coordonatele absolute, specificate în aceste instrucțiuni, trebuie să rămînă nealterate, pentru a se putea executa poziționări în punctele dorite înainte de generarea unor figuri cu RMOVE și RDRAW. Astfel, se permite utilizarea simultană și independentă a translației și rotației figurilor.

Exemplu :

```
10 MOVE 0, 0
20 RDRAW 10, 0
30 RDRAW 0, 10
40 RDRAW -10, 0
50 RDRAW 0, -10
60 END
```

Programul va trasa un pătrat cu latura de 10 unități, începînd din origine. Instrucțiunea 10 poziționează spotul în originea spațiului utilizator, care pentru inițializările implicite coincide cu originea suprafeței grafice.

Pătratul poate fi micșorat, mărit sau transformat în dreptunghi cu ajutorul instrucțiunii SCALE S 1, S 2. Dacă $S_1 = S_2 \neq 1$, pătratul va fi micșorat sau mărit, dar laturile vor rămîne egale între ele.

Pentru $S_1 = S_2 = 1$ pătratul va fi trasat neschimbat față de cel din exemplul de mai sus. În cazul în care $S_1 \neq S_2$, laturile trasate pe display nu vor mai fi egale, iar pătratul va fi reprezentat ca dreptunghi.

Următorul program va desena pătratul de două ori mai mare decît cel din exemplul precedent :

```
10 MOVE 0, 0
20 SCALE 1/2, 1/2
30 GOSUB 100
40 END
100 RDRAW 10, 0
110 RDRAW 0, 10
120 RDRAW -10, 0
130 RDRAW 0, -10
140 RETURN
```


Se observă că secvența de program care desena pătratul a fost scrisă ca subrutină. Dacă se dorește ca pătratul să fie desenat în alt loc pe suprafața display-ului, se va schimba instrucțiunea 10. De exemplu, punind :

```
10 MOVE 50, 50
```

pătratul va fi trasat (în sens trigonometric) începînd cu punctul de coordonate (50, 50).

Pătratul generat de subrutina 100 poate fi trasat începînd cu punctul de coordonate (50, 50) și rotit cu unghi $\text{PI}/4$ cu următorul program :

```
10 MOVE 50, 50
20 ROTATE PI/4 (45°)
30 GOSUB 100
40 END
```

În cazul în care se dorește ca pătratul să fie micșorat și rotit, în sens invers trigonometric, se va putea folosi programul :

```
10 MOVE 20, 30
20 SCALE 2, 2
30 ROTATE — PI/3
40 GOSUB 100
50 END
```

În acest exemplu, pătratul va fi trasat începînd cu punctul de coordonate (20, 30), micșorat la jumătate (latura de 5 UG) și rotit în sens orar cu unghi $\text{PI}/3$.

Instrucțiunea *GSINPUT* este utilizată pentru a realiza independența totală față de tipul perifericului grafic.

Formatul instrucțiunii este :

```
nr. linie GSINPUT V1, V2
```

unde : — V1, V2 sînt variabile. Execuția constă în atribuirea variabilelor V1, V2 a dimensiunilor exprimate în unități grafice (UG) ale suprafeței display-ului.

Utilizînd variabilele V1, V2 în instrucțiunea *VIEWPORT*, se vor obține programe care utilizează întreaga suprafață a display-ului, indiferent de forma sa.

Exemplu :

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

Instrucțiunea *AXIS* este utilizată pentru trasarea axelor pe suprafața grafică. În cazul în care pe nici una din axe mărimile nu iau valori de ambele semne, axele vor fi trasate astfel încît să se intersecteze în colțul din stînga jos. Instrucțiunea are doi parametri opționali, care reprezintă unitățile de măsură pe axele OX, respectiv OY, și, în cazul

cînd apar în instrucțiune, axele vor fi marcate. Pentru ca valorile de pe grafic să fie notate, trebuie folosită instrucțiunea LABEL.

Instrucțiunea CIRCLE servește la unele interpretoare BASIC pentru trasarea cercurilor.

Formatul instrucțiunii este :

nr. linie CIRCLE X, Y, R

unde : — X, Y sînt coordonatele centrului cercului iar R este raza cercului.

Instrucțiuni de calcul cu matrice

Cu toate că instrucțiunile deja prezentate permit efectuarea de prelucrări asupra tabolurilor, prin utilizarea variabilelor indexate (cu unul sau doi indici), limbajul BASIC conține un set de instrucțiuni care permit prelucrarea tablourilor (cu una sau două dimensiuni), fără referiri la fiecare element al tabloului.

MAT READ citește datele din instrucțiunile DATA pentru tablouri cu una sau două dimensiuni, dimensionate în prealabil prin DIM sau dimensionate chiar în instrucțiune.

Formatul instrucțiunii este :

nr. linie MAT READ A, B, C...

dacă A, B, C, ... au fost dimensionate prin DIM sau

nr. linie MAT READ A (n 1, m 1), B (n 2, m 2)...

cînd se dimensionează în instrucțiune.

Exemplu :

```
10 DIM B (3, 3), A (3)
20 MAT READ B (2, 3), A (3)
30 DATA 5, 11, -17, 1, 2, 3, 1E7, 0, 1
40 END
```

După execuția acestui program, matricele A și B vor conține :

A (1) = 10 ⁷	A (2) = 0	A (3) = 1
B (1, 1) = 5	B (1, 2) = 11	B (1, 3) = -17
B (2,1) = 1	B (2, 2) = 2	B (2, 3) = 3

În exemplul de mai sus instrucțiunea DIM nu era absolut necesară. Matricele pot fi alocate în momentul întîlnirii instrucțiunii MAT READ.

Instrucțiunea MAT INPUT este o instrucțiune de intrare pentru calculul cu matrice, datele fiind citite de la consolă.

Formatul instrucțiunii este :

nr. linie MAT INPUT A, B, C... sau

nr. linie MAT INPUT A (n 1, m 1), B (n 2, m 2)...

Exemplu :

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

Matricele A, B vor fi alocate în execuția instrucțiunii DIM. Instrucțiunea MAT INPUT redimensionează matricele A și B și apoi citește de la consolă valorile elementelor (pe linii).

Observație : 1. În cazul în care matricele au fost declarate în prealabil într-o instrucțiune DIM sau au fost introduse printr-o instrucțiune MAT anterioară, ele pot fi redimensionate cu respectarea condițiilor :

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

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

2. Dimensiunile specificate în instrucțiunile matriceale pot fi constate, variabile sau expresii.

MAT PRINT folosește pentru tipărirea matricilor. Matricele se tipăresc linie cu linie. În cadrul unei linii spațierea între elemente se face conform separatorului utilizat în lista de matrici din MAT PRINT (","; ";";").

Formatul instrucțiunii :

nr. linie MAT PRINT A, B, C.

Exemplu :

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

La execuția instrucțiunii 30 se vor tipări următoarele :

1	2	} matricea A
3	4	
5	6	
7	} vectorul B	
8		
9		

Vectorul B (3) va fi tipărit ca vector coloană.

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

Exemplu :

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

Execuția programului va afișa :

```
1      5 } A (2, 2)
3      2 }
7      } B (2)
8      }
```

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

$A (3, 3)$ (redimensionată) va fi :

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

Calculul inversei unei matrice

O matrice pătrată, nesingulară, poate fi inversată.

Formatul instrucțiunii :

nr. linie MAT A = INV (B)

De remarcat că o matrice poate fi inversată în ea însăși, adică matricea din membrul stâng al instrucțiunii poate fi aceeași cu cea din membrul drept.

Deci MAT A = INV (A) este corectă.

Dacă se dorește și calculul determinantului matricei B, se va folosi instrucțiunea : nr. linie MAT A = INV (B), V1.

La execuția instrucțiunii, variabilei V1 i se va atribui valoarea determinantului matricei B.

Exemplu :

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

Execuția acestui program va determina tipărirea matricelor.

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

$$C = A * B * A * A^{-1} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Se va tipări și valoarea determinantului matricii A

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

Transpusa unei matrice

Transpusa matricii A (m, n) va avea dimensiunile (n, m).

Formatul instrucțiunii este :

nr. linie MAT A = TRN (B)

Observații : a) O matrice nu poate fi transpusă în ea însăși (scrierea MAT A = TRN (A) este incorectă).

b) Transpusa unei matrice linie este o matrice coloană.

Produsul matricelor

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

Să considerăm instrucțiunea :

50 MAT A = B * C.

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

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

Exemplu :

10 DIM B (2, 3), C (3, 4), A (2, 4).

20 MAT READ B, C.

30 MAT A = B * C.

40 MAT PRINT A.

50 DATA 15, 4, 3, 2, 7, 10.

60 DATA 8, 7, 5, 6, 9, 10, 20, 15, 12, 25, 6, 1.

70 END.

RUN.

144 141 115.

36 39 35.

116 139 135.

= A.

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

10 MAT B = B * C
sau

10 MAT C = B * C

sînt incorecte.

Produsul unei matrice cu un scalar

Prin înmulțirea unei matrice cu un scalar se obține o matrice rezultat care are dimensiunile matricei date, iar elementele sînt înmulțite cu acel scalar.

Formatul instrucțiunii :

nr. linie MAT A = (expresie) * B.

Exemplu :

50 MAT A = (cos (X) + sin (X)) * B.

Adunarea și scăderea matricelor

Pot fi adunate sau scăzute numai matricele care au aceleași dimensiuni. Aceleași dimensiuni vor fi atribuite și matricei rezultat.

Formatul instrucțiunii este :

nr. linie MAT A = B + C.

Observație : a) Într-o instrucțiune nu se poate efectua decît o singură operație.

De exemplu, instrucțiunea :

10 MAT A = B + C - D

este incorectă.

b) Matricea din membrul stîng al atribuirii poate figura și în membrul drept.

20 MAT A = A + B.

Inițializarea unei matrice

a) Generarea unei matrice cu toate elementele nule :

nr. linie MAT A = ZER (m, n).

b) Generarea unei matrice cu toate elementele 1 :

nr. linie MAT A = CON (m, n).

c) Generarea unei matrice unitate :

nr. linie MAT A = IDN (N, N)

unde : N = min (m, n).

Muzică și culoare

Unele interpretoare BASIC, cum sînt cele folosite la calculatoarele românești HC-85, TIM-S, a MIC, PRAE au facilități muzicale sau de lucru în culori. În cele ce urmează mă voi referi la instrucțiunile specifice calculatorului HC-85 (compatibil SINCLAIR SPECTRUM) care generează sunete muzicale și culori.

Instrucțiunea BEEP este utilizată pentru producerea sunetelor muzicale.

Formatul instrucțiunii este :

nr. linie BEEP d, i

unde : d este o constantă, variabilă sau expresie numerică ce indică durata în secunde a sunetului respectiv ;

i este o constantă, variabilă sau expresie a căror evaluare reprezintă înălțimea sunetului, măsurat în semitonuri, respectiv la DO central.

Pentru a transcrie muzica este indicat să se scrie pe marginea fiecărui spațiu și linie a portativului înălțimea corespunzătoare, ținînd cont de armura cheii.

Exemplu :

10 PRINT "FRERE GUSTAV".

20 BEEP 1,0 : BEEP 1,2 : BEEP 5,3 : BEEP 5,2 : BEEP 1,0.

30 BEEP 1,0 : BEEP 1,2 : BEEP 5,3 : BEEP 5,2 : BEEP 1,0.

40 BEEP 1,3 : BEEP 1,5 : BEEP 2,7.

50 BEEP 1,3 : BEEP 1,5 : BEEP 2,7.

60 BEEP 75,7 : BEEP 25,8 : BEEP 5,7 : BEEP 5,5 : BEEP 5,3 :
BEEP 5,2 : BEEP 1,0.

70 BEEP 75,7 : BEEP 25,8 : BEEP 5,7 : BEEP 5,5 : BEEP 5,3 :
BEEP 5,2 : BEEP 1,0.

80 BEEP 1,0 : BEEP 1,—5 : BEEP 2,0.

90 BEEP 1,0 : BEEP 1,—5 : BEEP 2,0.

Pentru alcătuirea programului s-a procedat după cum urmează :

— s-au adăugat mai întîi deasupra și dedesubt cîte o linie de referință ;

— s-au numerotat liniile și spațiile, observînd că mi bemol din armura cheii afectează nu numai mi de sus, cît și mi de jos.

Pentru a schimba cheia partiturii, trebuie să se aducă la înălțimea fiecărei note o variabilă (de exemplu „cheie“) căreia trebuie să i se atribuie valoarea adecvată înaintea execuției piesei.

Astfel, linia 20 a programului devine :

20 BEEP 1, Cheia : BEEP 1,0...

În acest exemplu variabila „Cheie“ trebuie să aibă valoarea 0 pentru DO minor, 2 pentru RE minor, 12 pentru DO minor în octava superioară etc.

Cu acest sistem este posibilă acordarea calculatorului cu un alt instrument muzical, folosind valori zecimale pentru variabile „Cheie“.

În exemplul dat, „o pătrime“ a fost programată să dureze o secundă. Dacă se introduce o variabilă „PĂTRIME“ analog cu „Cheie“, linia 20 devine :

```
20 BEEP pătrime, cheie + 0 : BEEP pătrime, cheie + 2 : BEEP  
pătrime/2, cheie + 3 : BEEP pătrime/2, cheie + 2 : BEEP pătrime,  
cheie + 0.
```

În acest fel este posibilă execuția aceluiași program în orice cheie, cu orice acordare.

Programul : FOR n = 0 TO 1000 : BEEP · 5, n : NEXT n va produce note din ce în ce mai acute, până la limita posibilităților calculatorului.

Culori : Dacă un calculator are facilități color, atunci lista culorilor, în ordinea tastelor numerice, este :

- 0 — negru ;
- 1 — albastru (cian) ;
- 2 — roșu ;
- 3 — purpuriu (magenta) ;
- 4 — verde ;
- 5 — albastru deschis ;
- 6 — galben ;
- 7 — alb.

Într-un televizor alb-negru aceste numere corespund unor tonuri de gri ordonate de la închis spre deschis.

Orice caracter are asociate 2 culori (caracter și fond). La pornirea calculatorului, sistemul lucrează în alb-negru, cu caractere negre pe fond alb. Tipărirea poate fi făcută normal, dar există și posibilitatea să apară pe ecran pîlpîind (flash), fenomene care să existe inversînd conținutul culoarea caracterului cu culoarea fondului. Deoarece atributele de culoare și pîlpîire sînt asociate caracterelor, nu este posibil ca într-un caracter să fie mai mult de două culori. Valorile acestor atribute pot fi modificate cu instrucțiunile INK, PAPER și FLASH.

Formatele acestor instrucțiuni sînt :

- nr. linie PAPER n ;
- nr. linie INK n ;
- nr. linie FLASH m.

unde : n este un număr cuprins între 0 și 7 ;

m este un număr binar (0 pentru inactiv și 1 pentru activ).

Exemplu :

```
20 FOR N = 1 TO 11  
30 FOR C = 0 TO 7  
40 PAPER C : PRINT " " ; : REM spații colorate  
50 NEXT C : NEXT n  
60 PAPER 7  
70 FOR C = 0 TO 3  
80 INK C : PRINT C ; " "
```

```

90 NEXT C : PAPER 0
100 FOR C = 0 TO 7
110 INK C : PRINT C ; " "
120 NEXT C
130 PAPER 7 : INK 0

```

În afară de aceste 8 culori (0,7), mai pot fi folosite valorile 8 și 9. Tasta 8 poate fi folosită ca argument pentru toate cele 4 comenzi și specifică transparența, fapt ce nu alterează atributele poziției la tipărirea unui caracter. De exemplu: PAPER 8 face ca la tipărirea unui caracter culoarea fondului să fie aceeași cu a caracterului tipărit anterior. 9 poate fi folosit numai cu comenzile PAPER și INK și indică contrastul. Culoarea „cernelii” sau a „hîrtiei”, în funcție de comanda utilizată, este făcută să contrasteze cu cealaltă, punînd alb pe o culoare închisă (negru, albastru, roșu, magenta) și negru pe o culoare deschisă (verde, bleu, galben, alb).

Exemplu :

```
INK 9 : FOR C = 0 TO 7 : PAPER C : PRINT C : NEXT C.
```

Comanda *INVERSE 1* inversează fundalul cu cerneala pentru caracterul specificat.

Comanda *OVER 1* realizează supratipărirea.

Marginea display-ului poate lua oricare din cele 8 culori (0—7), cu comanda : *BORDER n*.

Comenzile INK, PAPER, FLASH pot apare în PRINT urmate de " ; ", efectul lor fiind temporar.

Exemplu :

PRINT PAPER 6 ; "X" : PRINT "Y" unde numai X va fi tipărit pe fond galben. De asemenea, se pot schimba culorile mesajului scris pe ecran cu comanda INPUT, inserînd în aceasta comanda INK, PAPER etc., ca și în cazul comenzii PRINT. Efectul lor este activ numai asupra comenzii următoare :

Exemplu :

```
INPUT FLASH 1 ; INK 1 ; "TEXT" ; n.
```

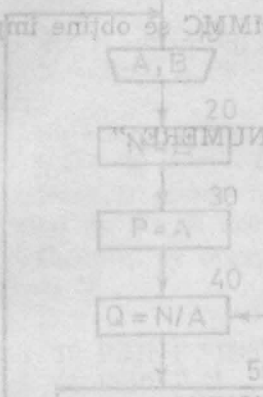
4.1. APLICAȚII ÎN MATEMATICĂ

DIVIZIBILITATE ȘI NUMERE PRIME

4.1.1. Calculul celui mai mare divizor comun și al celui mai mic multiplu comun (CMMDC și CMMMC)

Fie A și B două numere întregi nenule, cu $A < B$. Pentru a calcula CMMDC și CMMMC se poate utiliza algoritmul lui Euclid :

— se împarte B prin A ; citul este Q și restul R ;



APLICAȚII

Utilizînd calculatoarele personale, se pot realiza programe în limbaj BASIC, care să rezolve diverse calcule matematice, probleme de geometrie, fizică, chimie etc., agende personale, desene, melodii, jocuri și multe altele.

În general, în programele date spre exemplificare s-a căutat să se respecte limbajul BASIC-80. Pentru a putea fi utilizate și pe celelalte calculatoare (HC-85, TIM-S) sînt necesare adaptări minime, care sînt specificate la primele programe date spre exemplificare. Pentru programele care s-au realizat pe un anumit tip de calculator și care necesită modificări numeroase pentru a putea rula și pe celelalte tipuri de calculatoare, se va preciza tipul de calculator pe care sînt funcționale, rămîinînd în sarcina utilizatorului de a le adapta pe tipul de calculator care îl are la dispoziție.

Programele pot fi începute cu o instrucțiune de ștergere a ecranului : CLS.

Listingurile programelor sînt realizate pe o imprimantă de tip SCAMP (132 caractere pe rînd). De aceea, rîndul de listing nu va coincide cu rîndul de pe ecranul calculatorului (care poate avea 32 de caractere, de exemplu : la aMIC, PRAE, HC-85 și TIM-S). Pentru unele programe la care se precizează tipul de calculator pe care sînt funcționale, listingurile s-au realizat prin copierea imaginilor ecran, care conțineau programul listat la imprimantă. Astfel, unele listinguri de programe vor prezenta 32 caractere pe rînd, iar linia de instrucțiune va fi continuată pe rîndul următor în același mod în care se realizează afișarea pe ecran.

4.1. APLICAȚII ÎN MATEMATICĂ DIVIZIBILITATE ȘI NUMERE PRIME

4.1.1. Calculul celui mai mare divizor comun și al celui mai mic multiplu comun (CMMDC și CMMMC)

Fie A și B două numere întregi nenule, cu $A \leq B$. Pentru a calcula CMMDC și CMMMC se poate utiliza algoritmul lui Euclid :
— se împarte B prin A ; citul este Q și restul R ;

— dacă $R \neq 0$ se înlocuiește B prin A și A prin R și se continuă
procedul ;

— CMMDC este ultimul rest nenul, iar CMMMC se obține împărțind $A \times B$ prin CMMDC.

Programul este :

5 PRINT "INTRODUCETI CELE DOUA NUMERE :"

10 INPUT A , B

20 LET N = B

30 LET P = A

40 LET Q = N/P

50 LET R = N - P * INT (Q)

60 IF R = 0 THEN GO TO 100

70 LET N = P

80 LET P = R

90 GO TO 40

100 PRINT "CMMDC = " ; P

110 PRINT "CMMMC = " ; A * B / P

115 PRINT

120 GO TO 5

INTRODUCETI CELE DOUA NUMERE :

72

42

CMMDC = 6

CMMMC = 504

INTRODUCETI CELE DOUA NUMERE :

6

8

CMMDC = 3

CMMMC = 6

Observații :

— la calculatoarele aMIC, PRAE și TPD JUNIOR, instrucțiunile

20, 30, 40, 50, 70 și 80 pot fi scrise fără LET ;

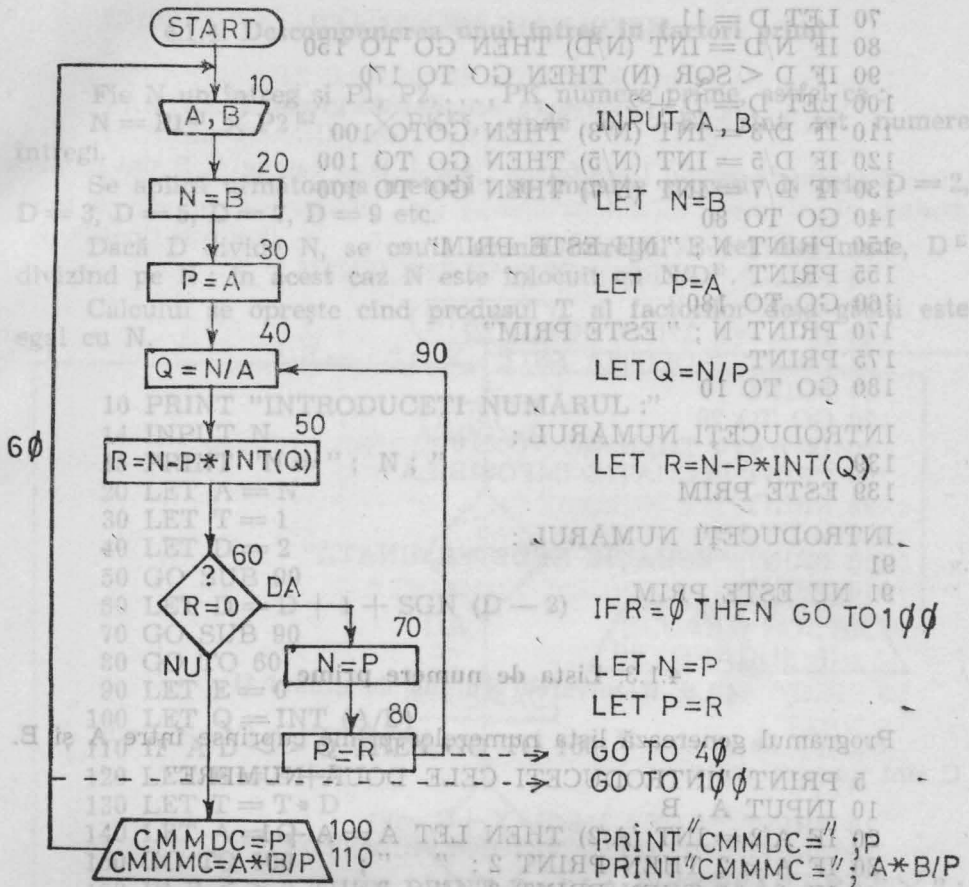
— la calculatoarele HC/85 și TIM-S, la instrucțiunea 50 se poate
scrie INT Q ;

— instrucțiunea 115 PRINT are rolul de a face să se lase un
rînd gol ;

— variabilele N și P se introduc pentru a se reține neschimbate
valorile lui A și B, necesare în calculul CMMMC (instrucțiunea 110) ;

— la calculatoarele PRAE, HC/85, TIM-S și TPD JUNIOR instruc-
țiunile 5 și 10 se pot scrie împreună sub forma : 10 INPUT "INTRO-
DUCETI CELE DOUA NUMERE" ; A, B.

— se împarte B prin A ;



4.1.2. Verificarea dacă un întreg este prim

Programul se bazează pe următoarele ipoteze:

— dacă D nu divide pe N, atunci nici un multiplu al său nu va divide pe N;

— este de ajuns să se încerce divizorii inferiori lui \sqrt{N} (deoarece dacă N are cel puțin doi divizori, unul din ei este în mod necesar inferior sau egal cu \sqrt{N}).

Programul funcționează pentru un N impar și superior lui 12.

```

5 PRINT "INTRODUCETI NUMARUL : "
10 INPUT N
20 IF N <= 12 THEN GO TO 10
30 IF N/2 = INT (N/2) THEN GO TO 10
40 IF N/3 = INT (N/3) THEN GO TO 150
50 IF N/5 = INT (N/5) THEN GO TO 150
60 IF N/7 = INT (N/7) THEN GO TO 150
  
```

```

70 LET D = 11
80 IF N/D = INT (N/D) THEN GO TO 150
90 IF D < SQR (N) THEN GO TO 170
100 LET D = D + 2
110 IF D/3 = INT (N/3) THEN GOTO 100
120 IF D/5 = INT (N/5) THEN GO TO 100
130 IF D/7 = INT (N/7) THEN GO TO 100
140 GO TO 80
150 PRINT N ; " NU ESTE PRIM"
155 PRINT
160 GO TO 180
170 PRINT N ; " ESTE PRIM"
175 PRINT
180 GO TO 10

```

INTRODUCETI NUMARUL :

```

139
139 ESTE PRIM

```

INTRODUCETI NUMARUL :

```

91
91 NU ESTE PRIM

```

4.1.3. Lista de numere prime

Programul generează lista numerelor prime cuprinse între A și B.

```

5 PRINT "INTRODUCETI CELE DOUA NUMERE"
10 INPUT A, B
20 IF A/2 = INT (A/2) THEN LET A = A + 1
30 IF A <= 2 THEN PRINT 2 ; " " ;
40 IF A <= 3 THEN PRINT 3 ; " " ;
50 FOR N = A TO B STEP 2
60 LET D = 3
70 IF N/D = INT (N/D) THEN GO TO 110
80 IF D > SQR (N) THEN GO TO 110
90 LET D = D + 2
100 GO TO 70
110 IF N >= 7 THEN PRINT N ; " " ;
120 NEXT N
125 PRINT
130 GO TO 10

```

INTRODUCETI CELE DOUA NUMERE :

```

6
2
7 11 13 17 19

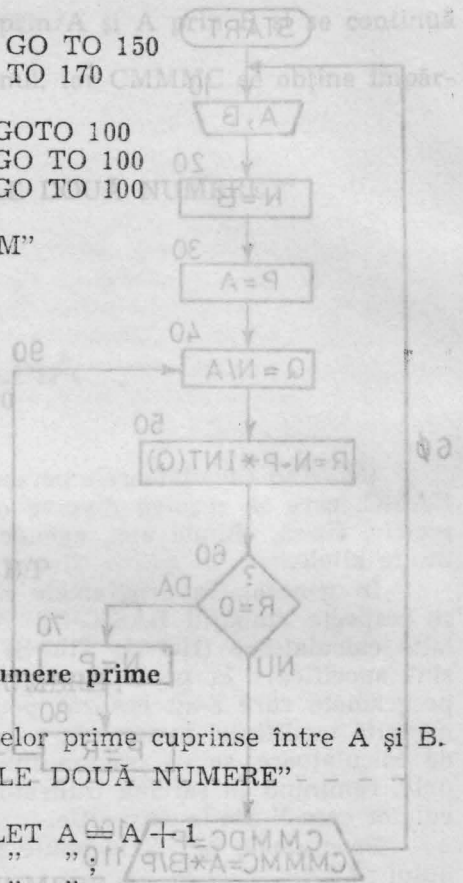
```

INTRODUCETI CELE DOUA NUMERE :

```

1
27
2 3 5 7 11 13 17 19 23

```



4.1.4. Descompunerea unui întreg în factori primi

Fie N un întreg și P_1, P_2, \dots, P_K numere prime, astfel ca :

$N = P_1^{E_1} \times P_2^{E_2} \dots \times P_K^{E_K}$, unde $E_1 \dots E_k$ sînt tot numere întregi.

Se aplică următoarea metodă : se împarte succesiv N prin $D = 2, D = 3, D = 5, D = 7, D = 9$ etc.

Dacă D divide N , se caută atunci întregul E cel mai mare, D^E divizînd pe N ; în acest caz N este înlocuit cu N/D^E .

Calculul se oprește cînd produsul T al factorilor deja găsiți este egal cu N .

```
10 PRINT "INTRODUCEȚI NUMĂRUL :"  
14 INPUT N  
15 PRINT "N = " ; N ; "  
20 LET A = N  
30 LET T = 1  
40 LET D = 2  
50 GO SUB 90  
60 LET D = D + 1 + SGN (D - 2)  
70 GO SUB 90  
80 GO TO 60  
90 LET E = 0  
100 LET Q = INT (A/D)  
110 IF A/D <> Q THEN GO TO 160  
120 LET E = E + 1  
130 LET T = T * D  
140 LET A = Q  
150 GO TO 100  
160 IF E <> 0 THEN PRINT D ; " LA PUTEREA " ; E ; " × " ;  
170 IF T ≥ N THEN GO TO 190  
180 RETURN  
185 PRINT  
190 GO TO 10
```

INTRODUCEȚI NUMĂRUL :

```
90  
90 = 2 × 3 LA PUTEREA 2 × 5  
90 = 2 × 3 LA PUTEREA 2 × 5  
336  
336 = 2 LA PUTEREA 4 × 3 × 7
```

Observații :

— subrutina de la instrucțiunea 90 pînă la instrucțiunea 180 realizează identificarea numerelor prime ;

— instrucțiunea 60 are ca efect atribuirea valorilor 3, 5, 7... adică a valorilor numerelor prime lui D ($\text{SGN}(D - 2)$ este 0 sau 1).

REZOLVĂRI DE ECUAȚII

4.1.5. Rezolvarea ecuației de gradul I

Ecuția de gradul I are forma : $AX + B = 0$ unde A, B sînt numere reale :

```
10 PRINT "AX + B = 0"
20 PRINT "INTRODUCEȚI COEFICIENTII : "
30 INPUT A, B
40 IF A = 0 THEN GO TO 80
50 PRINT "SOLUȚIA ESTE : X = " ; -B/A
60 PRINT
70 GO TO 20
80 IF B = 0 THEN GO TO 110
90 PRINT "ECUAȚIE IMPOSIBILĂ"
95 PRINT
100 GO TO 20
110 PRINT "ECUAȚIE NEDETERMINATĂ"
115 PRINT
120 GO TO 20
```

4.1.6. Rezolvarea ecuației de gradul II

Ecuția de gradul II are forma : $AX^2 + BX + C = 0$, unde A, B, C sînt numere reale.

```
10 PRINT "A * X ^ 2 + B * X + C = 0"
15 PRINT "INTRODUCEȚI COEFICIENTII : "
25 INPUT A, B, C
30 IF A < > 0 THEN GO TO 45
35 PRINT "ECUAȚIA NU ESTE DE GRADUL 2"
40 GO TO 105
45 LET M = B * B - 4 * A * C
50 IF M >= 0 THEN GO TO 80
55 PRINT "ECUAȚIA ARE RĂDĂCINI COMPLEXE"
60 PRINT "PARTEA REALĂ" ; -B / (2 * A)
70 PRINT "PARTEA IMAGINARĂ" ; SQR (-M) / (2 * A)
72 GO TO 105
80 PRINT "RĂDĂCINI REALE"
90 PRINT "X1 =" ; (-B + SQR (M)) / (2 * A)
100 PRINT "X2 =" ; (-B - SQR (M)) / (2 * A)
105 PRINT
110 GO TO 15
```

$A * X^2 + B * X + C = 0$

INTRODUCEȚI COEFICIENTII :

0, 23, 11

ECUAȚIA NU ESTE DE GRADUL 2

INTRODUCEȚI COEFICIENȚII :

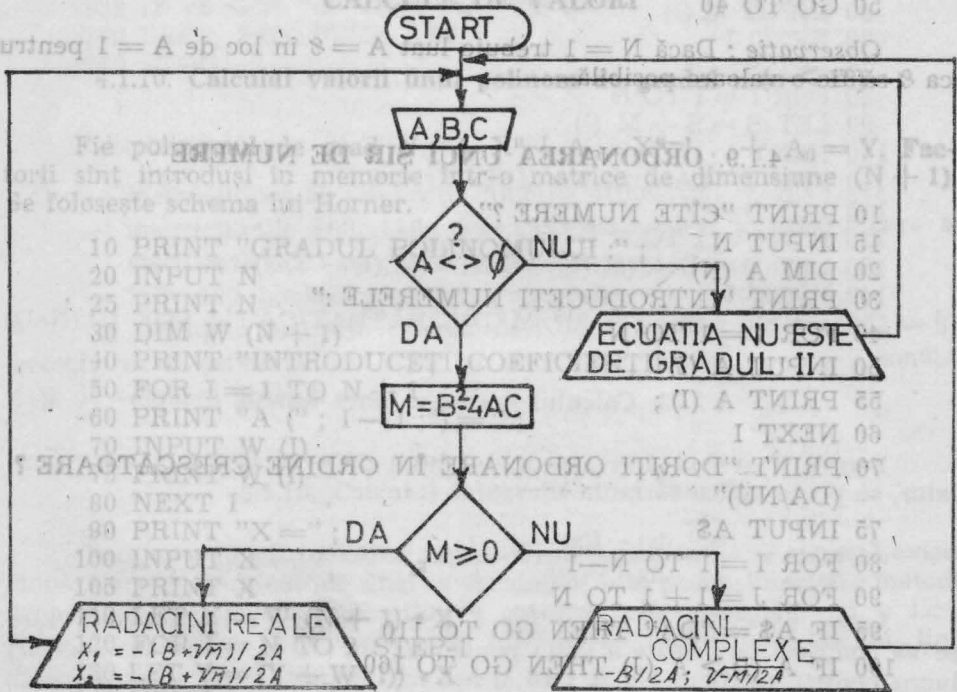
3, 4, 45.6

ECUAȚIA ARE RĂDĂCINI COMPLEXE

PARTEA REALĂ .666667

PARTEA IMAGINARĂ — 3.8413

Nu s-au luat în considerație variantele în care toți coeficienții sînt 0, caz în care ecuația este nedeterminată sau cazurile în care doi coeficienți sînt nuli.



GENERARE DE NUMERE ALEATOARE

4.1.7. Generarea de întregi aleatori într-un interval dat

Fie A și B două numere întregi care definesc intervalul.

Programul va genera la infinit (deci cu repetiție) numere întregi cuprinse în acest interval.

```
10 PRINT "INTRODUCEȚI MARGINILE INTERVALULUI :"  
15 INPUT A, B  
20 PRINT A + INT (1 + (B - A) * RND (1))  
30 GO TO 20
```

Observație : la HC-85 și TIM-S, funcția RND nu are argument.

4.1.8. Generarea de întregi aleatori avînd n cifre

Generarea de întregi aleatorii de n cifre este echivalentă cu generarea de întregi aleatori în intervalul $A = 10^{n-1}$; $B = 10^n$.

```
10 PRINT "INTRODUCEȚI NUMĂRUL DE CIFRE :"  
15 INPUT N  
20 LET A = 10 ↑ (N - 1)  
30 LET B = 10 ↑ N  
40 PRINT A + INT ((B - A) * RND (1))  
50 GO TO 40
```

Observație : Dacă $N = 1$ trebuie luat $A = 0$ în loc de $A = 1$ pentru ca 0 să fie o valoare posibilă.

4.1.9. ORDONAREA UNUI ȘIR DE NUMERE

```
10 PRINT "CÎTE NUMERE ?"  
15 INPUT N  
20 DIM A (N)  
30 PRINT "INTRODUCEȚI NUMERELE :"  
40 FOR I = 1 TO N  
50 INPUT A (I)  
55 PRINT A (I) ;  
60 NEXT I  
70 PRINT "DORIȚI ORDONARE ÎN ORDINE CRESCĂTOARE ?  
(DA/NU)"  
75 INPUT A$  
80 FOR I = 1 TO N - 1  
90 FOR J = I + 1 TO N  
95 IF A$ = "DA" THEN GO TO 110  
100 IF A (I) > A (J) THEN GO TO 160  
105 GO TO 115  
110 IF A (I) < A (J) THEN GO TO 160  
115 LET B = A (J)  
120 FOR K = J TO I + 1 STEP - 1  
130 LET A (K) = A (K - 1)  
140 NEXT K  
150 LET A (I) = B  
160 NEXT J  
170 NEXT I  
180 FOR I = 1 TO N  
190 PRINT A (I) ;  
200 NEXT I  
210 STOP  
220 END
```

CITE NUMERE ?

6

INTRODUCETI NUMERELE :

3 5 1 6 8 2
DORIȚI ORDONARE ÎN ORDINE CRESCĂTOARE ? (DA/NU)

DA

1 2 3 5 6 8

CALCULE DE VALORI

4.1.10. Calculul valorii unui polinom de gradul n ($n < 10$)

Fie polinomul de grad N : $A_n X^n + A_{n-1} X^{n-1} + \dots + A_0 = Y$. Factorii sînt introduși în memorie într-o matrice de dimensiune $(N + 1)$. Se folosește schema lui Horner.

```
10 PRINT "GRADUL POLINOMULUI .:" ;
```

```
20 INPUT N
```

```
25 PRINT N
```

```
30 DIM W (N + 1)
```

```
40 PRINT "INTRODUCETI COEFICIENTII .:"
```

```
50 FOR I = 1 TO N + 1
```

```
60 PRINT "A (" ; I - 1 ; ") = " ;
```

```
70 INPUT W (I)
```

```
75 PRINT W (I)
```

```
80 NEXT I
```

```
90 PRINT "X = " ;
```

```
100 INPUT X
```

```
105 PRINT X
```

```
110 LET Y = W (N + 1) * X
```

```
120 FOR I = N TO 2 STEP -1
```

```
130 LET Y = (Y + W (I)) * X
```

```
140 NEXT I
```

```
145 LET Y = Y + W (1)
```

```
150 PRINT "Y = " ; Y
```

```
160 STOP
```

```
170 END
```

4.1.11. Calculul valorii medii și a abaterii medii pătratice

Pentru a studia dispersia între mai multe valori răspindite în jurul unei valori medii, utilizăm următoarele formule:

media aritmetică : med. = $\frac{\text{suma valorilor}}{\text{numărul valorilor}}$;

abaterea medie pătratică : disp. = $\frac{\text{suma pătr. dif. dintre val. și medie}}{\text{numărul valorilor}}$

Dacă se dispune de o imprimantă, programul va furniza lista valorilor și rezultatele : media și abaterea medie pătratică.

```

10 REM CALCULUL VALORII MEDII ȘI ABATERII MEDII
    PĂTRATICE
20 PRINT "CÎTE MĂSURĂTORI AȚI FĂCUT ?"
30 INPUT N
35 DIM M (N)
40 PRINT "INTRODUCEȚI VALORILE MĂSURATE"
50 FOR I = 1 TO N
60 INPUT M (I)
70 NEXT I
75 LET S = 0
80 FOR I = 1 TO N
90 LET S = S + M (I)
100 NEXT I
110 PRINT "VALOAREA MEDIE :"; S/N
115 LET D = 0
120 FOR I = 1 TO N
130 LET D = D + (M (I) - S/N) * (M (I) - S/N)
140 NEXT I
150 PRINT "ABATEREA MEDIE PĂTRATICĂ :"; SQR (D/(N-1))

```

4.1.12. Calculul derivatei unei funcții

Pentru a evalua derivata unei funcții de o variabilă cu un calculator, se poate aplica formula :

$$\text{derivata} = \frac{f(x + dx) - f(x)}{dx} \text{ cu un } dx \text{ suficient de mic.}$$

Luînd $dx = 1/4$ se va obține o primă estimăție a derivatei, care se va compara cu cea de a doua pentru un $dx = 1/8$. Se va face același lucru pentru un $dx = 1/8$ și $1/16$ și așa mai departe pînă cînd diferența dintre 2 estimări succesive ale derivatei va fi inferioară unei valori impuse. Această valoare a fost fixată la 10^{-6} .

Programul se poate rula pe calculatoarele HC-85 și TIM-S fixîndu-se ca număr maxim de evaluări 19.

```

10 PRINT "Derivata unei funcții"
20 PRINT : PRINT "Expresia funcției"
30 BEEP, 2,20 : INPUT "Tastează expresia de variabilă x"; e$
40 PRINT PAPER 6 ; INK 9 ; AT 4,5 ; e$
50 DEF FN f (x) = VAL e$
60 LET eps = 1e - 6
100 PRINT : PRINT "valoarea lui x";
110 BEEP, 2,20 : INPUT "x"; x : PRINT x
120 PRINT "Derivata este estimată"
130 LET der 2 = 0
140 FOR i = 2 TO 20

```



```

150 LET der i = der 2
155 LET dx = .5 i
160 LET der 2 = (FN f (x + dx) - FN f (x))/dx
165 LET prec = ABS (der i - der 2)
170 IF prec <= eps THEN GO TO 200
180 NEXT i
200 LET derivata = 2 * der 2 - der i
210 PRINT : PRINT PAPER 6 ; INK 9 : derivata
230 BEEP. 2,20 : INPUT „Altă valoare a lui x (d/n)” : r$
240 IF r$ = "d" THEN GO TO 100
250 IF r$ <> "n" THEN GO TO 230
270 BEEP. 2,13 : BEEP. 2,16
280 STOP

```

Observații :

- instrucțiunea 50 : definiția funcției utilizator ;
- instrucțiunile 100, 110 : introducerea valorii x pentru care se calculează derivata ;
- 140—180 bucla de estimări succesive ale derivatei.

Ca exemplu se poate calcula derivata lui $\sin(x)$ pentru $x = 0$; aceasta va fi estimată la 1.0000203, deci un grad de precizie rezonabil față de valoarea reală (1).

4.1.13. Calculul integralei unei funcții

Pentru o funcție a cărei expresie este cunoscută, în general există două metode de calcul ale unei integrale definite pe un interval : metoda trapezelor (pentru care funcția de integrat este asimilată cu o linie frântă) și metoda Simpson (care înlocuiește segmentele acestei linii frânte prin arcuri de parabole).

Pentru metoda trapezelor, integrala va fi aproximată de suma suprafețelor trapezelor mici de înălțime li și cu bazele reprezentate de două valori succesive ale funcției.

Formula de calcul va fi :

$$\text{integrala} = [f(x_i)/2 + \sum_{i=1}^{n-1} [f(x_i + i \cdot li)] + f(x_f)/2] \cdot li$$

unde : n este numărul de subintervale.

Programul este realizat pentru calculatoarele HC-85 și TIM-S.

```

5 DEF FN i (t) = SIN t
10 PRINT "Integrala unei funcții"
20 PRINT "a cărei expresie este cunoscută"
30 PRINT "Metoda trapezelor"
40 PRINT : PRINT "intervalele [" ;

```

```

50 BEEP .2, 20 : INPUT "limita inferioară"; xi : PRINT xi;";
60 BEEP .2, 20 ; INPUT "limita superioară"; xf : PRINT xf "
70 PRINT "precizia integralei";
80 BEEP .2, 20 : INPUT "1 : mică, 1000 : mare"; ni
85 PRINT TAB 26-LEN STR$ ni ; ni ;" subintervale"
90 LET li = (xf-xi)/ni
100 LET integ = FN i(xi)/2
110 FOR i = 1 TO ni-1
120 LET integ = integ + FN i (xi + i * li)
130 NEXT i
140 LET integ = integ + FN i (xf)/2
150 LET integ = integ * li
170 PRINT : PRINT "valoarea integralei"; integ
190 BEEP .2, 20 : INPUT "altă precizie (d/n)"; r$
200 IF r$ = "d" THEN PRINT : GO TO 70
210 IF r$ = "n" THEN GO TO 240
220 GO TO 190
240 BEEP .2, 13 : BEEP .2, 16
250 STOP

```

Observații :

- instrucțiunea 5 : definiția funcției de integrat ;
- 90 mărimea unui subinterval ;
- 100 primul termen al sumei (punctul xi) ;
- 110—130 buclă de cumul pentru punctele intermediare ;
- 140 se adaugă ultimul termen (la punctul xf).

Pentru alte funcții se poate redefini linia 5.

De exemplu : pentru funcția t^2 linia 5 va fi :

```
5 DEF FN i (t) = t * t.
```

Valoarea este obținută prin exces (valoarea exactă este 9) și se ameliorează dacă numărul de subintervale crește.

Rezultatul unei rulări :

Integrala unei funcții a cărei expresie este cunoscută

Metoda trapezelor
Intervalul [0, 3]

Precizia calculului 10 subintervale

Valoarea integralei 9.045

Precizia calculului 50 subintervale

Valoarea integralei 9.0018

CALCULE GEOMETRICE

4.1.14. Calculul perimetrului și suprafeței unui triunghi

Se calculează perimetrul și suprafața unui triunghi cunoscându-se lungimile laturilor.

```
10 PRINT "INTRODUCETI LUNGIMILE LATURILOR :"  
20 INPUT X, Y, Z  
30 LET P = X + Y + Z  
40 LET Q = P/2  
50 LET S = SQR (Q * (Q - X) * (Q - Y) * (Q - Z))  
60 PRINT "PERIMETRUL ." ; P  
70 PRINT "SUPRAFAȚA ." ; S  
80 GO TO 10
```

REPREZENTĂRI DE FUNCȚII

4.1.15. Curbă digitală

Pentru reprezentarea cu ajutorul unei curbe a mai multor valori observate (de exemplu : talia unui copil în funcție de vârsta lui) putem utiliza un program ca cel de mai jos pentru HC-85 și TIM-S cu care prin simpla introducere a coordonatelor punctelor, acestea vor apărea pe ecran în locul respectiv.

```
5 BORDER 6 : PAPER 7 : INK 9  
10 PRINT "trasarea unei curbe digitale"  
15 LET d = .2  
20 BEEP d, 13 : BEEP d, 16  
30 INPUT "numărul de puncte" ; np  
40 DIM x (np) : DIM y (np)  
55 PRINT AT 2, 2 ; "punct" ; TAB 15 ; "x" ; TAB 25 ; "y"  
60 FOR i = 1 TO np  
70 PRINT TAB 5 ; i  
75 BEEP d, 20  
80 INPUT "x" ; x (i)  
82 PRINT TAB 12 ; x (i)  
85 BEEP d, 20  
90 INPUT "y" ; y (i)  
95 PRINT TAB 22 ; y (i)  
100 NEXT i  
110 PRINT : PRINT "căutarea extremelor"  
120 GO SUB 1000
```

```

135 CLS
140 LET xe = 255/(xmax-xmin)
150 LET ye = 175/(ymax-ymin)
160 FOR i = 1 TO np
170 PLOT (x(i)-xmin) * xe, (y(i)-ymin) * ye
180 NEXT i
200 BEEP d, 13 : BEEP d, 16
210 STOP
980 REM
1000 LET xmin = x (1)
1010 LET xmax = x (1)
1020 LET ymin = y (1)
1030 LET ymax = y (1)
1050 FOR i = 2 TO np
1050 IF x (i) < xmin THEN LET xmin = x (i)
1060 IF x (i) > xmax THEN LET xmax = x (i)
1070 IF y (i) < ymin THEN LET ymin = y (i)
1080 IF y (i) > ymax THEN LET ymax = y (i)
1090 NEXT i
1100 IF xmin = xmax OR ymin = ymax THEN BEEP d, 10 :
      BEEP d, 5 : PRINT "verifică coordonatele" : STOP
1110 RETURN

```

Observații :

- 60— 100 introducerea coordonatelor ;
- 140— 180 trasarea punctelor ;
- 1000—1030 inițializarea înainte de căutarea extremelor ;
- 1040—1090 buclă de căutare.

De exemplu : de fiecare dată când se găsește o valoare $x(i)$ mai mică decât cea mai mică valoare dintre precedentele, trebuie schimbat conținutul lui $xmin$.

1100 utilizare eficientă a ecranului, curba nu se poate reduce la o dreaptă orizontală sau verticală.

Exemplu de utilizare :

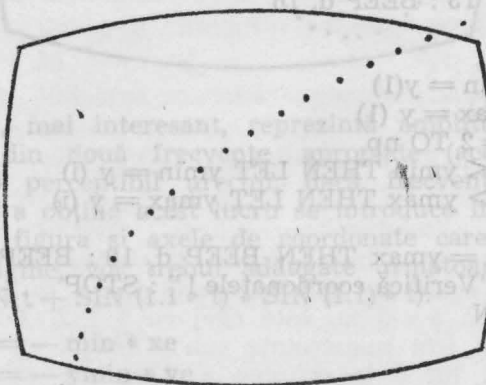
Reprezentarea grafică a taliei unui individ în funcție de vârsta lui.

Vârsta în ani	(x)	0	1	2	3	4	5	6	7	8	
Talia în cm	(y)	50	72	85	92	100	106	114	118	124	
Vârsta în ani	(x)	9	10	11	12	14	16	18	20	25	
Talia în cm	(y)	130	135	145	160	164	165	166	167	167	
Număr de puncte :		18									

Trasarea unei curbe digitale

punct	x	y
1	0	50
2	1	72
3	2	85
4	3	92
5	4	100
6	5	106
7	6	114
8	7	118
9	8	124
10	9	130
11	10	135
12	11	145
13	12	160
14	14	164
15	16	165
16	18	166
17	20	167
18	25	167

Căutarea extremelor



4.1.16. Reprezentarea unei funcții de o variabilă

Cîteodată reprezentarea evoluției unei funcții este imaginată cu dificultate atunci cînd argumentul variază. În acest caz, o imagine poate fi foarte eficientă. Valorile funcției se vor reprezenta pe verticală, iar cele ale argumentului funcției pe orizontală.

Programul funcționează pentru calculatoarele HC-85 și TIM-S.


```

5 BORDER 6 : PAPER 7 : INK 9
10 PRINT "trasarea funcției de o variabilă"
15 DEF FN y (t) = SIN t
20 LET d = .2
30 BEEP d, 20
40 INPUT "Valoarea minimă" ; xmin
50 BEEP d, 20
60 INPUT "Valoarea maximă" ; xmax
70 BEEP d, 20
80 INPUT "Numărul de puncte" ; np
90 DIM y(np)
110 PRINT : PRINT "Calculul ordonatelor"
120 LET x = xmin
130 LET xpas = (xmax-xmin)/(np-1)
140 FOR i = 1 TO np
150 LET y(i) = FN y (x)
155 LET x = x + xpas
160 NEXT i
180 PRINT : PRINT "Căutarea extremelor"
190 GO SUB 1000
210 CLS
220 LET xe = 255/(xmax-xmin)
230 LET ye = 175/(ymax-ymin)
240 LET x = xmin
250 FOR i = 1 TO np
260 PLOT (x-xmin) * xe, (y (i)-ymin) * ye
270 LET x = x + xpas
280 NEXT i
400 BEEP d, 13 : BEEP d, 16
410 STOP
990 REM
1000 LET ymin = y(1)
1010 LET ymax = y (1)
1040 FOR i = 2 TO np
1060 IF y (i) < ymin THEN LET ymin = y (i)
1060 IF y (i) > ymax THEN LET ymax = y (i)
1080 NEXT i
1100 IF ymin = ymax THEN BEEP d, 10 : BEEP d, 5 :
PRINT "Verifică coordonatele !" : STOP
1120 RETURN

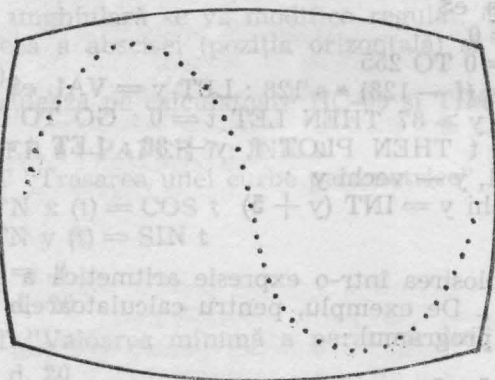
```

Observații :

- 15 — definirea funcției de studiat (s-a ales funcția sinus);
- 40— 60 — introducerea de limite variației parametrului ;
- 80 — introducerea fineței (numărul de puncte) ;
- 130 — calculul diferenței (pe orizontală) între două puncte succesive ale curbei ;
- 140— 160 — buclă de calcul a funcției

- 190 — utilizarea subprogramului de căutare a valorilor extreme ale funcției ;
- 220— 230 — calculul factorilor de scară orizontal și vertical ;
- 250— 280 — buclă de trasare a punctelor ;
- 1000—1120 — subprogram de căutare a extremelor ;
- 1040—1080 — buclă de căutare : de fiecare dată cînd este găsită o valoare y (i) mai mică decît precedentele, conținută în ymin, trebuie schimbat ymin ; același lucru pentru ymax ;
- 1100 — respingerea cazului în care punctele sînt aliniate orizontal sau vertical.

La comanda RUN programul va întreba limitele pentru argumentul t. Să presupunem că introducem 8 și 2 * PI sau 6,28. Apoi programul va întreba numărul de puncte de trasat. Dacă vom răspunde astfel, se va obține următorul ecran :



Alt exemplu, mai interesant, reprezintă amplitudinea vibrațiilor sonore rezultate din două frecvențe, apropiate (aceasta se traduce printr-un tremolo perceptibil urechii, dacă frecvențele sînt folosite apropiate). Pentru a obține acest lucru se introduce linia : 15 DEF FN

Pentru a se figura și axele de coordonate care se intersectează în punctul de origine, vor trebui adăugate următoarele instrucțiuni : $y(t) = \sin t * \sin t + \sin(1.1 * t) * \sin(1.1) * t$.

```

300 LET x8 = - min * xe
310 LET y8 = - ymin * ye
320 OVER 1
330 IF x8 >= 8 AND x8 <= 225 THEN PLOT x8, 8 : DRAW 8, 175
340 IF y8 >= 8 AND y8 <= 175 THEN PLOT 8, y8 : DRAW 255, 8
350 OVER 8

```

Pentru ilustrare se poate încerca o rulare cu modificarea instrucțiunii de definire a funcției : 15 DEF FN $y(t) = t * t - t - 2$ cu limite - 2 și + 3, cu 100 de puncte

Program de trasare de grafice de funcții de o variabilă pentru calculatoare HC-85 sau TIM-S

Aceasta este o versiune mai simplă, dar destul de eficientă, de trasare de grafice de funcții de o variabilă.

La început, programul cere introducerea unui număr n ; se vor trasa grafice între valorile $-n$ și n . Apoi, se va introduce însăși funcția (sub forma unui șir) al cărui grafic se dărește a se desena. Șirul va fi o expresie pentru care se va utiliza x ca argument al funcției. Pentru calculul valorilor funcției s-a folosit funcția VAL care convertește (evaluează) șirurile de caractere în numere (a doua instrucțiune din linia 50).

Ca exemplu de utilizare a programului, introduceți pentru n valoarea 10 și pentru funcție $10 * \text{TAN } x$. Se va obține graficul funcției $\text{TAN } x$, atunci când variază în intervalul $-10, 10$.

```
10 PLOT 0, 87 : DRAW 255, 0
20 PLOT 127, 0 : DRAW 0, 175
30 INPUT s, e$
35 LET t = 0
40 FOR f = 0 TO 255
50 LET x = (f - 128) * s / 128 : LET y = VAL e$
60 IF ABS y ≥ 87 THEN LET t = 0 : GO TO 100
70 IF NOT t THEN PLOT f, y + 88 : LET t = 1 : GO TO 100
80 DRAW 1, y - vechi y
100 LET vechi y = INT (y + 5)
110 NEXT f
```

Atenție la folosirea într-o expresie aritmetică a semnului de ridicare la putere (\uparrow). De exemplu, pentru calculatoarele HC-85 și TIM-S, dacă vom încerca programul :

```
10 LET a = 5 * 5
20 LET b = 5 ↑ 2
25 PRINT a, b
30 IF a = b THEN PRINT "O.K" : GO TO 50
40 PRINT "Nu e bine"
50 STOP
```

Vom obține, pentru instrucțiunea 25 valorile pentru a și b (bineînțeles amândouă egale cu 25), dar rezultatul : "Nu e bine", ceea ce ar sugera faptul că $5 * 5$ nu este egal cu $5 \uparrow 2$. Aceasta se întâmplă deoarece numerele sînt numerotate sub o formă cu multe cifre zecimale, iar calculul lui 5 la puterea a doua are ca rezultat, datorită modului de calcul, un număr foarte apropiat de 25, dar nu identic. Pentru a se evita unele rezultate contradictorii, dat fiind cele expuse, se va prefera introducerea funcțiilor folosind semnul de înmulțire repetat decît folosind semnul de ridicare la putere.

În utilizarea programului, propunem pentru experimentare cîteva funcții care oferă grafice interesante (în partea dreaptă se va indica valoarea s , care va fi introdusă pentru obținerea graficului în intervalul $(-s, s)$:

```

y = x ↑ 3 * cos x      30
y = cos x ↑ 3          5
y = x ↑ 5              2
y = x * SIN x         10
y = EXP x * x         2
y = (2 * x ↑ 3 - 9 * x) * (x * x - 5) 5
y = SQR (x * x + 2 * ABS x) 10
y = (2 + x) * SQR (1 - x) 2
y = (x * x - 4) / (x * x + 1) 4

```

4.1.17. Curbă parametrică

Legind evoluția coordonatelor orizontale și verticale de aceea a unui parametru auxiliar, este mai ușor de a se reprezenta o curbă. Cercul este un exemplu elocvent : se vor obține puncte regulat depărtate dacă poziția unghiulară se va modifica regulat. Vor trebui definite două funcții : aceea a abscisei (poziția orizontală) și aceea a ordonatei (poziția verticală).

Programul rulează pe calculatoare HC-85 și TIM-S.

```

5 BORDER 6 : PAPER 7 : INK 9
7 PRINT "Trasarea unei curbe parametrice"
10 DEF FN x (t) = COS t
20 DEF FN y (t) = SIN t
25 LET d = .2
30 BEEP d, 20
40 INPUT "Valoarea minimă a parametrului" ; tmin
50 BEEP d, 20
60 INPUT "Valoarea maximă a parametrului" ; tmax
70 BEEP d, 20
80 INPUT "Numărul de puncte" ; np
90 DIM x (np) : DIM y (np)
102 PRINT : PRINT "Calculul coordonatelor"
105 LET t = tmin
110 LET tpas = (tmax - tmin) / (np - 1)
120 FOR i = 1 TO np
130 LET x (i) = FN x (t)
140 LET y (i) = FN y (t)
150 LET t = t + tpas
160 NEXT i
175 PRINT : PRINT "Căutarea extremelor"
180 GO SUB 1000
195 CLS
200 LET xe = 255 / (xmax - xmin)

```

```

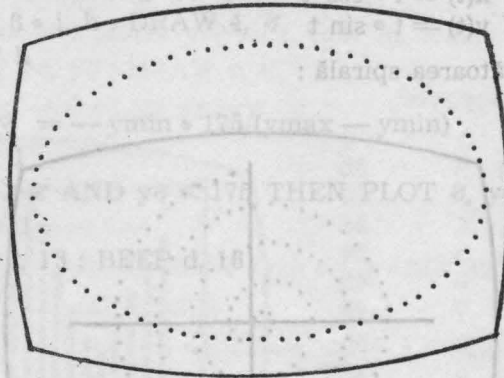
210 LET ye = 175/(ymax - ymin)
220 FOR i = 1 TO np
230 PLOT (x (i) - xmin) * xe, (y (i) - ymin) * ye
240 NEXT i
400 BEEP .2, 13 : BEEP .2, 16
410 STOP
990 REM
1000 LET xmin = x (1)
1010 LET xmax = x (1)
1020 LET ymin = y(1)
1030 LET ymax = y (1)
1040 FOR i = 2 TO np
1050 IF x (i) < xmin THEN LET xmin = x (i)
1060 IF x (i) > xmax THEN LET xmax = x (i)
1070 IF y (i) < ymin THEN LET ymin = y (i)
1080 IF y (i) > ymax THEN LET ymax = y (i)
1090 NEXT i
1100 IF xmin = xmax OR ymin = ymax THEN BEEP d, 10 :
    BEEP d, 5 : PRINT "Verifică coordonatele" : STOP
1110 RETURN

```

Observații :

- 10 — definirea funcției pentru calculul poziției orizontale a unui punct (abscisa) ;
- 20 — definirea funcției pentru calculul poziției verticale a unui punct (ordonata) ;
- 40— 60 — introducerea limitelor variației parametrilor ;
- 80 — introducerea numărului de puncte ;
- 90 — rezervarea de memorie pentru coordonate ;
- 110 — calculul definiției dintre două valori succesive a parametrului ;
- 120— 160 — buclă de calcul a coordonatelor punctelor ;
- 180 — utilizarea subprogramului de căutare a extremelor printre valorile absciselor și ale ordonatelor ;
- 220— 240 — buclă de trasare a punctelor ;
- 1000—1100 — subprogram de căutare a extremelor ;
- 1000—1030 — inițializare ;
- 1040—1090 — buclă de căutare : de fiecare dată când se caută o valoare $x (i)$ mai mică decât precedentele, conținută în $xmin$, se schimbă valoarea lui $xmin$; același lucru pentru $ymin$, $xmax$ și $ymax$;
- 1100 — respingerea unei curbe care se reduce la o linie orizontală sau verticală.

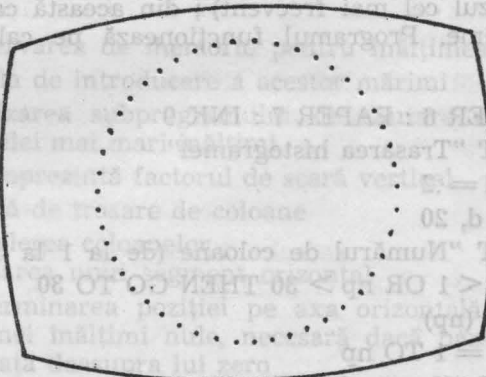
Cu limitele 0, 2 *PI și 200 de puncte se va obține ecranul următor:



Pentru a înțelege de ce s-a obținut un oval și nu un cerc, trebuie să ne reamintim că ecranul la HC-85 și TIM-S este mai lat decât înalt. Pentru a obține un cerc trebuie să se modifice factorul de scară orizontală determinat la linia 200 și poziția centrului (decalată spre dreapta) la linia 230. Noile linii vor fi :

```
200 LET xe = 175/(xmax-xmin)
230 PLOT 40 + (x(i) - xmin) * xe, (y (i) - ymin)*ye
```

Rezultatul va fi :



Pentru trasarea axelor de coordonate se vor adăuga liniile 300—350.

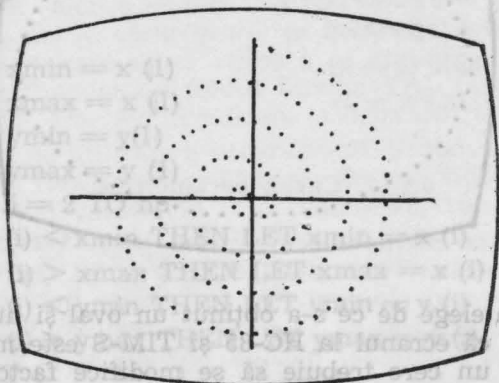
```
300 LET x0 = - xmin * xe + 40
310 LET y0 = - ymin * ye
320 OVER 1
330 IF x0 ≥ 0 AND x0 ≤ 255 THEN PLOT x0, 0 : DRAW 0, 175
340 IF y0 ≥ 0 AND y0 ≤ 175 THEN PLOT 0, y0 : DRAW 255, 0
350 OVER 0
```

Schimbându-se liniile 10 și 20 astfel :

```
10 DEF FN x(t) = t * cos t
```

```
20 DEF FN y(t) = t * sin t
```

se va obține următoarea spirală :



4.1.18. Histograma

Histograma este un mod de reprezentare grafică vizuală, care face să apară sub formă de coloane evoluția unui fenomen în timp. Presupunem că variația parametrului care este observat este regulată (este cazul cel mai frecvent); din această cauză coloanele vor avea aceeași lățime. Programul funcționează pe calculatoarele HC-85 și TIM-S

```
5 BORDER 6 : PAPER 7 : INK 9
10 PRINT "Trasarea histogramei"
20 LET d = .2
30 BEEP d, 20
40 INPUT "Numărul de coloane (de la 1 la 30)"; np
45 IF np < 1 OR np > 30 THEN GO TO 30
50 DIM y (np)
70 FOR i = 1 TO np
80 PRINT "Numărul coloanei"; i;
90 BEEP d, 20
100 INPUT "valoarea"; y (i) : PRINT TAB 20; y (i)
110 NEXT i
130 PRINT "Căutarea extremelor"
140 GO SUB 1000
160 CLS : INK 4
165 LET ye = 170/(ymax)
```

```

170 FOR i=1 TO np
180 FOR h=z TO (y (i)* ye)
190 PLOT s * i, h : DRAW 4, z
200 NEXT h
210 NEXT i
310 LET yz = — ymin * 175/(ymax — ymin)
320 OVER 1
340 IF yz > z AND yz < 175 THEN PLOT z, yz : DRAW 255, z
350 OVER 1
390 BEEP d, 13 : BEEP d, 16
400 INK 9
410 STOP
990 REM
1000 LET ymax = y (1)
1010 LET ymin = y (1)
1040 FOR i= 2 TO np
1060 IF y (i) < ymin THEN LET ymin = y (i)
1070 IF y (i) > ymax THEN LET ymax = y (i)
1080 NEXT i
1100 IF ymax — ymin = z THEN BEEP d, 10 ; BEEP d, 5 :
PRINT "Verifică coordonatele !" : STOP
1120 RETURN

```

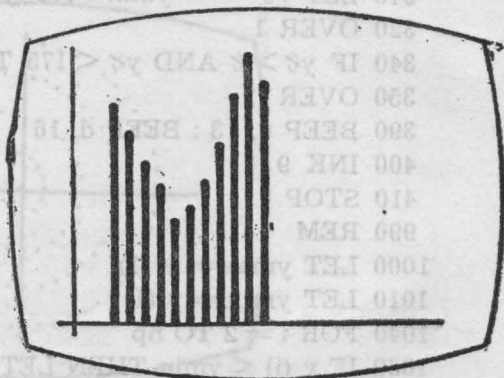
Observații :

- 50 — rezervarea de memorie pentru înălțimea fiecărei coloane
- 70— 110 — bucla de introducere a acestor mărimi
- 140 — utilizarea subprogramului de căutare a celei mai mici și celei mai mari înălțimi
- 165 — ye reprezintă factorul de scară vertical
- 170— 210 — buclă de trasare de coloane
- 180— 200 — umplerea coloanelor
- 190 — trasarea unui segment orizontal
- 310 — determinarea poziției pe axa orizontală corespunzătoare a unei înălțimi nule, necesară dacă baza coloanelor este situată deasupra lui zero
- 320 — axa va fi trasată cu suprainprimare
- 1000—1120 — subprogram de căutare a extremelor verticale
- 1000—1010 — ipoteză de pornire : prima valoare este în același timp cea mai mare și cea mai mică
- 1040—1080 — buclă de observare a celorlalte valori
- 1060 — dacă o valoarea mică este găsită, ea este conservată în ymin
- 1070 — același lucru pentru cea mai mare valoare

Exemplu de utilizare : reprezentarea înălțimii căderilor de apă (în mm) măsurate într-o localitate în cursul a 12 luni dintr-un an. Coloana numărul unu este fixată la zero pentru a se determina baza graficului; următoarele reprezintă în mod succesiv lunile ianuarie, februarie etc.

Trasare histogramă

Coloana numărul 1	0
Coloana numărul 2	80
Coloana numărul 3	72
Coloana numărul 4	63
Coloana numărul 5	54
Coloana numărul 6	40
Coloana numărul 7	45
Coloana numărul 8	50
Coloana numărul 9	55
Coloana numărul 10	65
Coloana numărul 11	85
Coloana numărul 12	100
Coloana numărul 13	85



4.1.19. Diagrama circulară

Pentru compararea vizuală a importanței relative a mai multor valori, se poate utiliza o diagramă circulară. Fiecare valoare este reprezentată de un sector, a cărui talie este proporțională cu valoarea respectivă. Reuniunea tuturor sectoarelor formează un cerc complet, ceea ce corespunde cu 100%.

```

5 BORDER 6 : PAPER 7 : INK 9
10 PRINT "Diagrama circulară"
15 PRINT
20 LET d = .2
30 BEEP d, 20
40 INPUT "Număr de valori" ; np
45 IF np < 1 THEN GO TO 30
50 DIM y (np)
60 LET s = 0
70 FOR i = 1 TO np
80 PRINT "valoarea numărul" ; i ;
90 BEEP d, 20
100 INPUT "Valoarea" ; y (i) : PRINT TAB 20 ; y (i)
110 LET s = s + y (i)
120 NEXT i
130 CLS
140 LET x0 = 127 : LET y0 = 87

```

```

150 LET r = 60
160 CIRCLE x%, y%, r
165 LET ae = 2 * PI/s
170 PLOT x%, y% : DRAW r, %
180 LET sy = 0 : LET a1 = %
185 FOR i = 1 TO np
190 LET sy = sy + y (i)
195 LET a2 = sy * ae
200 PLOT x%, y% : DRAW r * COS a2, r * SIN a2
205 LET am = (a1 + a2)/2
210 PRINT AT 11 - 4 * SIN am, 15 + 5 * COS am ; y (i)
220 LET a1 = a2
230 NEXT i
390 BEEP d, 13 : BEEP d, 16
410 STOP

```

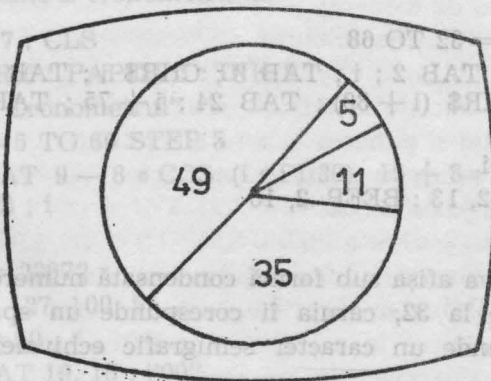
Observații :

- 140 — amplasarea centrului cercului ;
- 165 — factor de scară unghiular ;
- 190 — cumularea valorilor
- 210 — afișarea valorii în interiorul sectorului.

Exemplu :

Într-o centrală industrială, unitatea A produce 11% din producția centralei, unitatea B 5%, unitatea C 49%, iar unitatea D 35% din producția totală a centralei.

Reprezentarea grafică va arăta astfel :



Dacă unele valori nu pot intra în sectoarele lor, se pot renumera sectoarele și afișa o legendă în afara diagramei. În acest caz, trebuie adăugate liniile pentru afișarea legendei :

```
240 PRINT AT 1, 20 ; "legenda"  
250 FOR i = 1 TO np  
260 PRINT AT i + 2, 20 ; i ; TAB 24 ; y (i)  
270 NEXT i
```

și modificată linia 210

```
210 PRINT AT 11 - 4 * SIN am, 9 + 5 * COS am ; i
```

4.2. OPERAȚII CARE NECESITĂ COMPARAREA ȘIRURILOR DE CARACTERE

4.2.1. Coduri ASCII

Deoarece compararea a 2 șiruri de caractere (necesare în realizarea unor programe) se face după codificarea ASCII, este necesară cunoașterea acestora sau modul de obținere a lor.

Fiecărui caracter (literă a alfabetului, cifră etc.), îi corespunde un cod numeric numit cod ASCII. Astfel, după cum vom vedea, de exemplu, semnului " " îi corespunde codul 34, cifrei 3 — codul 51, literei A — codul 65 etc.

Codurile ASCII se pot obține cu ajutorul calculatorului prin intermediul funcției CHR\$ cu care se va obține codul ASCII al argumentului numeric.

Prezentăm un program prin care se va obține tabelul condensat al codurilor ASCII pentru calculatoare HC-85 și TIM-S :

```
10 PRINT "cod car. cod car. cod car."  
20 BEEP .2,20  
60 PRINT  
70 FOR i = 32 TO 68  
80 PRINT TAB 2 ; i ; TAB 8 ; CHR$ i ; TAB 13 ; i + 38 ; TAB  
19 ; CHR$ (i + 38) ; TAB 24 ; i + 75 ; TAB 30 ; CHR$  
(i + 75)  
100 NEXT i  
130 BEEP .2, 13 : BEEP .2, 16  
140 STOP
```

Programul va afișa sub formă condensată numerele și codul ASCII corespunzător de la 32, căruia îi corespunde un spațiu, până la 143, căruia îi corespunde un caracter semigrafic echivalent cu un pătrățel negru.

4.2.2. Cronometru

Ceasul care conferă o anumită frecvență operațiilor elementare într-un calculator personal poate, în particular, să servească drept ceas în sens clasic. Se pot astfel măsura duratele din secundă în secundă. La calculatoarele HC-85 și TIM-S acest lucru se realizează citindu-se adresa de memorie 23672.

Iată o primă versiune de cronometru :

```
10 PRINT "Numărător de la 0 la 60 secunde"
20 LET h = 23672
30 POKE h, 1
40 PRINT AT 10, 15 ; "00"
60 FOR s = 1 TO 60
70 IF PEEK h < 50 THEN GO TO 70
80 BEEP .04, 30 : POKE h, 3
90 PRINT AT 10, 15 ;
100 IF s < 10 THEN PRINT 0 ;
110 PRINT s
120 NEXT s
```

Observații: În instrucțiunea 20— h este adresa octetului de "ceas"; 30— inițializarea acestui octet (1 pentru a se ține cont de timpul de executare a instrucțiunii POKE); 80— notă muzicală, durind 4/1000 sec., apoi punerea la "zero" a octetului de "ceas" (3 și nu 0 pentru a se ține cont de durata sunetului care întrerupe evoluția octetului, atât timp cât durează execuția instrucțiunii POKE); 90—110 afișarea numărului de secunde scurse (în două cifre).

Versiunea cu dublu afișaj, analogic și digital

Se adaugă în afara dublului afișaj posibilitatea întreruperii și repunerii în funcțiune a cronometrului.

```
5 PAPER 7 : CLS
10 BORDER 6 : PAPER 5 : INK 0
15 PRINT "Cronometrul"
20 FOR i = 5 TO 60 STEP 5
30 PRINT AT 9 - 8 * COS (i * PI/30), 15 + 8 * SIN
   (i * PI/30) ; i
40 NEXT i
50 LET h = 23672
60 CIRCLE 127, 100, 55
70 LET l = 50
80 PRINT AT 19, 16 ; "00"
```

```

90 PLOT 127, 100
100 DRAW 0, 1
110 PRINT AT 21, 2 ; "Pentru a începe, 'ENTER'"
120 IF INKEY$ = "" THEN GO TO 120
130 POKE h, 1
140 LET a = 0
150 FOR s = 1 TO 60
160 PRINT AT 21, 2 ; "Pentru oprire, 'SPACE'"
170 IF INKEY$ = "" THEN GO TO 400
180 IF PEEK h < 50 THEN GO TO 170
190 BEEP .04, 30 : POKE h, 3
200 PRINT AT 19, 16 ;
210 IF s < 10 THEN PRINT 0 ;
220 PRINT s
230 PLOT 127, 100
240 DRAW OVER 1 ; 1 * SIN a, 1 * COS a
250 LET a = s * PI / 30
260 PLOT 127, 100
270 DRAW 1 * SIN a, 1 * COS a
280 NEXT s
350 PRINT AT 21, 2 ; "SFÎRȘIT !"
360 BEEP .2, 13 : BEEP .2, 16
370 STOP
400 LET ti = PEEK h
410 PRINT AT 21, 2 ; "Pentru a relua, ENTER"
420 IF INKEY$ <> CHR$ 13 THEN GO TO 420
430 POKE h, ti + 1
440 GO TO 160

```

Observații : instrucțiunile 20—40 : afișarea numerelor de la 5 la 60 din 5 în 5 ; 60 bordul ceasului analogic ; 80 afișarea numerică inițială.

90—100 trasarea limbii în repaus
110—120 faza de demaraj
130 inițializarea octetului de "ceas"
140 inițializarea unghiului de reper al limbii
160—170 posibilitate de întrerupere provizorie
190 sunet și punerea la zero a octetului
200—220 actualizarea afișajului numeric
230—240 ștergerea limbii
250 noua poziție a limbii
400 memorarea conținutului octetului de "ceas"
410—420 posibilitatea ruperii în mers
430 înlocuirea în octetul de "ceas" cu conținutul în momentul întreruperii

4.2.3. Agenda

De multe ori, utilizarea unui calculator personal pentru înregistrarea și apoi regăsirea (atunci când este necesar) a unor mesaje, note, indicații referitoare la unele zile dintr-o lună sau dintr-un an este foarte eficientă. De exemplu : anumite întâlniri importante, zilele de naștere ale rudelor sau prietenilor, termene de lucrări etc.

Cu ajutorul programului Agenda, funcțional pe calculatoarele HC-85 sau TIM-S, se pot introduce mesaje de diverse lungimi pentru orice dată și, mai mult, pentru o anumită dată, se pot introduce mai multe mesaje. Modul de lucru este ușurat de faptul că pe ecran sînt afișate opțiunile pe care utilizatorul le poate alege :

- (1) introducere de mesaje ;
- (2) ștergerea din memorie a înregistrărilor de la o anumită dată ;
- (3) afișarea înregistrărilor dintr-o anumită perioadă, care se va specifica prin introducerea de către utilizator a datei de început a intervalului și a datei de sfîrșit a intervalului ;
- (4) salvarea programului Agenda împreună cu înregistrările făcute pentru o utilizare ulterioară.

Exemplu de introducere pentru o dată :

— se afișează pe ecran cuvîntul Ziua ; utilizatorul va tasta 28 sau 3 sau 03 (data respectivă) și <CR> ;

— se afișează pe ecran cuvîntul Luna ; utilizatorul va tasta numărul lunii din an, adică, pentru ianuarie 1 sau 01, pentru august 8 sau 08 și <CR> ;

— se afișează pe ecran cuvîntul Anul ; utilizatorul va tasta anul 1987 sau 87.

Conform instrucțiunilor din MENU se pot afișa toate mesajele dintr-o anumită perioadă sau se pot șterge.

Agenda se lansează în execuție automat, de la linia 1520, conform salvării ei cu instrucțiunea din linia 1510, făcînd și o verificare a înregistrării ei pe caseta magnetică (imediat după încărcare), conform instrucțiunilor din liniile 1520 și 1540.

Dacă la execuția programului, în mod accidental, se iese din program în sistemul BASIC, Agenda se va putea relansa cu comanda : GO TO 130, comanda RUN 130 ștergînd înregistrările făcute pînă în acel moment.

```
10 REM Pregătire
20 DEF FN f (x) = INT (x - 100 * INT (x/100))
30 DEF FN f$ (x) = ("O" AND FN f (x) < 10) + STR$ FN f (x)
40 DEF FN g$ (x$, x) = x$ (x + 4 TO x + 5) + "/" + x$
   (x + 2 TO x + 3) + "/" + x$ (x TO x + 1)
50 LET a$ = " " ; LET l$ = " "
60 LET c = 0
```

```

70 DIM b$ (5)
100 REM Menu
110 PRINT AT 20, 0 ; "Apasă orice pentru a continua",,,
120 PAUSE 0
130 CLS
140 PRINT AT 2, 10 ; "Agenda"
150 PRINT AT 5, 5 ; " (1) ... Introducere mesaj"
160 PRINT AT 7, 5 ; " (2) ... Șterge ziua"
170 PRINT AT 9, 5 ; " (3) ... Afișează"
180 PRINT AT 11, 5 ; " (4) ... Salvează Agenda"
190 PRINT AT 21, 0 ; " INTRODUCEȚI opțiunea"
200 INPUT n : LET n = INT n : IF n < 1 OR n > 4 THEN GO
    TO 200
210 CLS
220 IF n = 1 THEN GO TO 300
230 GO TO 300 * n + 300
300 REM Introducerea datelor
310 PRINT AT 0, 12 ; "Începutul agendei"
320 LET c = c + 1
330 LET m$ = " " : LET a = 1
340 INPUT "Ziua" ; d' "Luna" ; m' "Anul" ; y'
350 LET d$ = FN f$ (y) + FN f$ (m) + FN f$ (d) : LET
    e$ = FN g$ (d$, 1)
360 CLS : PRINT AT 0, 0 ; e$ ; TAB 11 ; "Introducere în agendă"
370 INPUT "Paragraf" ; (a), i$ : PRINT AT 2, 2 ; i$
380 LET l = LEN i$ - 32 * INT (LEN i$ / 32)
390 FOR z = 1 TO 29 : LET i$ = i$ + ". " : NEXT z
400 LET m$ = m$ + ". " + i$
410 PRINT AT 21, 0 ; "Mai sînt introduceri ? (d/n)"
420 IF INKEY$ = "d" THEN LET a = a + 1 : GO TO 360
430 IF INKEY$ <> "n" THEN GO TO 410
440 LET b$ = STR$ (LEN a$ + 1) : LET l$ = l$ + b$
450 PRINT AT 20, 0 ; m$ : LET a$ = a$ + m$
460 LET b$ = STR$ (LEN m$) : LET l$ = l$ + b$ + d$
470 FOR z = 1 TO 16 * c - 16 STEP 16
480 LET p = VAL l$ (z TO z + 4)
490 LET r = (l$ (z + 10 TO z + 15) = d$) + (2 AND l$ (z + 10
    TO z + 15) > d$)
500 IF r = 1 THEN LET c = c - 1 : GO TO 660
510 IF r = 2 THEN GO TO 540
520 NEXT z
530 GO TO 100

```



```

540 LET v = VAL 1$ (16 * c - 15 TO 16 * c - 11) : LET
    p = VAL 1$ (z TO z + 4)
550 LET a$ = a$ (TO p - 1) + a$ (v TO) + a$ (p TO v - 1)
560 LET h$ = 1$ (16 * c - 10 TO 16 * c)
580 LET 1$ (y + 21 TO y + 31) = 1$ (y + 5 TO y + 15)
590 NEXT y
600 LET 1$ (z + 5 TO z + 15) = h$
610 FOR y = z TO c * 16 - 16 STEP 16
620 LET b$ = STR$ (VAL 1$ (y TO y + 4) + VAL 1$ (y + 5
    TO y + 9))
630 LET 1$ (y + 16 TO y + 20) = b$
640 NEXT y
650 GO TO 100
660 FOR y = z + 16 TO 16 * c STEP 16
670 LET 1$ (y TO y + 4) = STR$ (VAL 1$ (y TO y + 4) + LEN
    m$)
680 NEXT y
690 LET v = VAL 1$ (z TO z + 4) + VAL 1$ (z + 5 TO z + 9)
700 LET a$ = a$ (TO v - 1) + m$ + a$ (v TO LEN a$ - LEN
    m$)
710 LET 1$ (z + 5 TO z + 9) = STR$ (VAL 1$ (z + 5 TO
    z + 9) + LEN 1$ (z + 5 TO z + 9) + LEN m$)
720 LET 1$ = 1$ (TO LEN 1$ - 16)
730 GO TO 100
900 REM Șterge din memorie
910 PRINT AT 0, 10 ; "Șterge înregistrarea"
920 INPUT "Ce zi șterg ?" "Ziua" ; d "Luna" ; m "Anul" ; y'
930 LET d$ = FN f$ (y) + FN f$ (m) + FN f$ (d)
940 LET e$ = FN g$ (d$, 1)
950 FOR z = 1 TO 16 * c STEP 16
960 LET p = VAL 1$ (z TO z + 4)
970 IF 1$ (z + 10 TO z + 15) = d$ THEN GO TO 1010
980 IF 1$ (z + 10 TO z + 15) < d$ THEN NEXT z
990 PRINT AT 5, 3 ; e$ ; "nu este înregistrată"
1000 GO TO 100
1010 LET a$ = a$ (TO p - 1) + a$ (VAL 1$ (z + 5 TO z + 9)
    + p TO)
1020 LET v = VAL 1$ (z TO z + 4)
1030 FOR y = z TO 16 * c - 20 STEP 16
1040 LET 1$ (y + 5 TO y + 15) = 1$ (y + 21 TO y + 31)
1050 LET y = VAL 1$ (y TO y + 4) + VAL 1$ (y + 5 TO y + 9)
1060 LET 1$ (y + 16 TO y + 20) = STR$ v
1070 NEXT y
1080 LET 1$ = 1$ (TO 16 * c - 16)
1090 PRINT AT 5, 8 ; "Înregistrarea : " ; e$ ; TAB 12 ; "este
    ștearsă"
1100 LET c = c - 1
1110 GO TO 100

```

```

1200 REM Extragere
1210 PRINT AT 0,9 ; "Introducere"
1220 INPUT "De unde ?" "Ziua"; d "Luna"; m "Anul";
1230 LET d$ = FN f$ (y) + FN f$ (m) + FN f$ (d)
1240 INPUT "Pină unde [" "Ziua"; d "Luna"; m "Anul"; y
1250 LET e$ = FN f$ (y) + FN f$ (m) + FN f$ (d)
1260 IF e$ < d$ THEN LET i$ = e$ : LET e$ = d$ : LET
      d$ = i$
1270 LET f = 0
1280 FOR z = 1 TO 16 * c STEP 16
1290 LET p = VAL 1$ (z TO z + 4)
1300 IF d$ < 1$ (z + 10 TO z + 15) THEN GO TO 1340
1310 NEXT z
1320 LET f = 1
1330 GO TO 1440
1340 IF e$ < 1$ (z + 10 TO z + 15) THEN LET f = z : GO TO
      1410
1350 CLS
1360 PRINT AT 0, 0 ; FN g$ (1$, z + 10)
1370 PRINT AT 2, 0 ; a$ (p TO p - 1 + VAL 1$ (z + 5 TO z + 9))
1380 PRINT AT 20, 0 ; "Apasă 'd' pentru a continua"
1390 PRINT AT 21, 0 ; "Apasă 'n' pentru sfârșit display"
1400 INKEY $ = "n" THEN GO TO 130
1410 IF INKEY$ < > "d" THEN GO TO 1400
1420 NEXT z
1430 IF f < > 1 THEN FOR z = 1 TO 50 : NEXT z : GO TO 100
1440 LET d$ = FN g$ (d$, 1)
1450 LET e$ = FN g$ (e$, 1)
1460 PRINT AT 5, 7 ; "Nici o înregistrare între"; TAB 12 ; d$ ;
      "și"; TAB 12 ; e$
1470 GO TO 100
1500 REM Salvare
1510 SAVE "Agenda" LINE 1520
1520 PRINT AT 5, 5 ; "Derulează banda — pentru verificare";
      AT 7, 5 ; "Apasă orice tastă când ești gata"
1530 PAUSE 0
1540 VERIFY "Agenda"
1550 GO TO 100

```

4.2.4. Fișier, microbază de date

Calculatoarele pot fi utilizate (și aceasta este una din aplicațiile frecvente ale calculatoarelor) și pentru a mînuî fișiere, adică colecții de date. Pentru a se înțelege mai bine ce este un fișier, putem să ne închipuim acesta ca o cutie în care sînt ținute informațiile într-un mod organizat, de exemplu : în sertare sau pe niște fișe.

Într-un mod similar, calculatorul poate stoca informațiile pe caseta magnetică (sau discul flexibil) utilizînd fișiere de date.

Utilizarea unui fișier — microbază de date — trebuie să permită câteva posibilități de bază : crearea fișierului, regăsirea unei (unor) înregistrări (articole) din fișier, modificarea unei înregistrări, adăugarea de noi înregistrări, salvarea fișierului pe un suport extern de memorie. De asemenea, unele fișiere sînt proiectate pentru a admite sortări ale înregistrărilor (alfabetice sau numerice) după anumite chei sau diverse operații (calcul) asupra înregistrărilor din fișier.

În exemplul ales pentru calculatorul HC-85 sau TIM-S se va putea crea un fișier al cunoștințelor (maximum 370 de persoane), cu detalii despre acestea (telefon, adresă etc.), iar acest fișier se va putea salva pe casetă sau lista la imprimantă.

Informațiile care se vor înregistra pentru fiecare cunoștință sînt :

— numărul critic. Numerele sînt alocate secvențial pe măsură ce noi articole (cunoștințe) sînt adăugate fișierului, pornindu-se cu un număr inițial introdus atunci cînd fișierul este creat ;

— nume, număr de telefon și adresă, pînă la un total de 94 de caractere ;

— două „chei“ a cîte 3 caractere, care pot fi utilizate pentru a păstra informații ca luna și data nașterii.

Dacă programul se întrerupe, el nu se va relua cu RUN, căci această comandă va distruge toate înregistrările. În loc de RUN se va utiliza GO TO 1000.

Pătratul negru este un „cursor“ care este utilizat în program pentru a semnala următoarea dată de introdus. În acest caz, se poate alege din :

```
100 LET m = 370 : DIM d$(m, 100)
110 DIN z$(25) : DIM x$(3) : DIM k$(1)
120 INK 0 : PAPER 7 : FLASH 0 : BRIGHT 0 : OVER 0 : IN-
    VERSE 0 : BORDER 7 : CLS
200 PRINT AT 8, 4 : "FIȘIER NOU"
210 PRINT AT 15, 0 : "NUMĂRUL PRIMULUI MEMBRU" ;
    FLASH 1 : " "
220 GO SUB 9000 : IF i = 0 THEN BEEP .1, 20 : GO TO 220
230 LET primul = i : LET următorul = i : LET CR = 0
1000 REM listare meniu principal
1010 CLS
1020 PRINT AT 3, 4 : "Nr : " ; AT 5, 2 : "Nume : "
1030 PRINT AT 7, 3 : "Tel : " ; AT 9, 3 : "Adr : "
1040 PRINT AT 10, 7 : " : " ; AT 11, 7 : " : " ; AT 12, 7 :
    " : " ; AT 13, 7 : " : "
1050 PRINT AT 15, 0 : "che 1 : " ; AT 16, 0 : "che 2 : "
2000 REM alegerea din meniul principal
2010 PRINT AT 21, 1 : "Număr + — M N P S" ; FLASH 1 : " ."
2020 GO SUB 9000 : PRINT AT 21, 0, : IF i > 0 THEN GO TO
    3000
2030 IF CODE i$ > 90 THEN LET i$ = CHR$(CODE i$ - 32)
2040 IF i$ = "+" THEN GO TO 3100
2050 IF i$ = "-" THEN GO TO 3200
```

```

2060 IF i$ = "M" THEN GO TO 4000
2070 IF i$ = "N" THEN GO TO 4100
2080 IF i$ = "P" THEN GO TO 5000
2090 IF i$ = "S" THEN GO TO 6000
2100 GO TO 2000
3000 REM regăsirea numărului înregistrării
3010 IF i < primul OR ≥ următorul THEN GO TO 2000
3020 LET CR = i : GO TO 3300
3100 REM regăsirea înregistrării următoare
3110 IF CR ≥ următor - 1 THEN GO TO 2000
3120 LET CR = CR + 1 : GO TO 3300
3200 REM regăsirea înregistrării anterioare
3210 IF CR ≤ primul THEN GO TO 2000
3220 LET CR = CR - 1
3300 REM regăsirea înregistrării curente
3310 FOR a = 3 TO 16 : PRINT AT a, 7 ; z$ : NEXT a
3320 PRINT AT 3, 8 ; CR : LET poz = 7
3330 FOR l = 5 TO 13 : IF l = 6 OR l = 8 THEN LET l = l + 1
3340 PRINT AT l, 8 ;
3350 IF poz > 100 THEN GO TO 3500
3360 LET q = CODE d$ (CR + 1 - primul, poz) : LET
    poz = poz + 1
3370 IF q < 128 THEN PRINT CHR$ q ; : GO TO 3350
3380 PRINT CHR$ (q - 128) ;
3390 NEXT l
3500 PRINT AT 15, 8 ; d$ (CR + 1 - primul, TO 3)
3510 PRINT AT 16, 8 ; d$ (CR + 1 - primul, 4 TO 6)
3520 GO TO 2000
4000 REM modificarea înregistrării curente
4010 IF CR < primul OR CR ≥ următor THEN GO TO 2000
4020 GO TO 4200
4100 REM adăugarea de noi înregistrări
4110 IF următor ≥ primul + m THEN PRINT AT 19, 0 ; FLASH 1 ;
    "FIȘIER PLIN" : BEEP 1, 10 : GO TO 2000
4120 LET CR = următor : LET următor = următor + 1
4130 FOR a = 3 TO 16 : PRINT AT a, 7 ; z$ : NEXT a
4200 REM introducerea datelor pentru înregistrare
4210 PRINT AT 3, 8 ; CR : LET poz = 7
4220 FOR l = 5 TO 13 : l = 6 OR l = 8 THEN LET l = l + 1
4230 PRINT AT l, 7 ; FLASH 1 ; " ."
4240 LET j$ = z$ : IF poz > 100 THEN GO TO 4300
4250 INPUT LINE i$ : IF LEN i$ > 24 THEN LET i$ = i$ (TO 24)
4260 IF i$ = "" THEN LET i$ = " ."
4270 LET k = LEN i$ : IF poz + k > 100 THEN LET i$ = i$
    (TO 101 - poz)
4280 LET k = LEN i$ : LET j$ (2 TO k + 1) = i$ : LET i$
    (k) = CHR$ (CODE i$ (k) + 128)
4290 LET d$ (CR + 1 - primul, poz TO poz + k - 1) = i$ : LET
    poz = poz + k

```



```

4300 PRINT AT 1, k ; j$
4310 NEXT 1
4320 PRINT AT 15, 7 ; FLASH 1 ; ". ."
4330 INPUT LINE i$ : LET d$ (CR + 1 — primul, TO 3) = i$
4340 PRINT AT 15, 7 ; ". ." ; d$ (CR + 1 — primul, TO 3)
4350 PRINT AT 16, 7 ; FLASH 1 ; ". ."
4360 INPUT LINE i$ : LET d$ (CR + 1 — primul, 4 TO 6) = i$
4370 PRINT AT 16, 7 ; ". ." ; d$ (CR + 1 — primul, 4 TO 6)
4380 GO TO 2000
5000 REM listare imprimantă
5010 CLS : PRINT AT 5, 0 ; "che 1 : " ' ' "che 2 : " ' ' " tot : "
5020 PRINT AT 5, 7 ; FLASH 1 ; ". ."
5030 INPUT LINE i$ : LET x$ = i$ : IF i$ <> "" THEN LET
i$ = x$
5040 PRINT AT 5, 7 ; ". ." ; i$ ; AT 7, 7 ; FLASH 1 ; 2. "
5050 INPUT LINE j$ : LET x$ = j$ : IF j$ <> "" THEN LET
j$ = x$
5060 PRINT AT 7, 7 ; ". ." ; j$ ; AT 9, 7 ; FLASH 1 ; ". ."
5070 INPUT LINE k$ : IF CODE k$ > 90 THEN LET k$ = CHR$
(CODE k$ — 32)
5080 IF k$ <> "D" AND k$ <> "N" THEN GO TO 5070
5090 PRINT AT 9, 7 ; ". ." ; k$
5100 LPRINT : LPRINT
5110 FOR c = primul TO următor — 1 : LET l$ = d$
(c — primul + 1)
5120 IF (i$ <> "" AND i$ <> i$ (TO 3)) OR (j$ <> "" AND
j$ <> j$ (4 TO 6)) THEN GO TO 5230
5130 IF k$ = "D" THEN LPRINT
5140 LPRINT c ; l$ (TO 6) ; ". ."
5150 LET p = 7
5160 FOR l = 1 TO 7 : IF k$ = "N" AND l > 1 THEN GO TO
5220
5170 IF p < 100 THEN GO TO 5210
5180 LET q = CODE l$ (p) : LET p = p + 1
5190 IF q < 128 THEN LPRINT CHR$ q ; : GO TO 5180
5200 LPRINT CHR$ (q — 128) ;
5210 LPRINT
5220 NEXT l
5230 NEXT c
5240 LPRINT
5250 GO TO 1000
6000 REM salvare date și program
6010 INPUT "Nume fișier" Line n$ : IF n$ = "" THEN GO TO
6010
6020 IF LEN n$ > 10 THEN LET n$ = n$ (TO 10)
6030 LET i$ = "" : SAVE n$ LINE 6100
6040 CLS : PRINT AT 6, 0 ; "Acum se pot verifica datele" " " sal-
vate pe bandă : "
6100 INPUT "Verificare (d/n) ?" ; LINE i$

```



```

6110 IF i$ = "n" OR i$ = "N" THEN GO TO 1000
6120 IF i$ <> "d" AND i$ <> "D" THEN GO TO 6100
6140 PRINT AT 20, 0 ; "Rebobinați pentru verificare"
6150 LET i$ = " " : VERIFY n$
6160 GO TO 1000
9000 LET i = 0 : INPUT LINE i$
9010 FOR a = 1 TO LEN i$
9020 IF (i$ (a) < "0" OR i$ (a) > "9") AND i$ (a) <> " ." THEN
GO TO 9040
9030 NEXT a
9040 IF a > 1 THEN LET i = VAL i$ (TO a - 1)
9050 RETURN

```

Într-un program (ca cel din exemplul dat) în care se memorează date de lungime variabilă (în cazul nostru numele și adresa), este întotdeauna o neconcordanță între utilizarea de înregistrări de lungime fixă, care pot fi accesate rapid, dar care sînt consumatoare de spațiu de memorie (întrucît au fost în prealabil definite ținînd cont de cea mai lungă dată posibilă) și utilizarea de înregistrări de lungime variabilă pentru care algoritmul de accesare este mai lent. Programul ales face un compromis prin utilizarea unei lungimi fixe de 100 de octeți pentru o înregistrare (cunoștința), permițînd însă și înregistrări de date de lungime variabilă (de exemplu : numele) în cadrul unui articol. Variabilele mai importante din program sînt :

- m — nr. maxim de înregistrări din program (maxim 270 de cunoștințe) ;
- d\$(m, 100) — matrice utilizată pentru a memora fișierul de date. Fiecare înregistrare ocupă 100 de octeți, din care primii 6 reprezintă cele două chei de octeți. Numele, numărul de telefon și adresa sînt reținute în cei 94 octeți rămași ;
 - 128 este adăugat codului ultimului caracter în fiecare articol pentru a acționa ca un separator de sfîrșit de linie ;
- primul — primul număr al cunoștinței din fișier ;
- CR — numărul curent al cunoștinței din fișier care se accesează ;
- următor — numărul de ordine al cunoștinței următoare care se adaugă la fișier ;
- Număr — introducerea numărului de ordine al unei cunoștințe va avea ca efect afișarea datelor legate de acea cunoștință ;
- + — afișarea detaliilor legate de cel mai mare număr de ordine al unei cunoștințe ;

- afișarea detaliilor legate de cunoștința cu număr de ordine anterior ;
- M — modificarea datelor cunoștinței cu numărul de ordine curent ;
- N — adăugarea de noi articole (cunoștințe) fișierului ;
- P — listarea la imprimantă a fișierului ;
- S — salvarea programului și a fișierului pe caseta magnetică.

Dacă se va opta pentru M sau N, cursorul se va muta imediat după „Nume” și va trebui introdus numele cunoștinței. După ce s-a realizat acest lucru, cursorul se va muta pe rândul de mai jos, cerînd introducerea numărului de telefon și apoi adresa.

Dacă se vor introduce mai multe caractere pentru un articol, programul va ignora ceea ce este în exces și va afișa numai acele caractere pentru care există loc.

Dacă se va apela la opțiunea P (Prins), afișarea se va schimba în :

che 1 :
che 2 :
tot :

și se va aștepta introducerea de date pentru aceste trei linii înainte de a se începe listarea. Pentru che 1 și che 2 se va putea introduce direct ENTER sau o cheie de maximum 3 caractere. În acest ultim caz se vor putea lista de exemplu : toate cunoștințele care au ziua de naștere în luna august, dacă prima cheie va fi "AUG".

Ultima linie (tot) necesită un răspuns de da (D) sau nu (N). Pentru D se vor lista toate detaliile pentru fiecare cunoștință. Pentru N se va lista numai numărul de ordine, cele două chei și numele cunoștinței.

În final, opțiunea S (Save) este utilizată pentru salvarea ultimei versiuni a fișierului pe casetă. Programul va întreba numele fișierului și după salvare înregistrarea va putea fi verificată.

Programul va fi salvat împreună cu datele astfel încît va rula automat atunci cînd va fi încărcat.

4.3. PRELUCRARE DE TEXTE

4.3.1. Caractere cu accent

Tastatura calculatoarelor nu are taste speciale pentru caracterele cu accent din limba română. Dar, este posibil să se compună pînă la 21 de motive diverse, pentru calculatoarele HC-85 și TIM-S, care ar putea ține locul (pe ecran) a cîte unui caracter existent.

Pentru realizarea unor caractere noi trebuie să se înnegrească anumite căsuțe dintr-o grilă pătratică de 64 de căsuțe. Vom utiliza această proprietate a calculatoarelor HC-85 și TIM-S pentru a obține următoarele caractere : ă, î, ș, ț.

```
10 CLS
15 PRINT "Creare de caractere cu accent"
20 FOR i=1 TO 4
30 READ a$
40 FOR j=0 TO 7
50 READ y : POKE USR a$ + j, y
60 NEXT j
70 NEXT i
80 DATA "A", 56, 0, 56, 4, 60, 68, 60, 0
90 DATA "I", 32, 80, 0, 48, 16, 16, 56, 0
100 DATA "S", 0, 0, 56, 64, 56, 4, 56, 16
110 DATA "T", 0, 16, 16, 56, 16, 16, 28, 8
```

```
— memorarea POKE USR a$, 16
      POKE USR a$ + 1, 40
      POKE USR a$ + j, valoare
      POKE USR a$ + 7, 0
```

S-a ales pentru "i" tasta i ; acest caracter va fi deci obținut prin acționarea tastei I în modul grafic (G).

Utilizarea programului :

— Programul se lansează cu RUN, iar rularea lui nu va produce pe ecran nici un efect, dar valorile pentru caracterele cu accent au fost memorate. Astfel, dacă vrem să scriem pe ecran cuvântul „șanț“, vom proceda astfel :

- vom introduce comanda PRINT urmată de " ;
- apoi vom acționa tastele următoare :

```
GRAPHICS S GRAPHICS A N GRAPHICS T GRAPHICS ;
```

— se închid ghilimelele și se acționează ENTER.

Pe ecran va apare cuvântul „șanț“.

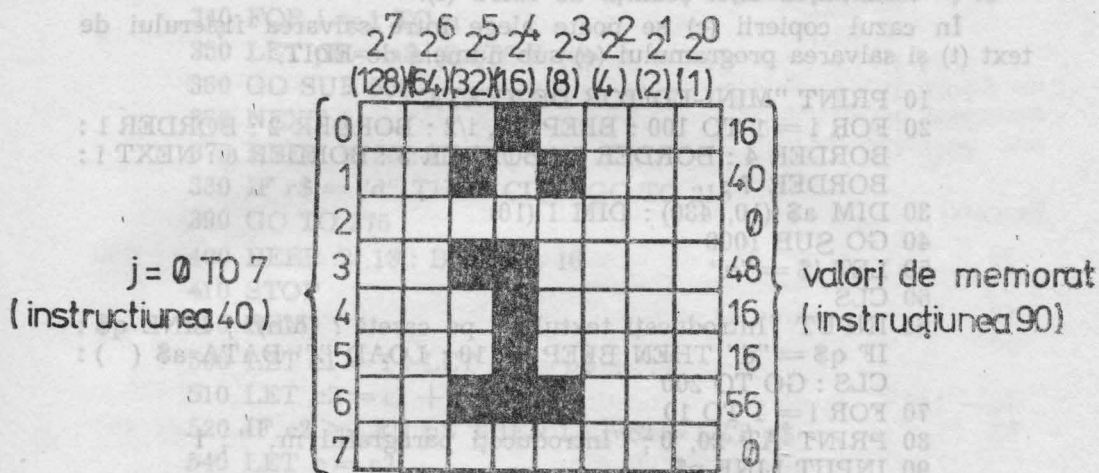
Observații :

— instrucțiunile 80—110 descrierea celor 4 caractere, câte o instrucțiune DATA pentru fiecare caracter, în ordinea anunțată ;

— instrucțiunile 40—60 memorarea fiecărei descrieri în 4 din cele 21 de grupe de memorie rezervate pentru diverse motive de definit.

Pentru ilustrare să luăm cazul caracterului "i"

DESCRIEREA GRILEI



5.3.2. Minieditor de texte

O aplicație importantă a calculatoarelor este aceea de prelucrare și editare a textelor, aceasta facilitând introducerea de texte în memoria calculatorului, modificarea ușoară a unor părți din text la dorința utilizatorului, salvarea textului pe un suport de memorie externă cu posibilitatea reincărcării și deci a modificării textului, imprimarea textului la imprimantă. Tastarea este mai ușoară decât la o mașină de scris obișnuită, nefiind necesară despărțirea cuvintelor la sfârșitul rîndului scris. Dacă ultimul cuvînt nu încapă pe rînd el va fi automat trecut pe rîndul următor, iar cuvintele din cadrul rîndului redistribuite.

Programul ales ca exemplu utilizabil pe un calculator HC-85 sau TIM-S are următorul obiectiv: introducerea unui text prin paragrafe, care vor putea fi astfel selecționate pentru a constitui textul final. Lungimea fiecărei linii este limitată la 32 de caractere, iar dacă ultimul cuvînt nu încapă pe rînd el va fi transferat pe rîndul următor. Fiecare paragraf poate ocupa pînă la 15 linii. Caracterele specifice limbii române ă, â, î, ș, ț, sînt disponibile prin intrarea în modul grafic (cs + 9) și acționarea tastei corespunzătoare (a pentru ă, i pentru î, b pentru â etc.). Pentru utilizarea programului minieditor de texte se va începe tastarea textului; toate tastele sînt permise (cs + 0 funcționînd pentru ștergerea caracterului din stînga cursorului) și în plus se poate merge cu cursorul la începutul sau la sfârșitul textului cu ajutorul săgeților ← și →. Două alte taste au un efect special: ENTER pentru terminarea unui paragraf și STEP pentru trecerea la editare. Meniul de editare oferă o alegere între opțiunile:

- vizualizarea pe ecran a unui anumit paragraf (tasta p);
- salvarea pe caseta magnetică a programului (c);

— inițializarea în vederea introducerii unui nou text (i);

— terminarea unei ședințe de lucru (s).

În cazul copierii (c) se poate alege între salvarea fișierului de text (t) și salvarea programului (e) sub numele de EDIT.

```
10 PRINT "MINI-EDITOR DE TEXTE"
20 FOR i=1 TO 100 : BEEP .01, i/2 : BORDER 2 : BORDER 1 :
  BORDER 4 : BORDER 3 : BORDER 5 : BORDER 6 : NEXT i :
  BORDER 7 :
30 DIM a$(10, 480) : DIM l(10)
40 GO SUB 1000
50 LET i$ = "n"
60 CLS
65 INPUT "Introduceți textul de pe casetă ? (d/n)"; LINE q$ :
  IF q$ = "d" THEN BEEP .5, 10 : LOAD "" DATA a$ ( ) :
  CLS : GO TO 200
70 FOR i=1 TO 10
80 PRINT AT 20, 0 ; "Introduceți paragraful nr. "; i
90 INPUT LINE p$
95 IF LEN p$ > 480 THEN LET p$ = p$ ( TO 480)
100 IF p$ = CHR$ 205 THEN GO TO 200
110 LET a$(i) = p$
120 LET l(i) = LEN p$
125 CLS
130 GO SUB 500
140 NEXT i
200 CLS : LET n = i - 1
210 BEEP .2, 20 : INPUT "Editare :
  — un paragraf o singură dată      p
  — stop                               s
  — copiere                            c
  — inițializare                       i      "; LINE r$
212 IF r$ = "i" THEN GO TO 65
215 IF r$ = "p" THEN GO TO 250
225 IF r$ = "s" THEN GO TO 400
227 IF r$ = "c" THEN GO TO 2000
230 GO TO 210
250 BEEP .2, 20 : INPUT "ce paragraf doriți ?"; i
255 IF i < 1 OR i > n THEN GO TO 250
260 LET p$ = a$(i) (TO l(i))
270 LET i$ = "n" : GO SUB 500
275 BEEP .2, 20 : INPUT „doriți să-l imprimați ?”; i$
277 CLS : GO SUB 500
280 BEEP .2, 20 : INPUT "un alt paragraf ?"; r$
290 IF r$ = "d" THEN CLS : GO TO 250
295 IF r$ = "n" THEN CLS : GO TO 210
300 GO TO 280
```



```

320 CLS
330 BEEP .2, 20 : INPUT "doriți să-l imprimați (d/n) ? " ; i$
340 FOR i = 1 TO n
350 LET p$ = a$ (i) (TO 1 (i))
360 GO SUB 500
370 NEXT i
375 BEEP .2, 20 : INPUT r$
380 IF r$ = "d" THEN CLS : GO TO 210
390 GO TO 375
400 BEEP .2, 13 : BEEP .2, 16
410 STOP
480 REM
500 LET c1 = 1 : LET p$ = p$ + " "
510 LET c2 = c1 + 32
520 IF c2 > LEN p$ THEN LET c2 = LET p$
540 LET c = c2
545 IF p$ (c) < > " " THEN LET c = c - 1 : GO TO 545
550 IF p$ (c) = " " THEN LET c2 = c + 1 : LET c = c - 1 :
    IF c > c1 THEN GO TO 550
560 PRINT p$ (c1 TO c)
565 IF i$ = "d" THEN LPRINT p$ (c1 TO c)
570 LET c1 = c2
580 IF c1 < LEN p$ THEN GO TO 510
600 RETURN
980 REM
1000 DIM a (5)
1010 DATA BIN 00100100, BIN 00011000, BIN 0, BIN 00111000,
    BIN 00000100, BIN 00111100, BIN 01000100 BIN 00111110
1011 DATA BIN 00011000, BIN 00100100, BIN 0, BIN 00111000,
    BIN 00000100, BIN 00111100, BIN 01000100, BIN 00111110
1012 DATA BIN 00001000, BIN 00010100, BIN 0, BIN 00011000,
    BIN 00001000, BIN 00001000, BIN 00001000, BIN 00011100
1013 DATA BIN 0, BIN 00010000, BIN 00111000, BIN 00010000,
    BIN 00010000, BIN 00010000, BIN 00010000, BIN 01001100
1014 DATA BIN 0, BIN 0, BIN 00011100, BIN 00100000, BIN
    00011100, BIN 00000010, BIN 00011100, BIN 01000000
1015 FOR x = 0 TO 7 : READ a : POKE USR "a" + x, a : NEXT x
1016 FOR x = 0 TO 7 : READ b : POKE USR "b" + x, b : NEXT x
1017 FOR x = 0 TO 7 : READ i : POKE USR "i" + x, i : NEXT x
1018 FOR x = 0 TO 7 : READ t : POKE USR "t" + x, t : NEXT x
1019 FOR x = 0 TO 7 : READ s : POKE USR "s" + x, s : NEXT x
1240 RETURN

```

```

2000 CLS
2010 INPUT AT 8, 4 ; "SALVARE :
      — text
      — EDIT " ; LINE r$
2020 IF r$ = "e" THEN SAVE "EDIT" LINE 0
2030 IF r$ = "t" THEN GO TO 3000
3000 INPUT "nume : " ; x$ ; SAVE x$ DATA a$ ( )
3010 GO TO 210

```

Remarci :

- 30 — rezervarea de memorie pentru 10 paragrafe a câte 480 de caractere fiecare
- 40 — crearea de caractere accentuate specifice
- 50 — indicatorul de tipărire la imprimantă după alegerea din linia 275 și 330 inițializat aici la "n"
- 70— 140 — buclă pentru constituirea paragrafelor
- 95 — un paragraf prea lung este trunchiat ; ce trece de 480 caractere este ignorat
- 100 — dacă în locul paragrafului se acționează tasta STEP se trece la Editare înainte de a se ajunge la paragraful 10
- 200—n — numărul de paragrafe create
- 210 — afișarea meniului editării
- 260 — transferul în p\$ a paragrafului ales
- 300 — dacă răspunsul nu este nici d nici n se repetă întrebarea
- 340— 370 — buclă de explorare sistematică a paragrafelor existente
- 390 — răspuns greșit (de exemplu o tastare involuntară)
- 500— 600 — subprogram de afișare și eventual de imprimare a unui paragraf fără ruperea cuvântului de la sfârșitul liniei
- 500— 510 — c1, c2 : poziția primului și ultimului caracter afișat ; se adaugă un spațiu la începutul paragrafului pentru evitarea unei excepții
- 520 — caz în care c2 punctează sfârșitul paragrafului
- 540— — indică ultimul caracter al liniei de afișat, fără ruperea cuvântului ; valoarea de pornire : c2
- 545 — deplasare spre stînga (c se diminuează) pînă cînd se găsește un spațiu : ultimul cuvînt se îndepărtează dacă nu încapă pe linie
- 550 — deplasare spre stînga pînă cînd se găsește altceva decît un spațiu
- 560— 565 — afișare și eventual imprimarea unei linii
- 570 — se reîncepe de la linia 510 atît cît nu a fost atins sfârșitul programului
- 1000—1240 — subprogram de creare de caractere accentuate
- 1000—1014 — descrierea celor 5 caractere, o instrucțiune pentru fiecare caracter
- 1014—1019 — definirea apelării și memorarea acestor caractere.

4.4. GRAFICĂ ȘI MUZICĂ

4.4.1. Floare muzicală

Un program pe calculatoarele HC-85 și TIM-S care prezintă un amestec de imagini și sunete. O floare ale cărei petale se desenează progresiv în sunete din ce în ce mai ascuțite, apoi petalele cad una câte una, iar sunetele recoboară.

```
10 BORDER 5 : PAPER 7 : INK 0 : CLS
20 PRINT "FLOARE MUZICALĂ"
30 LET TON = 0 : LET TON1 = 1
50 INK 4 : PLOT 125, 0 : DRAW 0, 70
55 GO SUB 500
65 INK 3
70 FOR R = 1 TO 10
80 CIRCLE 15, 80, R
90 GO SUB 500
100 NEXT R
120 LET P = 0 : GO SUB 200
130 LET H = 50 : GO SUB 400
140 LET P = 1 : LET TON1 = - 1 : GO SUB 200
155 INK 0
160 STOP
180 REM
200 CIRCLE INVERSE P ; 125, 101, 10
210 GO SUB 500
220 CIRCLE INVERSE P ; 107, 90, 10
230 GO SUB 500
240 CIRCLE INVERSE P ; 143, 90, 10
250 GO SUB 500
290 CIRCLE INVERSE P ; 107, 70, 10
300 GO SUB 500
310 RETURN
380 REM
400 FOR M = 10 TO 20 STEP 10
405 PLOT 125, H
410 DRAW M, M, 2
415 GO SUB 500
420 DRAW - M, - M, 2
425 GO SUB 500
430 DRAW - M, M, 2
435 GO SUB 500
440 DRAW M, - M, 2
442 GO SUB 500
445 LET H = H - 40
450 NEXT M
460 RETURN
```

480 REM
500 LET TON1 = TON + TON1
510 BEEP .2, TON
520 RETURN

4.4.2. Curcubeu

Programul care rulează pe calculatoare HC-85 și TIM-S va face să apară o serie de benzi verticale ale căror culori sînt cele ale curcubeului : violet, albastru închis, albastru deschis, verde, galben, portocaliu și roșu. Violetul și portocaliul care nu sînt disponibile se vor obține prin amestecarea a două culori care sînt disponibile prin definirea unui motiv reprezentînd o grilă în care din două pătrățele alăturate, una reprezintă culoarea cernelii, și cealaltă a hîrtiei.

Violetul se va obține indicînd pentru hîrtie culoarea albastru închis și pentru cerneală roșu, iar pentru portocaliu se va indica pentru hîrtie culoarea galben, iar pentru cerneală culoarea roșie.

```
10 BORDER 5 : PAPER 7
12 PRINT "Curcubeu"
20 DATA 85, 170, 85, 170, 85, 170, 85, 170
30 FOR p=0 TO 7 : READ g : POKE USR CHR$ 71 + p, g :
  NEXT p
50 DATA 1, 2, 1, 1, 5, 5, 4, 4, 6, 6, 6, 2, 2, 2
70 FOR c=2 TO 29 STEP 4
80 READ hîrtie
90 PAPER hîrtie
100 READ culoare
110 INK culoare
120 FOR l=2 TO 20
130 PRINT AT l, c ; "GGGG"
140 NEXT l
150 NEXT c
170 INK 0
180 PAPER 7
190 BEEP .2, 13 : BEEP .2, 16
200 STOP
```

Observație : instrucțiunea 30 — crearea de caractere utilizator GRAPHICS G. Linia 130 afișează 4 caractere grafice diferite (tasta G în modul grafic).

4.4.3. 3D — Grafică tridimensională

Programul 3D realizat pentru calculatoare HC-85 sau TIM-S permite desenarea figurilor în spațiu, vizualizarea lor din orice unghi, modificarea mărimii lor și chiar adăugare de perspectivă la desen.

Fiecare desen-figură se construiește din linii drepte. Coordonatele x, y, și z (pentru fiecare se pot introduce pînă la 400 de linii) sînt

memorate într-o matrice care poate fi salvată sau încărcată de pe caseta magnetică.

Programul este împărțit în module (opțiuni) care pot fi selectate dintr-o secțiune de meniu principal.

La rulare se afișază următorul meniu :

Apasă o tastă :

- D — afișare date ;
- I — introducere date ;
- L — încărcare date ;
- P — imprimare date ;
- Q — sfârșit ;
- S — salvare date ;
- V — vizualizare.

La opțiunea de afișare date (D) se va întreba de la care linie se dorește afișarea datelor ; liniile utilizate pentru desenare sînt numerotate de la 1 la 400. Se vor afișa apoi coordonatele x, y, și z pentru fiecare sfârșit a 17 linii consecutive, apoi se va întreba dacă se mai dorește afișarea altor date sau reîntoarcerea la meniul principal.

Opțiunea de introducere date va permite adăugarea de noi date sau modificarea valorilor existente.

Pentru fiecare linie de desen vor trebui introduse :

- numărul de linie (un întreg cuprins între 1 și 400) ;
- coordonatele x, y, z pentru fiecare sfârșit și început de linie.

Fiecare din coordonatele de valori trebuie să fie un întreg cuprins între -100 și 100. Cele 7 numere vor fi introduse pe un rînd (linie), valorile fiind separate prin spații. Dacă una din valori nu este validă, programul va ignora întreaga linie și va aștepta introducerea unei alte versiuni pentru linia respectivă.

Dacă toate valorile sînt acceptate, atunci ele vor fi afișate pe ecran, apoi cursorul va reapare în partea de jos a ecranului așteptînd introducerea unei alte linii. Se poate continua astfel introducerea liniilor în acest fel sau se poate realiza întoarcerea la meniul principal tastînd <CR> (ENTER).

Liniile se pot introduce în orice ordine, iar vechile valori se modifică prin introducerea unei noi linii cu același număr de linie.

Opțiunea de introducere date (I) va permite încărcarea unui fișier de date pe bandă și ștergerea oricărei date memorate anterior. Odată ce fișierul de date a fost încărcat, se poate alege verificarea, iar dacă în acest proces programul se oprește, acesta se va putea retasta cu comanda GO TO 1000.

Opțiunea de imprimare date (P) va realiza o copie a ecranului la imprimantă. Se va întreba primul și ultimul număr de linie al blocului de date care se va dori să se tipărească. Se vor introduce cele două numere de linie separate prin spațiu.

Opțiunea de sfârșit (Q) va opri execuția programului. Pentru restartare se va utiliza GO TO 1000 pentru păstrarea datelor sau RUN pentru ștergerea lor.

Opțiunea de salvare date (S) va produce salvarea datelor sub numele fișierului de date. Fiind vorba de calculatoare HC-85 și TIM-S

numele fișierului nu trebuie să aibă mai mult de 10 caractere lungime. Când datele au fost salvate, programul va cere verificarea înregistrării. La fel ca la încărcare, dacă programul se va opri, atunci se va putea restarta cu GO TO 1000 pentru păstrarea datelor.

Pentru salvarea programului, acesta va trebui oprit (cu opțiunea de sfârșit — Q, sau cu CAPS/SHIFT și BREAK). Apoi se va introduce CLEAR înainte de salvarea programului pentru a șterge fișierul de date și alte variabile. În acest fel, timpul de salvare se va reduce substanțial.

Cu opțiunea vizualizare (V) se va putea vedea pe ecran desenul realizat. Vor trebui introduse :

— primul și ultimul număr al liniilor blocului de linii, care reprezintă datele figurii ce se va desena ;

— un factor de scalare care trebuie să fie un întreg între 1 și 999. Acesta va determina mărimea figurii (desenului). Se poate începe de exemplu : cu valoarea 50 ;

— unghiul de rotire orizontală, în grade. Acesta trebuie să fie un întreg cuprins între 0 și 360 ;

— unghiul de rotire verticală, în grade (tot întreg cuprins între 0 și 360) ;

— o valoare a „adâncimii“ care va determina gradul „perspectivei“ utilizate. Aceasta va fi un întreg cuprins între 0 și 9, valorile 3 și 4 dând de obicei cel mai bun efect.

Toate valorile trebuie introduse într-o linie și separate prin spații. Dacă o valoare nu este validă, programul va ignora întreaga linie și va aștepta introducerea ei încă o dată.

Programul va construi figura (desenul) pe baza datelor din blocul de linii specificat. Aceasta va permite memorarea datelor într-un fișier care va servi la desenarea mai multor figuri sau la desenarea numai a unei părți dintr-o figură mai mare.

Dacă rezultatul desenului nu va intra în ecran, programul se va opri ; pentru repornirea lui se va utiliza GO TO 1000.

Figura desenată va putea fi copiată la imprimantă.

100 LET ml = 400

110 DIM p\$ (ml, 6) : DIM z\$ (6)

120 LET t\$ = " linia x1 y1 z1 x2 y2 z2"

1000 REM Meniu principal

1010 CLS : PRINT AT 5, 7 ; "Apasă o tastă ; "

1020 PRINT TAB 8 ; "D — afișare date"

1030 PRINT TAB 8 ; "I — introducere date"

1040 PRINT TAB 8 ; "L — încărcare date"

1050 PRINT TAB 8 ; "P — listare date"

1060 PRINT TAB 8 ; "Q — ieșire"

1070 PRINT TAB 8 ; "S — salvare date"

1080 PRINT TAB 8 ; "V — vizualizare"

1100 LET k\$ = FN a\$ (INKEY\$)

1110 IF k\$ = "D" THEN GO TO 2000

1120 IF k\$ = "I" THEN GO TO 3000

```

1130 IF k$ = "L" THEN GO TO 4000
1140 IF k$ = "P" THEN GO TO 5000
1150 IF k$ = "S" THEN GO TO 6000
1160 IF k$ = "V" THEN GO TO 7000
1170 IF k$ <> "Q" THEN GO TO 1100
1200 CLS : PRINT AT 5, 0 ; FLASH 1 ; "Acum programul se va
opri"
1210 PRINT ' ' "Pentru restartare, introduceți ;"
1220 PRINT AT 10, 5 ; "RUN (șterge datele)"
1230 PRINT "sau GO TO 1000" ; TAB 5 ; "(păstrează datele)"
1240 STOP
2000 REM afișare date
2010 CLS : PRINT AT 10, 5 ; "De la linia"
2020 GO SUB 8100 : LET l = i : IF l < 1 OR l > ml THEN GO
TO 2020
2100 CLS : PRINT t$
2120 FOR a = 1 TO l + 16 : IF a > ml THEN GO TO 2200
2125 PRINT 'a ; : IF p$ (a) = z$ THEN GO TO 2140
2130 FOR b = 1 TO 6 : PRINT TAB 1 + 4 * b ; CODE p$ (a, b)
— 133 ; : NEXT b
2140 NEXT a
2200 PRINT ' ' "Încă (D/N) ?"
2210 LET k$ = FN a$ (INKEY$)
2220 IF k$ <> "N" AND k$ <> "D" THEN GO TO 2210
2230 IF k$ = "N" OR a > ml THEN GO TO 1000
2240 LET l = a : GO TO 2100
3000 REM Introducere date
3010 CLS : PRINT "Introduceți numărul de linie urmat de coordo-
natele x, y, z ale primului punct, apoi x, y, z, ale celui de
al doilea punct. Toate coordonatele vor fi separate prin spații
3020 PRINT ' "Sau, tasteți ENTER pentru întoarcerea la meniul
principal" ' ' t$
3100 INPUT LINE i$ : IF i$ = " " THEN GO TO 1000
3110 GO SUB 8000 : LET l = i : IF l < 1 OR l > ml THEN GO
TO 3100
3120 FOR a = 1 TO 6
3130 GO SUB 8000 : IF ABS i > 100 THEN GO TO 3100
3140 LET p$ (l, a) = CHR$ (133 + i)
3150 NEXT a
3160 POKE 23692, 0 : PRINT 'l ;
3170 FOR a = 1 TO 6 : PRINT TAB 2 + 4 * a ; CODE p$ (l, a)
— 133 ; : NEXT a
3180 GO TO 3100
4000 REM încărcare date de pe bandă
4010 CLS : PRINT AT 5, 0 ; "Numele fișierului de date ?" ; CHR$8
4020 INPUT LINE i$ : PRINT i$
4030 LOAD i$ DATA p$ ( )
4040 INPUT "Verificare (D/N) ?" ; LINE a$

```

```

4050 IF a$ = "N" OR a$ = "n" THEN GO TO 1000
4060 IF a$ < > "D" AND a$ < > "d" THEN GO TO 4040
4070 PRINT " " "Start cas pentru verificare"
4080 VERIFY i$ DATA p$ ( )
4090 GO TO 1000
5000 REM listare date
5010 GO SUB 8200
5020 LPRINT 't$
5030 FOR a = 11 TO 12 : LPRINT 'a ; : IF p$ (a) = z$ THEN GO
    TO 5050
5040 FOR b = 1 TO 6 : LPRINT TAB 2 + 4 * b ; CODE p$ (a, b)
    - 133 ; : NEXT b
5050 NEXT a
5060 LPRINT " "
5080 GO TO 1000
6000 REM salvare date
6010 CLS : PRINT AT 5, 0 ; "Numele fişierului de date, " ; CHR$8;
6020 INPUT LINE i$ : IF i$ = " " THEN GO TO 6020
6030 IF LEN i$ > 10 THEN LET i$ = i$ (TO 10)
6040 PRINT i$
6050 SAVE i$ DATA p$ ( )
6060 PRINT " " "Rebobinează pentru verificare"
6070 PRINT " " "Dacă verificarea nu merge programul se va opri
    " " "Utilizați GO TO 1000 pentru startare program. " " "nu
    RUN"
6080 VERIFY i$ DATA p$ ( )
6090 GO TO 1000
7000 REM Vizualizare
7010 CLS : PRINT AT 5, 0 ; "Introduceți ;" "
7020 PRINT TAB 4 ; "Numărul primei linii" ; TAB 4 ; " și a ulti-
    mei linii"
7030 PRINT TAB 4 ; "Factorul de scalare (1—999)"
7040 PRINT TAB 4 ; "Rotație orizontală (0—360)"
7050 PRINT TAB 4 ; "Rotație verticală (0—360)"
7060 PRINT TAB 4 ; "Adâncimea (0—9)"
7070 PRINT " " "Separate prin spații"
7100 GO SUB 8100 : LET 11 = i : IF 11 < 1 OR 11 > ml THEN
    TO 7100
7110 GO SUB 8000 : LET 12 = i : IF 12 < 11 OR 12 > ml THEN
    GO TO 7100
7120 GO SUB 8000 : LET s = 1/100 : IF s < .01 OR s > 9.9 THEN
    GO TO 7100
7130 GO SUB 8000 : LET hr = i : IF hr < 0 OR hr > 360 THEN
    GO TO 7100
7140 GO SUB 8000 : LET vr = i : IF vr < 0 OR vr > 360 THEN
    GO TO 7100
7150 GO SUB 8000 : LET d = i : IF d < 0 OR d > 9 THEN GO
    TO 7100
7200 CLS
7210 LET ch = COS (hr * PI/180) : LET sh = SIN (hr * PI/180)

```

```

7220 LET cv = COS (vr * PI/180) : LET sv = SIN (vr * PI/180)
7300 FOR l=11 TO 12 : IF p$ (l) = z$ THEN GO TO 7400
7310 LET x1 = FN b (1) : LET y1 = FN b (2) : LET z1 = FN
      b (3)
7320 LET x2 = FN b (4) : LET y2 = FN b (5) : LET z2 = FN b (6)
7330 LET xx1 = x1 * ch - y1 * sh : LET xx2 = x2 * ch -
      y2 * sh
7340 LET yy1 = y1 * ch + x1 * sh : LET yy2 = y2 * ch + x2
      * sh
7350 LET yy1 = yy1 * cv - z1 * sv : LET yy2 = yy2 * cv - z2
      * sv
7360 LET zz1 = z1 * cv + yy1 * sv : LET zz2 = z2 * cv + yy2
7370 LET p = i + d * zz1/1000 : LET xx1 = xx1 * p : LET yy1
      = yy1 * p
7380 LET p = 1 + d * zz2/1000 : LET xx2 = xx2 * p : LET yy2
      = yy2 * p
7390 PLOT xx1 + 127, yy1 + 87 : DRAW xx2 - xx1, yy2 - yy1
7400 NEXT l
7410 INPUT "Imprimare (D/N) ? " ; LINE i$
7420 IF i$ = "d" OR i$ = "D" THEN COPY : GO TO 1000
7430 IF i$ = "n" OR i$ = "N" THEN GO TO 1000
7440 GO TO 7410
8000 REM Conversia primului număr în i$
8010 LET sgn = 1
8020 LET i = 0 : IF i$ = "" OR i$ = "." THEN RETURN
8030 IF i$ (1) < > " - " AND (i$ (1) < "0" OR i$ (1) > "9")
      THEN LET i$ = i$ (2 TO) : GO TO 8020
8040 IF i$ (1) = " - " THEN LET sgn = 1 : LET i$ = i$ (2 TO) :
      GO TO 8020
8050 FOR c = 1 TO LEN i$ : LET c$ = i$ (c)
8060 IF c$ < "0" OR c$ > "9" THEN GO TO 8030
8070 NEXT c
8080 LET i = sgn X VAL i$ (TO c - 1) : LET i$ = i$ (c TO)
8090 RETURN
8100 REM introducere i$
8110 INPUT LINE i$ : GO SUB 8000
8200 REM prima și ultima linie
8210 CLS : PRINT AT 10, 0 ; "Introduceți numărul primei și ulti-
      mei linii separate prin spațiu ;"
8220 INPUT LINE i$ : GO SUB 8000 : LET l1 = i
8230 IF l1 < 1 OR l1 > m1 THEN GO TO 8200
8240 GO SUB 8000 : LET l2 = i
8250 IF l2 < 1 OR l2 > m1 THEN GO TO 8200
8260 RETURN
9000 DEF FN a$ (a$) = CHR$ (CODE a$ - (32 AND CODE a$
      > 96))
9100 DEF FN b (z) = (CODE p$ (l, z) - 133) * s

```


Remarci :

- 1000—1170 — afișarea meniului principal și alegerea opțiunii utilizatorului, apoi salt la acea opțiune ;
- 2000—2240 — secțiunea de afișare date ;
- 2010—2100 — introducerea numărului primei linii, verificarea lui, apoi ștergerea ecranului și afișarea liniei de început t\$;
- 2110—2140 — buclă executată de 17 ori pentru a afișa 17 linii ;
- 2120—2130 — afișarea numărului de linie și apoi afișarea celor 6 articole de date (dacă există) ;
- 2200—2240 — utilizatorul este întrebat dacă dorește să vadă mai multe linii sau să se întoarcă la meniul principal ;
- 3000—3180 — modul de introducere date ;
- 3010—3020 — afișarea instrucțiunilor și a liniei de început t\$;
- 3100—3180 — buclă executată odată pentru fiecare linie ;
- 3100 — introducerea liniei șir de către utilizator, întoarcerea în meniul principal dacă șirul este vid ;
- 3110 — separarea numărului de linie din șir și verificarea lui ;
- 3120—3150 — buclă pentru separarea celor 6 coordonate din șir ;
- 3160—3180 — afișarea numărului de linie și cele 6 coordonate introduse, apoi introducerea următoarei linii ;
- 4000—4090 — modul de încărcare date. Introducerea numelui fișierului, afișarea lui (CHR\$8 din linia 4010 mută cursorul înapoi astfel încât atunci când numele fișierului este afișat se va reafișa simbolul " ? "). Apoi încărcarea fișierului de date p\$ (), verificarea lui la cererea utilizatorului apoi, înapoi la meniul principal ;
- 5000—5080 — modul de imprimare date ;
- 5010—5020 — introducerea primului și ultimului număr de linie (11 și 12) care vor fi afișate. Afișarea liniei de început t\$;
- 5030—5050 — buclă executată pentru afișarea liniilor de la 11 la 12, evitarea imprimării coordonatelor dacă nu sînt date în linie ;
- 5040 — buclă pentru imprimarea celor 6 coordonate dintr-o linie ;
- 6000 — modul de salvare date ;
- 6010—6040 — introducerea numelui fișierului fără acceptarea unui șir vid și păstrarea a maximum 10 caractere din nume (dacă este mai mare de 10 caractere). Afișarea numelui ;
- 6050—6090 — salvarea fișierului de date p\$ () pe bandă și verificarea lui ;
- 7000—7440 — modul de vizualizare ;
- 7010—7150 — introducerea primului și ultimului număr de linie a factorilor de scalare, de rotație și de adîncime, apoi verificarea lor ;
- 7210—7220 — calcularea unor valori utile ;
- 7300—7400 — buclă executată odată pentru fiecare linie de desenat ;
- 7310—7320 — extragerea celor 6 coordonate ale liniei din P\$ () ;
- 7330—7340 — calculul noilor coordonate pentru rotația orizontală ;

- 7370—7380 — calculul pentru obținerea efectului de adâncime ;
 7390 — desen al liniei pe ecran ;
 7410—7440 — imprimarea unei copii pe ecran dacă utilizatorul dorește sau intoarcerea la meniul principal ;
 8000—8090 — subrutină de căutare a unui număr întreg în i\$, atribuirea valorii găsite lui i. Dacă nici un număr nu se găsește în i\$ atunci i va lua valoarea 2. Apeluri succesive ale acestei subrutine vor produce extragerea valorilor succesive din i\$;
 8100 — subrutină de introducere a șirului i\$ de către utilizator și apoi introducerea primei valori utilizând subrutina de la linia 8000.
 8200 — subrutină de introducere a unei linii care să conțină două numere de linie, apoi extragerea numerelor (11 și 12) și verificarea validității lor ;
 9000 — definirea unei funcții utilizator care să dea litere mari pentru caracterele care au fost introduse sau cu litere mici sau cu litere mari ;
 9100 — definirea unei funcții utilizator pentru extragerea valorii coordonatei z (z de la 1 la 6) din linia l a fișierului de date.

Datele sînt memorate într-o matrice șir p\$ (400, 6), un caracter fiind utilizat pentru a memora fiecare coordonată după formula p\$ (L, A) = CHR\$ (valoare coord. + 133).

Aceasta va permite programului să memoreze valori între -100 și +100 fără nici o valoare care să semene cu un caracter spațiu (codul 32), care este utilizat cu semnificația de "nici o dată".

Conform liniilor 100 și 110 se pot memora maximum 400 de linii (ml linia maximă). Acest număr se poate extinde pînă la valoarea 5800.

4.4. CREION MAGIC

Programul permite trasarea figurilor compuse din drepte orizontale, verticale și înclinate cu unghiuri de 45 de grade, existînd și posibilitatea ridicării și coborîrii „creionului“.

Deplasarea este coordonată de tastele :

- | | |
|-------------|----------------------|
| 5 — stînga | 5 + SS — stînga jos |
| 6 — sus | 6 + SS — stînga sus |
| 7 — jos | 7 + SS — dreapta sus |
| 8 — dreapta | 8 + SS — dreapta jos |

0 — ridicare „creion“

1 — coborîre „creion“

După acționarea tastelor de deplasare nu este necesară și acționarea tastei — CR —, folosindu-se funcția INKEY\$ (instrucțiunea 80).

Pentru poziționarea spotului se utilizează PLOT (instrucțiunea 60) iar pentru deplasare DRAW și PLOT.

```

5 BORDER 6 : PAPER 7 : INK 0
10 PRINT "CREION MAGIC"
20 LET a = 0
30 LET x = 125
40 LET y = 87
60 PLOT x, y
80 LET C$ = INKEY$
90 IF C$ = " " THEN GO TO 80
100 IF C$ = "0" THEN LET a = 0 : GO TO 80
110 IF C$ = "1" THEN LET a = 1 : PLOT x, y : GO TO 80
115 LET dx = 0 : LET dy = 0
120 IF C$ = "5" THEN LET dx = -1 : GO TO 250
130 IF C$ = "6" THEN LET dy = -1 : GO TO 250
140 IF C$ = "7" THEN LET dy = 1 : GO TO 250
150 IF C$ = "8" THEN LET dx = 1 : GO TO 250
160 IF C$ = CHR$ 8 THEN LET dx = -1 : LET dy = 1 : GO
    TO 250
170 IF C$ = CHR$ 10 THEN LET dx = -1 : LET dy = -1 :
    GO TO 250
180 IF C$ = CHR$ 11 THEN LET dx = 1 : LET dy = 1 : GO
    TO 250
190 IF C$ = CHR$ 9 THEN LET dx = 1 : LET dy = -1 : GO
    TO 250
200 GO TO 80
250 IF a = 0 THEN PLOT INVERSE 1 ; x, y
260 LET x = x + dx
265 IF x > 255 OR x < 0 THEN BEEP .2, 10 : BEEP .2, 5 :
    LET x = x - dx
270 LET y = y + dy
275 IF y > 175 OR y < 0 THEN BEEP .2, 10 : BEEP .2, 5 : LET
    y = y - dy
280 PLOT x, y
290 GO TO 80

```

Remarci :

- instrucțiunea 20 — lăsarea creionului ;
- instrucțiunile 30 și 40 — se poziționează spotul în centrul ecranului, respectându-se organizarea ecranului la HC-85 și TIM-S, adică punctul (0, 0) în colțul din stînga jos și (255, 175) colțul din dreapta sus ;
- instrucțiunea 120—150 — pregătesc deplasarea creionului în funcție de tasta acționată ;
- instrucțiunea 160—190 — deplasare pe diagonală.

4.4.5. Coliziune

Este un joc de îndeminare și reflexe exemplificat pe un calculator HC-85 sau TIM-S.

Un vehicul este condus pe o pistă (din cele 4 care apar desenate pe ecran), dar din sens contrar apar alte vehicule care trebuiesc evitate prin trecerea pe altă pistă. Cu tasta "I" se va muta vehiculul pe o pistă din interior, iar cu tasta "O" se va muta pe o pistă spre exterior. Rezultatul final va indica numărul de circuitate care au fost realizate comparativ cu scorul cel mai mare realizat.

```

100 GO SUB 9000 : REM inițializare pentru prima cursă
200 IF yy = 21 - yt THEN LET nyx = yx + 1 : IF yx = 30 - yt
    THEN LET nyy = yy - 1 : LET y$ = CHR$ 145
210 IF yy = yt THEN LET nyx = 1 : IF yx = yt + 1 THEN LET
    nyy = yyt + 1 : LET y$ = CHR$ 145
220 IF yx = yt THEN LET nyy = yy + 1 : IF yy = 20 - yt THEN
    LET nyx = yx + 1 : LET y$ = CHR$ 144
230 IF yx = 31 - yt THEN LET nyy = yy - 1 : IF yy = yt + 1
    THEN LET nyx = yx - 1 : LET y$ = CHR$ 144
240 IF nyx = 15 + (yy > 12) OR nyy = 11 THEN LET yf = 2
    * ((yt < 7 AND INKEY$ = "I") - (yt > 1 AND INKEY$ =
    "O"))
250 IF yf < > 0 THEN LET nyx = nyx + ((nyx = yt) - (nyx =
    31 - yt)) * SGN yf : LET nyy = nyy + ((nyy = yt) - (nyy
    = 21 - yt)) * SGN yf : LET yt = yt + SGN yf : LET yf = yf
    - SGN yf
260 IF nyy = cy AND nyx = cx THEN GO TO 400
300 IF cy = 21 - ct THEN ncx = cx - 1 : IF cx = ct + 1 THEN
    LET ncy = ncy - 1 : LET c$ = CHR$ 145
320 IF cx = ct THEN LET ncy = cy - 1 : IF cy = ct + 1 THEN
    LET ncx = cx + 1 : LET c$ = CHR$ 144
330 IF cx = 31 - ct THEN LET ncy = cy + 1 : IF cy = 20 - ct
    THEN LET ncx = cx - 1 : LET c$ = CHR$ 144
340 IF (ncx = 15 + (ncy > 8)) OR ncy = 11 THEN IF RND < s/10
    THEN LET cf = 2 * SGN (yt - ct)
350 IF cf < > 0 THEN LET ncx = ncx + ((ncx = ct) - (ncx = 31 -
    ct)) * SGN cf : LET ncy = ncy + ((ncy = ct) - (ncy = 21 -
    ct)) * SGN cf : LET ct = ct + SGN cf : LET cf = cf - SGN cf
400 PRINT AT yy, yx ; ". " ; AT cy, cx ; ". " ; AT nyy, nyx ;
    INK 1 ; y$ ; AT ncy, ncx ; INK 2 ; c$
410 LET yx = nyx : LET yy = nyy : LET cx = ncx : LET cy =
    ncy
420 IF yx = 16 AND yy < 8 THEN LET lap = lap + 1 : PRINT
    AT 9, 12 ; lap ; "LAP" ; ("s" AND lap > 1)
430 IF yx < > cx OR yy < > cy THEN GO TO 200
500 FOR a = 1 TO 6 : FOR b = 144 TO 145 : BEEP .03, -40 :
    PRINT INK a ; AT yy, yx ; CHR$ b : NEXT b : NEXT a

```

```

510 PRINT AT yy, yx ; CHR$ 146
520 IF lap > hi THEN LET hi = lap : PRINT AT 11, 12 ; "SCOR
    MAX" ; AT 12, 15 ; hi
530 INPUT "Tastați ENTER pentru altă cursă" ; LINE i$
540 PRINT AT yy, yx ; ". " : GO SUB 9200 : GO TO 200
9000 REM desenare piste
9010 INK 0 : PAPER 7 : FLASH 0 : BRIGHT 0 : OVER 0 : IN-
    VERSE 0 : BORDER 7 : CLS
9020 FOR a = 32 TO 160 STEP 32 : PLOT a/2, a/2 - 13
9030 DRAW 256 - a, 0 : DRAW 12, 12, PI/2
9040 DRAW 0, 176 - a : DRAW - 12, 12, PI/2
9050 DRAW a - 256, 0 : DRAW - 12, - 12, PI/2
9060 DRAW 0, a - 176 : DRAW 12, - 12, PI/2
9070 NEXT a
9080 FOR a = 2 TO 6 : PRINT AT a, 15 ; ". . ." ; AT a + 13, 15 ;
    ". . ." : NEXT a
9090 FOR a = 10 TO 12 : PRINT AT a, 12 ; "....." ; AT a, 25 ;
    "....." : NEXT a
9100 REM vehicule
9110 DATA 231, 66, 255, 255, 255, 255, 66, 231
9120 DATA 189, 255, 189, 60, 60, 189, 255, 189
9130 DATA 36, 90, 189, 126, 126, 189, 90, 36
9140 RESTORE 9100
9150 FOR a = 0 TO 23 : READ b : POKE USR "a" + a, b : NEXT a
9160 LET hi = 0
9200 REM valori inițiale
9210 LET yx = 5 : LET yy = 1 : LET yt = 1 : LET nyx = yx :
    LET nyy = yy : LET y$ = CHR$ 144 : LET yf = 0
9220 LET cx = 16 : LET cy = 7 : LET ct = 7 : LET ncx = cx : LET
    ncy = cy : LET c$ = CHR$ 144 : LET cf = 0
9230 LET lap = 0 : PRINT AT 9, 12 ; "....."
9240 INPUT "Nivel (1 - 9) ?" ; s
9999 RETURN

```

- 100 — chemare subrutine desen piste și inițializarea caracterelor
definite de utilizator
- 200—430 — bucla principală de program ; mișcă ambele vehicule cu
un bit
- 200—230 — calculul poziției următoare pentru vehiculul condus
- 240 — se iau în considerare tastele "I" și "O"
- 250 — mută o jumătate de drum de la o pistă la alta dacă yf
<> 0
- 260 — terminare dacă vehiculele s-au ciocnit ;
- 300—330 — calculul următoarei poziții pentru vehiculul oponent
- 340 — inițializarea variabilei de mișcare "cf" dacă vehiculul
oponent decide schimbarea pistei

- 350 — mută vehiculul oponent 1/2 din drum spre următoarea pistă dacă $\langle \rangle 0$
- 400—410 — șterge vechea imagine și amîndouă vehiculele
- 430 — întoarcere la linia 200 dacă vehiculul oponent nu a reușit să lovească vehiculul condus de utilizator
- 500—540 — rutină de sfîrșit ; calculează cel mai mare scor (hi), dacă ultimul a fost îmbunătățit și propune o altă cursă
- 9000 — subrutină pentru inițializarea cursei principale
- 9200 — subrutină pentru inițializarea poziției de start pentru altă cursă

Variabile folosite :

- yx, yv — coordonatele x și y pentru vehiculul condus
- yt — numărul pistei vehiculului condus ; pista 1 este în afară, iar pistele sînt numerotate 1, 3, 5, 7
- cx, cy, ct — coordonatele și numărul pistei pentru vehiculul oponent
- nyx, nyy — valorile yx și yy următoare
- ncx, ncy — valorile cx și cy următoare
- yf, cf — variabile inițializate cu $+2$ sau -2 cînd vehiculul este pe punctul de a schimba pista

2. Obiective urmările

A. OBIECTIVE GENERALE

- Cunoașterea, înțelegerea și însușirea prevederilor documentelor de partid cu privire la revoluția tehnico-științifică în informatică, robotizării și automatizării proceselor de producție.
- Orientarea școlară și profesională în strînsă corelare cu cerințele actuale și de perspectivă ale economiei naționale și ale revoluției tehnico-științifice contemporane.
- Organizarea unor acțiuni politico-educative de masă, metodice și de instruire în scopul popularizării acestei activități în rândul copiilor, tineretului și al cadrelor didactice.
- Colaborarea cu alte cercuri științifice și tehnico-aplicative în vederea realizării unor lucrări de creație tehnico-științifică, a unor produse-program cu o largă aplicabilitate în diferite domenii de activitate și care să încorporeze un grad din ce în ce mai ridicat de înțelegință umană.
- 44 — Însușirea învățămîntului de practică (producție) și cercetare
- Formarea concepției științifice, materialistă-dialectice despre lume și viață la elevi.

PLANIFICAREA ACTIVITĂȚII ÎN CEROURILE DE INFORMATICĂ

PROGRAM TEMATIC ANUAL

al cercului de informatică pentru grupe de începători

1. Organizarea cercului

2. Obiective urmărite

A. OBIECTIVE GENERALE

— Cunoașterea, înțelegerea și însușirea prevederilor documentelor de partid cu privire la revoluția tehnico-științifică, la informatizarea, automatizarea și automatizarea proceselor de producție.

— Educarea prin muncă și pentru muncă a pionierilor și școlărilor.

— Orientarea școlară și profesională în strânsă corelare cu cerințele actuale și de perspectivă ale economiei naționale și ale revoluției tehnico-științifice contemporane.

— Organizarea unor acțiuni politico-educative de masă, metodice și de instruire, în scopul popularizării acestei activități în rândul copiilor, părinților și al cadrelor didactice.

— Colaborarea cu alte cercuri științifice și tehnico-aplicative, în vederea realizării unor lucrări de creație tehnico-științifică, a unor produse-program cu o largă aplicabilitate în diferite domenii de activitate și care să încorporeze un grad din ce în ce mai ridicat de inteligență umană.

— Legarea învățămîntului de practică (producție) și cercetare.

Formarea concepției științifice, materialist-dialectice despre lume și viață.

B. OBIECTIVE OPERAȚIONALE SPECIFICE

a) Obiective cognitive

- Dobîndirea de către pionieri a unui sistem minimal de cunoștințe științifice despre calculatoare electronice.
- Însușirea unor elemente de programare a calculatoarelor personale.
- Învățarea unor limbaje de programare: BASIC, LOGO etc.
- Consolidarea unor cunoștințe dobîndite în procesul de învățămînt (în special la disciplinele fundamentale) prin transferul și utilizarea acestora în cadrul activităților de învățare a programării și reciproc.

b) Obiective educativ-formative

- Formarea și dezvoltarea unor priceperi și deprinderi practice de utilizare a calculatoarelor personale.
- Formarea la copii a gîndirii algoritmice.
- Stimularea interesului acestora pentru informatică, în vederea utilizării, cu eficiență maximă, a tehnicii de calcul în diverse domenii de activitate.
- Înțelegerea de către copii a importanței cunoașterii și utilizării acestei tehnici de vîrf în economia națională.
- Realizarea unei juste preorientări și orientări școlare și profesionale a pînierilor și școlărilor.

3. PROGRAMAREA CONȚINUTULUI ACTIVITĂȚII

I. Clasele III—IV

Nr. crt.	TEMA PROPUȘĂ	Nr. ore
1.	Activități organizatorice și instructaj N.T.S.M.	4
2.	Noțiuni introductive	6
3.	Algoritmi și scheme logice	16
4.	Noțiuni de programare	44
Total ore		70

II. Clasele V—VI

Nr. crt.	TEMA PROPUȘĂ	Nr. ore
1.	Activități organizatorice și instructaj N.T.S.M.	2
2.	Noțiuni introductive	4
3.	Baze de numerație	6
4.	Algoritmi și scheme logice	18
5.	Noțiuni de programare	40
Total ore		70

III. Clasele VII—VIII

Nr. crt.	TEMA PROPUȘĂ	Nr. ore
1.	Activități organizatorice și instructaj N.T.S.M.	2
2.	Noțiuni introductive	4
3.	Baze de numerație	6
4.	Algoritmi și scheme logice	12
5.	Noțiuni de programare ; limbajul BASIC	46
Total ore		70

4. ACTIVITĂȚI POLITICO-EDUCATIVE DE MASĂ

— Vizite la centre de calcul electronic, institute de cercetare științifică și inginerie tehnologică, la întreprinderi cu un grad mare de producție asistată de calculator.

— Prezentarea unor seturi de diapozitive, filme de documentare științifică etc.

— Întâlniri cu muncitori specialiști fruntași în acest domeniu, cu oameni de știință și cultură.

— Mozaicuri, matinee, almanahuri științifice (spre exemplu : „Calculatorul ? Nimic mai simplu !“, „Proiectarea asistată de calculator“, „Instruirea asistată de calculator“, „Informatica și astronomia“, „Calculatorul și cucerirea Cosmosului“, „Informatică, robotică și inteligență artificială“, „Generația a 5-a“, „Informația — o bogăție inepuizabilă“, „Societatea informatizată“, „Știința la frontierele cunoașterii“ ș.a.).

— Medalioane științifice dedicate unor personalități științifice din acest domeniu (Grigore Moisil, Ștefan Odobleja ș.a.).

— Expuneri, dezbateri, mese rotunde pe teme ca : „Pionierii de azi, specialiștii de mâine“, „Problemele de mâine, azi“, „Ne pregătim pentru viitor“, „Ce știm despre revoluția științifico-tehnică contemporană ?“.

— Demonstrații practice : „Calculatorul desenează“, „Calculatorul și muzica“, „Șahul și calculatorul“, „Jocuri logice pe calculator“.

— Concursuri gen „Cine știe, răspunde!“ — pe grupe și între grupe.

— Realizarea și prezentarea de către pionieri a unor referate și comunicări științifice pentru diferite sesiuni de acest fel.

5. ACTIVITĂȚI METODICE ȘI DE INSTRUIRE

— Întocmirea unor materiale cu caracter metodic.

— Demonstrații practice de instruire asistată de calculator pentru profesorii care predau discipline fundamentale, din zona de influență.

— Instruiri ale conducătorilor cercurilor de profil din școli.

— Organizarea și participarea la consfătuiri, simpozioane, schimburi de experiență, acțiuni interjudețene de profil.

6. TEME DE STUDIU ȘI CERCETARE

— Aspecte metodice privind însușirea de către copii a unui limbaj de programare de nivel înalt (BASIC, LOGO FORTH etc.).

— Considerații metodice privind antrenarea pionierilor în realizarea unor programe sau produse program.

— Formarea și dezvoltarea deprinderilor practice în activitatea cercului de informatică.

— Dezvoltarea și stimularea creativității pionierilor și școlărilor prin activitățile de informatică.

— Probleme privind orientarea conținutului activității în cercurile de informatică în perspectiva dezvoltării acestui domeniu și a necesarului de forță de muncă.

— Modalități de conlucrare a cercului de informatică cu diferite cercuri tehnico-aplicative și științifice.

— Instruirea asistată de calculator în cadrul cercului de informatică.

- Demonstrații practice : „Calculatorul desenează”, „Calculatorul și muzica”, „Șahul și calculatorul”, „Jocuri logice pe calculator”.
- Concursuri gen „Cine știe, răspunde !” — pe grupe și între grupe.
- Realizarea și prezentarea de către pionieri a unor rețete și experimente științifice pentru diferite sesiuni de seară lei.
1. Activități de recreere și sport.
2. Activități de învățare.
3. Activități de bază.
4. Activități de informare și instruire.
- 51 — Intocmirea unor materiale cu caracter metodice pentru demonstrații practice de instruire asistată de calculator pentru profesorii care predau discipline fundamentale din zona de instruire.
- 52 — Instruire ale conducătorilor cercurilor de profil din școli.
- Organizarea și participarea la conferințe, simpozioane, schimburi de experiență, acțiuni întreprinse de profil.

6. TEME DE ȘTIUDIU ȘI CERCETARE

1. Aspecte metodice privind instruirea de către copii a unui număr de programe de nivel înalt (BASIC, LOGO FORTH etc.).
2. Considerații metodice privind antrenarea pionierilor în realizarea unor programe sau produse program.
3. Formarea și dezvoltarea deprinderilor practice în activitatea de informatică.
4. Dezvoltarea și stimularea creativității pionierilor și școlărilor în activitatea de informatică.
5. Probleme privind orientarea conținutului activității în cercurile de informatică în perspectiva dezvoltării acestui domeniu și a necesității de instruire în cadrul cercului de informatică.
6. Modalități de construire a cercului de informatică cu diferite activități aplicative și științifice.
7. Instruirea asistată de calculator în cadrul cercului de informatică.
8. Prezentarea unor activități de instruire sau activități științifice etc.
9. Utilizarea teoriei și metodelor didactice în instruirea de informatică.
10. Calculul (cu ajutorul calculatoarelor) al unor probleme de calcul, proiectarea asistată de calculator, „Instruirea asistată de calculator”, „Informatică și economie”, „Calculul asistat de calculator”, „Informatică și medicină”, „Informatică și agricultură”, „Informatică și industrie”, „Informatică și comerț”, „Informatică și educație”, „Informatică și sport”, „Informatică și artă”, „Informatică și știință”, „Informatică și societate”.
11. Medalioane științifice dedicate unor personalități științifice din acest domeniu (Grigore Moisil, Ștefan Odobleja ș.a.).
12. Exerciții, dezbateri, mere rotunde pe teme ca : Pionierii de azi, „Informatică și viitor”, „Problemele zilei”, „Știință și informatică”, „Știința și informatică”, „Știința și informatică”, „Știința și informatică”, „Știința și informatică”.

PROGRAM TEMATIC DEFALCAT

al cercului de informatică, grupe de începători, clasele III—IV

Nr. crt.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
1.	Activitate organizatorică : — formarea grupelor — stabilirea responsabilităților — prezentarea tematicii cercului — instructaj N.T.S.M.	2	Planșe, schițe, diapozitive, filme NTSM, calculatoare din dotare	—
2.	Verificarea și fixarea însușirii instructajului	2	Idem	—
3.	Ce este și cum funcționează un calculator electronic ? Generalități despre calculatorul electronic : — modul de alcătuire și funcționare — scurt istoric — evoluția (generații de calculatoare)	2	Fotografii, planșe, filme documentare științifice, calculatoarele din dotare	—
4.	Familiarizarea copiilor cu calculatoarele personale din dotare	2	Idem	— Vizită la un centru de calcul electronic. Activitate comună a 3—4 grupe
5.	Calculatoare electronice românești	2	Documentele de partid și de stat, filme documentare științifice	—
6.	Introducerea noțiunii de algoritmi prin exemple simple : — adunarea a două numere — efectuarea sumei primelor trei numere naturale — înmulțirea ca adunare repetată — calculul ariei și perimetrului unui dreptunghi, când se cunosc L și l	2	Planșe didactice, manuale, programe de instruire asistată de calculator, concepute pentru cele din dotare	—
7.	Împărțirea a două numere naturale : — calculul vitezei unui mobil	2	Idem	—
8.	Algoritmi : — definiție ; noțiunea de pas al unui algoritmi	2	Idem	—

Nr. crt.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
	— operația de atribuire; „regula celor trei scaune“			
	— aplicație la ordonarea crescătoare a trei numere naturale			
	— schema logică a unui algoritm (realizarea schemelor logice ale algoritmilor studiați anterior)			
9.	Algoritmii și schemele logice ale unor probleme simple	2	Planșe didactice, manuale, programe de instruire asistată de calculator, conștute pentru cele din dotare	—
10.	Algoritmii și schemele logice ale unor probleme simple	2	Idem	—
11.	Algoritmii și scheme logice : — aruncarea monedei — alte jocuri	2	Idem	—
12.	Limbaje de programare ; prezentarea generală a limbajului BASIC	2	Idem	—
13.	Constante, variabile, expresii	2	Idem	—
14.	Modurile de lucru ale calculatoarelor din cerc ; folosirea acestora ca mașini de scris	2	Idem	—
15.	Instrucțiuni și comenzi în limbajul BASIC ; PRINT ; RUN ; DELETE	2	Idem	—
16.	LIST, CLS, NEW	2	Idem	—
17.	GO TO, REM	2	Idem	—
18.	LET	2	Idem	—
19.	BEEP, PAUSE	2	Idem	—
20.	LOAD, SAVE, VERIFY	2	Idem	—
21.	INPUT	2	Idem	—
22.	PLOT	2	Idem	—
23.	DRAW	2	Idem	—

Nr. crt.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
24.	CIRCLE; OVER	2	Programa de instruire asistată de calculator, concepută pentru cele din domeniul...	Activitate practică grup și individuală
25.	FOR... NEXT	2	Lectură și exerciții de aplicare a...	—
26.	FOR	2	Lectură și exerciții de aplicare a...	—
27.	END	2	Lectură și exerciții de aplicare a...	—
28.	IF	2	Lectură și exerciții de aplicare a...	—
29.	IF	2	Lectură și exerciții de aplicare a...	—
30.	STOP; CONTINUE	2	Lectură și exerciții de aplicare a...	—
31.	FLASH; PAPER; INK; INVERSE; BORDER; POINT; ATTR	2	Lectură și exerciții de aplicare a...	—
32.	Conceperea și rularea de programe de către pionieri	2	Lectură și exerciții de aplicare a...	—
33.	Pregătirea pentru concursul anual de informatică	2	Lectură și exerciții de aplicare a...	—
34.	Pregătirea pentru concursul anual de informatică	2	Lectură și exerciții de aplicare a...	—
35.	Concursul anual de informatică	2	Lectură și exerciții de aplicare a...	—

15. Metode de lucru ale calculatoarelor din domeniul...

16. Metode de lucru ale calculatoarelor din domeniul...

17. Metode de lucru ale calculatoarelor din domeniul...

18. Metode de lucru ale calculatoarelor din domeniul...

19. Metode de lucru ale calculatoarelor din domeniul...

20. Metode de lucru ale calculatoarelor din domeniul...

21. Metode de lucru ale calculatoarelor din domeniul...

22. Metode de lucru ale calculatoarelor din domeniul...

23. Metode de lucru ale calculatoarelor din domeniul...

24. Metode de lucru ale calculatoarelor din domeniul...

25. Metode de lucru ale calculatoarelor din domeniul...

26. Metode de lucru ale calculatoarelor din domeniul...

27. Metode de lucru ale calculatoarelor din domeniul...

28. Metode de lucru ale calculatoarelor din domeniul...

29. Metode de lucru ale calculatoarelor din domeniul...

30. Metode de lucru ale calculatoarelor din domeniul...

31. Metode de lucru ale calculatoarelor din domeniul...

32. Metode de lucru ale calculatoarelor din domeniul...

33. Metode de lucru ale calculatoarelor din domeniul...

34. Metode de lucru ale calculatoarelor din domeniul...

35. Metode de lucru ale calculatoarelor din domeniul...

PROGRAM TEMATIC DEFALCAT
al cercului de informatică, grupe de începători, clasele V—VI

Nr. crt.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
1.	Activitatea organizatorică : — formarea grupelor — stabilirea reponsabilităților — prezentarea tematicii cercului — instructaj N.T.S.M.	2	Planșe, schițe, diapozitive, filme NTSM, calculatoare din dotare	—
2.	Ce este și cum funcționează un calculator electronic ? Generalități despre calculatorul electronic : — modul de alcătuire și funcționare — scurt istoric — evoluția (generații de calculatoare)	2	Fotografii, planșe, filme documentare științifice, calculatoarele din dotare	—
3.	Calculatoarele electronice românești. Cunoașterea politicii partidului și statului nostru în domeniul tehnicii de calcul și informaticii	2	Documentele de partid și de stat ; filme documentare științifice	Vizită la un centru de calcul electronic. Activitate comună a 3—4 grupe
4.	Baze de numerație : reprezentarea (scrierea) numerelor în baza 10 și în baza 2	2	Planșe didactice, manuale, lucrări de specialitate, calculatoare	Activitate practică de grup și individuală
5.	Baze de numerație : reprezentarea numerelor în baza 10	2	Idem	Idem
6.	Trecerea (conversia) numerelor dintr-o bază de numerație în alta : — jocul „Ghicește numărul prin da și nu“	2	Planșe didactice, manuale, lucrări de specialitate, calculatoare	Idem
7.	Introducerea noțiunii de algoritmi prin exemple simple — adunarea a două numere — efectuarea sumei primelor trei numere naturale — calculul ariei și perimetrului unui dreptunghi, când se cunosc L și l — împărțirea a două numere naturale — calculul vitezei unui mobil	2	Programa de instruire asistată de calculator, concepute pentru cele din dotare	Idem

Nr. crt.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
8.	Algoritmi : — definiție ; noțiunea de pas al unui algoritm — operația de atribuire ; „regula celor trei scaune” — aplicație la ordonarea crescătoare a trei numere naturale — schema logică a unui algoritm (realizarea schemelor logice ale algoritmilor studiați anterior)	2	Programe de instruire asistate de calculator, concepute pentru cele din dotare	Activitate practică de grup și individuală
9.	Algoritmi și scheme logice : — aflarea unei fracții (procent) — aflarea unei fracții (procent) dintr-un număr — aflarea unui număr când se cunoaște o fracție (un procent) din el	2	Idem	—
10.	Algoritmi și scheme logice : — conversia unui număr dintr-o bază mai mică decât 10, în baza 10 — conversia unui număr din baza 10 într-o bază $b < 10$ ($b = 2, 8$)	2	Idem	—
11.	Algoritmi și scheme logice : — c.m.m.d.c. — c.m.m.m.c.	2	Idem	—
12.	Algoritmi și scheme logice : — jocul „Ghicește numărul” — aruncarea monedei — clasificarea algoritmilor	2	Idem	—
13.	Algoritmi și scheme logice recapitulative	2	Idem	—
14.	Limbaje de programare ; prezentarea generală a limbajului BASIC ; constante, variabile, expresii, clasificare	2	Idem	—
15.	Modurile de lucru ale calculatoarelor din dotare ; folosirea calculatorului ca mașină de scris	2	Idem	—

Nr. crt.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
16.	Instrucțiuni și comenzi în limbajul BASIC : PRINT ; RUN ; DELETE	2	Programe de instruire asistate de calculator, concepute pentru cele din dotare	—
17.	LIST ; GO TO ; CLS ; NEW	2	Idem	—
18.	PRINT AT ; PRINT TAB	2	Idem	—
19.	REM, LET	2	Idem	—
20.	BEEP, PAUSE	2	Idem	—
21.	LOAD, SAVE, VERIFY	2	Idem	—
22.	INPUT, INPUT LINE	2	Idem	—
23.	PLOT, DRAW	2	Idem	—
24.	CIRCLE, OVER	2	Idem	—
25.	Scrierea și rularea unor programe proprii de către copii	2	Idem	—
26.	FOR... NEXT	2	Idem	—
27.	RND, RANDOMIZE, INT, LEN, ABS	2	Idem	—
28.	IF	2	Idem	—
29.	IF, STOP, CONTINUE	2	Idem	—
30.	DIM, READ, DATA	2	Idem	—
31.	GO SUB, RETURN	2	Idem	—
32.	FLASH, PAPER, INK, INVERSE, BORDER, POINT, ATTR	2	Idem	—
33.	Conceperea unor programe utilizând instrucțiunile studiate	2	Idem	—
34.	Pregătirea pentru concursul anual de informatică	2	Idem	—
35.	Concursul anual de informatică	0,2	Idem	—

PROGRAM TEMATIC DEFALCAT

al cercului de informatică, grupe de începători, clasele VII—VIII

Nr. crt.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
1.	<p>Activitate organizatorică :</p> <ul style="list-style-type: none"> — formarea grupelor — stabilirea responsabilităților — prezentarea tematicii cercului — instructaj N.T.S.M. 	2	Planșe, schițe, diapozitive, filme NTSM, calculatoarele din dotare	—
2.	<p>Ce este și cum funcționează un calculator electronic ?</p> <ul style="list-style-type: none"> — alcătuire și mod de funcționare — scurt istoric — evoluție (generații de calculatoare) 	2	Fotografii, planșe, calculatoare din dotare, filme de documentare științifică	—
3.	<p>Calculatoare electronice românești ; cunoașterea politicii partidului și statului nostru în domeniul tehnicii de calcul și informaticii</p>	2	Documentele de partid și de stat ; film documentar-științific	Vizită la un centru de calcul electronic. Activitate comună a 3—4 grupe
4.	<p>Baze de numeratie : reprezentarea numerelor în baza 10 și în baza 2</p>	2	Planșe didactice ; calculatoarele personale ; diferite manuale sau lucrări de specialitate	Activitate practică de grup și individuală (independentă)
5.	<p>Baze de numeratie : reprezentarea numerelor în baza 8 și în baza 16</p>	2	Idem	Idem
6.	<p>Trecerea (conversia) numerelor dintr-o bază de numeratie în alta</p>	2	Idem	Idem
7.	<p>Noțiunea de algoritm prin exemple simple : rezolvarea ecuației de gr. I (aplicație : $F 1 = G h$ planul înclinat $P = \frac{G h}{\eta}$ ($\eta =$ randamentul mecanic)</p>	2	Planșe didactice ; diferite manuale sau lucrări de specialitate ; programe de instruire asistată la calculator, concepute pentru calculatoarele din dotare	Idem

— efectuarea sumei a trei numere naturale
— calculul ariei unui triunghi

Nr. ord.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
—	<p>— extragerea rădăcinii pătrate dintr-un număr rațional</p> <p>— calculul ipotenuzei unui triunghi dreptunghic, când se cunosc catetele b și c</p> <p>— calculul distanței focale a unei lentile subțiri</p> <p>$f = \sqrt{d \cdot d'}$</p> <p>— aflarea modului unui număr</p>	2		
8.	<p>Algoritm : definiție, noțiunea de pas al unui algoritm și operația de atribuire ; „regula celor trei scaune“</p> <p>— ordonarea crescătoare a 3 numere naturale</p> <p>— schema logică a unui algoritm (realizarea schemelor logice ale algoritmilor studiați în ședința anterioară)</p>	2	<p>Planșe didactice, diferite manuale sau lucrări de specialitate, programe de instruire asistată de calculator, concepte pentru calculatoarele din dotare</p>	<p>Activitate practică de grup și individuală (independentă)</p>
9.	<p>Algoritmi și scheme logice :</p> <p>— conversia unui număr dintr-o bază mai mică decât 10 în baza 10</p> <p>— conversia unui număr din baza 10 în bază b, $b < 10$ ($b = 2, 8$)</p>	2	Idem	Idem
10.	<p>Algoritmi și scheme logice :</p> <p>— concentrația unei soluții</p> <p>— calcule chimice pe baza formulelor și reacțiilor chimice</p> <p>— c.m.m.d.c.</p>	2	Idem	Idem
11.	<p>Algoritmi și scheme logice :</p> <p>— suma primelor cinci numere naturale</p> <p>— produsul primelor n numere naturale</p> <p>— clasificarea algoritmilor</p>	2	Idem	Idem
12.	<p>Algoritmi și schemele logice ale unor probleme propune de copii ; clasificarea algoritmilor</p>	2	Idem	Idem
13.	<p>Limbaje de programare ; prezentarea generală a limbajului BASIC ; constante, variabile, expresii, clasificare</p>	2	<p>Planșe didactice, calculatoare, manual de utilizare, alte manuale sau lucrări, programe de instruire asistată de calculator</p>	Idem

Nr. crt.	Tema (continutul) activității	Nr. ore	Materiale necesare	Observații
14.	Modurile de lucru ale calculatoarelor din dotare ; folosirea calculatorului ca mașină de scris	2	Planșe didactice, calculatoare, manual de utilizare, alte manuale sau lucrări, programe de instruire asistată de calculator	Activitate practică de grup și individuală (independentă)
15.	Instrucțiuni și comenzi în limbajul BASIC ; PRINT (comandă calcule directe ; instrucțiuni) ; RUN ; DELETE	2	Idem	Idem
16.	LIST, GO TO, CLS, NEW	2	Idem	Idem
17.	PRINT AT, PRINT TAB, REM, LET	2	Idem	Idem
18.	BEEP, PAUSE	2	Idem	Idem
19.	LOAD, SAVE, VERIFY, MERGE	2	Idem	Idem
20.	INPUT, INPUT LINE	2	Idem	Idem
21.	PLOT, DRAW, CIRCLE, OVER	2	Idem	Idem
22.	Conceperea, realizarea și rularea de către copii a unor programe care utilizează instrucțiunile învățate	2	Idem	Idem
23.	Conceperea, realizarea și rularea de către copii a unor programe care utilizează instrucțiunile învățate	2	Idem	Idem
24.	Recapitularea cunoștințelor dobândite în trimes-trele anterioare, prin exerciții aplicative	2	Idem	Idem
25.	FOR... NEXT	2	Idem	Idem
26.	RND, RANDOMIZE ; FUNCȚII : SIN, COS, TAN, SQRT, ABS, INT, LEN	2	Idem	Idem
27.	IF	2	Idem	Idem
28.	IF, CONTINUE, STOP	2	Idem	Idem
29.	DIM, READ, DATA, RESTORE	2	Idem	Idem
30.	DEFN	2	Idem	Idem
31.	GO SUB... RETURN	2	Idem	Idem

Nr. crt.	Tema (conținutul) activității	Nr. ore	Materiale necesare	Observații
32.	FLASH, PAPER, INK, INVERSE, BORDER, POINT, ATTR	2	Planșe didactice, calculatoare, manualul de utilizare, alte manuale sau lucrări, programe de instruire asistată de calculator	Activitate practică de grup și individuală (independentă)
33.	Conceperea și realizarea unor programe utilizând instrucțiunile studiate	2	Idem	Idem
34.	Pregătirea pentru concursul anual de informatică	2	Idem	Idem
35.	Concursul anual de informatică	2	Idem	Idem

CÎTEVA MODALITĂȚI DE INTEGRARE A CALCULATORULUI CA MIJLOC DE ÎNVĂȚĂMÎNT

Utilizarea calculatorului în procesul de învățămînt are o oarecare tradiție. Primele încercări au fost realizate cu ajutorul terminalelor de tip display, cuplate la un calculator central. Deși utile, sistemele de acest fel nu s-au extins, datorită costurilor ridicate, nici chiar în țările avansate din punct de vedere industrial.

Apariția microcalculatoarelor permite abordarea acestei probleme de pe noi poziții, deoarece aceste „calculatoare personale“ pot fi folosite de către elevi, atît la școală, cît și la domiciliu. Facilitățile hardware și software de care dispun în acest moment „calculatoarele personale“ permit cuplarea lor la un calculator central, dotat cu bază de date (cunoștințe). În acest context asistarea procesului de învățămînt de către calculator capătă noi dimensiuni.

Asistarea procesului de învățămînt cu calculatorul este o problemă de dată recentă, dar de mare perspectivă și cu consecințe pozitive pentru optimizarea predării-învățării, pentru integrarea învățămîntului cu cercetarea, cu producția, cu viața.

Conceptul de asistare cu calculatorul a procesului de învățămînt include atît predarea cu calculatorul a unei lecții de comunicare a noilor cunoștințe, de aplicare, de sistematizare și consolidare a acestora, cît și verificarea automată a unei lecții sau a unui grup de lecții, a unei discipline școlare la finele unui trimestru sau an școlar. Tot în cadrul „asistării“ este inclusă și prezentarea sau verificarea cu calculatorul a unor secvențe ale lecțiilor desfășurate după metodologia clasică. În acest fel, calculatorul devine un auxiliar al procesului de învățămînt.

Învățarea cu ajutorul calculatorului se poate realiza în cadrul unor clase sau laboratoare dotate cu unul sau mai multe microcalculatoare (ideal ar fi ca fiecare elev să aibă calculatorul său pe pupitrul). Lecția, pregătită de profesor în prealabil, se derulează secvență cu secvență pe ecranul calculatorului sau, în cazul utilizării calculatorului ca auxiliar al procesului de învățămînt, numai anumite secvențe ale lecției clasice vor fi urmărite pe ecran.

Predarea lecției prin intermediul ecranului se realizează printr-o succesiune de imagini, numite „imagini ecran“, a căror derulare este

dirijată de către profesor, în concordanță cu particularitățile psihopedagogice și de vîrstă ale copiilor, nivelul clasei și particularitățile obiectului predat.

În lecțiile predate cu ajutorul calculatorului sînt prevăzute și exerciții de control, răspunsurile putînd fi adeseori selecționate dintr-o listă de variante afișată pe terminal. Metoda folosită în acest caz este, oarecum, similară învățămîntului programat. În cazul în care lecția urmărește testarea cunoștințelor, atunci fiecărei imagini-ecran i se asociază una sau mai multe întrebări.

Răspunsurile la întrebări sînt cotate cu un anumit punctaj, acesta depinzînd de calitatea răspunsului și de timpul în care a fost dat.

Instruirea cu ajutorul calculatorului se poate realiza și în afara schemei clasice a procesului de învățămînt. Elevii sau orice altă persoană interesată pot beneficia, într-un anumit cadru, de posibilitatea autoinstruirii automatizate. Aceasta se realizează fără profesor, activitatea sa fiind însă materializată în lecțiile afișate pe ecran. Dialogul dintre calculator și cel care învață singur se realizează prin intermediul consolei. Lecția sau grupul de lecții sînt stocate în memoria calculatorului și acestea sînt afișate treptat, imagine cu imagine, pe ecranul acestuia, furnizînd astfel materiale de învățat.

În cadrul autoinstruirii cu calculatorul, utilizatorul își poate regla singur ritmul de afișare a imaginilor pe ecran; el își poate întrerupe pregătirea în orice moment al lecției, reluînd-o de la secvența la care s-a oprit.

În dialogul dintre calculator și cel care învață, se folosește și funcția de „help“ a calculatorului. Prin aceasta sînt puse la dispoziția utilizatorului, sub formă de imagini ecran, comentariile și explicațiile necesare înțelegerii și însușirii corecte a unor noțiuni — fără de care lecția nu poate continua. Calculatorul are posibilitatea de a detecta imediat eroarea în însușirea unei secvențe din lecție și de a facilita remedierea acesteia prin comentarii făcute și indicarea informațiilor suplimentare necesare continuării instruirii. Cel care învață poate trece la secvența următoare a lecției numai dacă a răspuns corect la întrebările sau exercițiile propuse, interacțiunea dintre utilizator și calculator realizîndu-se permanent pe bază de dialog.

În cele mai multe situații, calculatorul poate fi înzestrat și cu posibilitatea de a memora performanțele celui care învață: timpul necesar pentru răspuns, nota obținută, întrebările la care frecvența răspunsurilor slabe este mare etc. În acest mod sînt furnizate fie elevului, fie profesorului, o serie de date pedagogice individuale, sau caracteristice unei anumite populații școlare, care sînt necesare optimizării acestor programe și lecțiilor recapitulative sau de control curente ce se vor organiza.

În cazul Palatului pionierilor și șoimilor patriei, la cercurile de informatică, astronomie și radiotelegrafie se utilizează cu rezultate deosebite atît formele instruirii asistate de calculator, cît și cele ale autoinstruirii.

Cercul de astronomie utilizează 1/4 din lecțiile specifice sub forma prezentării lor secvență cu secvență pe calculator, iar circa 1/3 din ele ca lecții combinate, unde calculatorul este folosit ca auxiliar. Din acest

punct de vedere, utilizarea calculatorului în predarea-învățarea astronomiei a demonstrat că acest mod de lucru oferă lecției noi posibilități de dezvoltare și evaluare. Astfel, prin intermediul consolei, pot fi simulate pe ecran procese și fenomene în evoluția lor, unele experiențe greu accesibile de realizat în practică, datorită depărtării lor în timp sau în spațiu.

Autoinstruirea cu ajutorul calculatorului a dat rezultate deosebite în cadrul cercului de informatică. Trebuie subliniat că mai mult de 70% din copiii care au frecventat cercul în anul școlar 1986/1987 au învățat să utilizeze calculatorul, limbajele de programare LOGO și BASIC, precum și modul de realizare a programelor în aceste limbaje, prin utilizarea unui set de 16 lecții rulate pe calculatoarele TPD-JUNIOR, HC-85, TIM-S aflate în dotarea cercului. Aceste lecții de programare, foarte ingenios realizate cu sprijinul unor specialiști de la I.T.C.I. și al studenților de la Institutul Politehnic București s-au dovedit deosebit de eficiente în inițierea și instruirea copiilor în limbajele BASIC și LOGO ; majoritatea copiilor au început să realizeze lecții de matematică, fizică, chimie, istorie, geografie, astronomie.

Folosirea calculatorului în cadrul cercurilor de la Palatul Pionierilor și șoimilor patriei a demonstrat că există multiple posibilități de optimizare a predării-învățării cu ajutorul ecranului. Posibilitatea de a construi o largă varietate de exemple (în funcție de pregătirea, ingeniozitatea și experiența profesorului), a numeroase modele asociate unor secvențe ale lecției (de o mare eficiență se bucură animația pe calculator), concură la adâncirea și lărgirea orizontului noțiunilor predate, uneori extrapolându-le dincolo de disciplina predată ; tehnică, economie, știință, practică etc. În acest mod s-ar putea vorbi despre o „dilatare“ a sferei aplicative a noțiunilor predate, precum și de o „comprimare“ a timpului necesar însușirii și aplicării cunoștințelor dobândite. Acest aspect conduce la dezvoltarea în rîndul copiilor a spiritului de creativitate, inventivitate, de anticipație tehnico-științifică. De asemenea, utilizarea calculatorului are consecințe importante asupra formării intelectuale a copiilor în spiritul autoeducației și, prin aceasta, al educației permanente, fiind stimulate în permanență toate componentele gândirii logice în sens larg : gândirea logico-deductivă, introductivă, analogică, cu accent pe gândirea euristică.

Însușindu-și cunoștințele în fața consolei, elevul învață singur și sigur, adeseori în ritm optim propriu, fără emoții și fără perturbări ale comportamentului de către diverși factori legați de mediul său înconjurător. În fața consolei, elevul privește obiectiv și cu optimism „nota“ acordată răspunsurilor și deci pregătirii sale ; învață de aici să-și aprecieze calitatea și durata pregătirii, performanțele atinse și, mai ales, învață să-și mobilizeze resursele de energie și de voință pentru o nouă calitate a muncii sale.

Experiența dobîndită pînă în acest moment în ceea ce privește utilizarea calculatorului în procesul instructiv-educativ specific caselor pionierilor și șoimilor patriei, precum și în unele unități școlare, pledează cu suficiente argumente pentru o acțiune mai fermă din partea factorilor de răspundere în formarea și educarea tinerei generații în

vederea introducerii pe scară largă a calculatorului în desfășurarea procesului de învățămînt la aproape toate disciplinele școlare, inclusiv la cele tehnologice, putîndu-se astfel realiza în mai bune condițiuni și componenta tehnică a educației multilaterale a personalității umane. Este necesar să se acorde o mai mare atenție problemei integrării calculatorului ca mijloc de învățămînt, cu atît mai mult cu cît tînăra generație aflată astăzi pe băncile școlii va fi cea care va folosi cu precădere tehnica de calcul modernă în momentul în care va intra în viața productivă. Trebuie să pregătim oamenii viitorului cu uneltele viitorului!

Într-o utilizare a calculatorului în procesul de învățămînt, este necesar să se acorde o mai mare atenție problemei integrării calculatorului ca mijloc de învățămînt, cu atît mai mult cu cît tînăra generație aflată astăzi pe băncile școlii va fi cea care va folosi cu precădere tehnica de calcul modernă în momentul în care va intra în viața productivă. Trebuie să pregătim oamenii viitorului cu uneltele viitorului!

Utilizarea calculatorului în procesul de învățămînt este o problemă complexă care implică o serie de aspecte tehnice, pedagogice și psihologice. Este necesar să se acorde o mai mare atenție problemei integrării calculatorului ca mijloc de învățămînt, cu atît mai mult cu cît tînăra generație aflată astăzi pe băncile școlii va fi cea care va folosi cu precădere tehnica de calcul modernă în momentul în care va intra în viața productivă. Trebuie să pregătim oamenii viitorului cu uneltele viitorului!

Utilizarea calculatorului în procesul de învățămînt este o problemă complexă care implică o serie de aspecte tehnice, pedagogice și psihologice. Este necesar să se acorde o mai mare atenție problemei integrării calculatorului ca mijloc de învățămînt, cu atît mai mult cu cît tînăra generație aflată astăzi pe băncile școlii va fi cea care va folosi cu precădere tehnica de calcul modernă în momentul în care va intra în viața productivă. Trebuie să pregătim oamenii viitorului cu uneltele viitorului!

BIBLIOGRAFIE

1. Nicolae Ceaușescu — „Raport la cel de-al XIII-lea Congres al Partidului Comunist Român“, Editura Politică, București, 1984.
2. Nicolae Ceaușescu — „Cuvîntare la Congresul Științei și Învățămîntului“, Editura Politică, București, 1985.
3. Nicolae Ceaușescu — „Cuvîntare la Congresul Educației politice și Culturii Socialiste“, Scînteia, 18 august 1987.
4. Nicolae Ceaușescu — „Raport la Conferința Națională a Partidului Comunist Român“, Editura Politică, București, 1987.
5. * * * — „Programul Partidului Comunist Român de făurire a societății socialiste multilateral dezvoltate și înaintare a României spre comunism“, Editura Politică, București, 1975.
6. Perjeriu, E., Văduva, I. — „Îndrumar pentru lucrări de laborator la cursul de bazele informaticii anul I“, București, 1986.
7. Preoteasa, P., Serbănași, I. D. — „Matematică aplicată în tehnica de calcul“, manual pentru clasa a XI-a, Editura Didactică și Pedagogică.
8. Tomescu, I., Leu, A. — „Matematică aplicată în tehnica de calcul“, manual pentru clasa a X-a, Editura Didactică și Pedagogică, 1981.
9. Petrescu, A. și colectiv — „Totul despre... calculatorul personal aMIC“, Editura Tehnică, 1985.
10. * * * — „HC-85, manual de utilizare“, I.C.E., 1985.
11. * * * — „TIM-S — microcalculator personal, manual de funcționare și utilizare“, I.T.C.I. Timișoara.
12. * * * — „Sistemul CP/M implementat pe microcalculatoarele TPD și JUNIOR“, IEPER București.
13. Patrubany, N. și colectiv — „Familia de calculatoare personale românești PRAE și limbajul lor BASIC“, volumul amc 51, 1985.
14. Albu, I., Vass, Gh. — „Astronomie, matematică, informatică“, 1985.

Lucrarea executată sub comanda nr. 1651 la Oficiul Economic:
Central „Carpați”, Întreprinderea Poligrafică „Bucureștii Noi”,
str. Hrisovului nr. 18 A, sectorul 1, București

