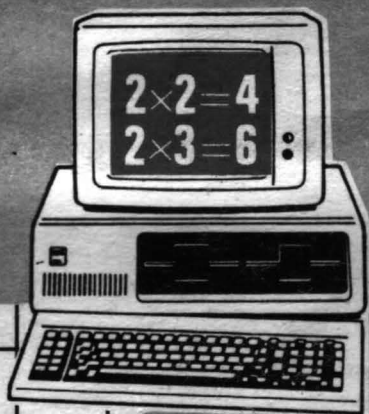


MARIAN GHEORGHE

Cine ești tu BASIC?



Editura
Agni

*Biblioteca
de
Informatică*

Marian Gheorghe

**Cine ești tu,
BASIC?**

Editura *Agni*

București

1994

Redactor : *Rodica Ceterchi*
Tehnoredactare computerizată : *Aurelian Lavric*
Coperta : *Adina Dumitriu*

ISBN 973-95626-4-7

© Toate drepturile sînt rezervate Editurii AGNI.

Editura AGNI,
CP:30-107, BUCUREȘTI

CUPRINS

Introducere	3
1 Programe simple în limbajul BASIC	7
2 Comenzi de bază. Ecranul de afișare	14
3 Instrucțiunile de citire și scriere. Instrucțiunea STOP	20
4 Constante, variabile numerice, expresii aritmetice. Instrucțiunea de atribuire. Comentariul	27
5 Expresii logice. Instrucțiunile IF și GO TO	35
6 Scheme logice	47
7 Instrucțiunea FOR...NEXT. Valoarea absolută, partea întreagă, radicalul	53
8 Instrucțiunile READ, DATA, RESTORE	68
9 Șiruri de caractere. Variabile și expresii de tip șir de caractere	75
10 Probleme rezolvate în limbajul BASIC	85
11 Variabile structurate	95
12 Subrutine	108
13 Instrucțiunea BEEP	116
14 Elemente de grafică în limbajul BASIC	122
15 Variabile structurate de tip tablou. Variabile structurate, șiruri de caractere	137
16 Probleme rezolvate și propuse	158
17 Limbajul GW-BASIC. Diferențe față de limbajul prezentat	168
Anexa 1 Lista instrucțiunilor	184
Anexa 2 Lista comenzilor	185
Anexa 3 Lista operațiilor	186
Anexa 4 Lista exemplurilor de programe	187
Bibliografie	191

Au apărut:

Ion Diamandi

- ◆ Cum să realizăm jocuri pe calculator

Luminița State

- ◆ Hello, BASIC

Mihaela Cârstea, Ion Diamandi

- ◆ Un PC pentru fiecare

Ion Diamandi

- ◆ Calculatorul, coleg de bancă

Adrian Atanasiu

- ◆ Cum se scrie un algoritm? simplu

În curînd vor apărea:

Vlad Atanasiu

- ◆ Minunata lume a HC-ului

Ion Diamandi

- ◆ Cine știe LOGO ?

Răzvan Andonie, Ilie Gârbacea

- ◆ Algoritmi fundamentali în C++

Puteți să primiți aceste cărți scriindu-ne pe adresa:

EDITURA AGNI C.P. 30-107 București

Model de cerere:

Numele... Localitatea... Str... Nr... Bl... Et... Ap... Judet... Cod...

Vă rog să-mi expediați prin colet poștal cu ramburs:

Cartea... Nr.exemplare... Data... Semnătura...

INTRODUCERE

Cartea cuprinde noțiunile cele mai des întâlnite în limbajul BASIC, dar face unele precizări cu privire la varianta acestui limbaj disponibilă pe calculatoarele personale din familia Sinclair SPECTRUM (TIM-S, HC-85, HC-90, COBRA etc.) și un scurt rezumat al caracteristicilor variantei GW-BASIC (disponibilă pe calculatoare compatibile IBM-PC).

Limbajul BASIC creat în jurul anului 1965 la Colegiul Dartmouth și ajuns apoi la un nucleu standardizat este destinat în general inițierii în informatică avînd caracteristicile necesare unui asemenea scop: simplitate, conciziune, arie de exprimare a problemelor relativ largă.

Dacă în anii 70 limbajul era destinat inițierii în informatică a studenților din primii ani ai facultăților tehnice, economice și de știință, în anii 80, odată cu apariția calculatoarelor personale și a creșterii interesului pentru informatică al copiilor la vîrste din ce în ce mai mici, limbajul BASIC a devenit limba de comunicare a micilor informaticieni cu calculatoarul. Acesta este motivul pentru care propunem limbajul BASIC ca limbaj de inițiere în informatică pentru elevii de gimnaziu. Cartea se adresează elevilor claselor a VI-a (primele 10 capitole) și celor din clasa a VII-a, pentru trimestru I (capitolele 11 la 16).

Cartea nu urmărește prezentarea sistematică a noțiunilor limbajului BASIC ci o introducere graduală (cu exemple și exerciții rezolvate, dar și propuse) a materialului. Astfel, manualul conține atît noțiuni specifice limbajului BASIC (instrucțiunile de intrare/ieșire, de atribuire, IF, GO TO, FOR NEXT, READ, DATA, RESTORE, la clasa a VI-a și subrutine, variabile structurate, elemente de grafică la clasa a VII-a), cît și unele concepte ale limbajului ce modelează definiții din matematică sau fizică (valoarea absolută, rădăcina pătrată, partea întreagă) și probleme, în general de matematică, care sînt rezolvate cu calculatorul (probleme de divizibilitate, exprimări procentuale, operații cu mulțimi, calcularea valorii unor polinoame, construcția unor figuri geometrice).

Materialul de față presupune citirea cărților destinate elevilor de clasa a V-a și înțelegerea unor noțiuni și fapte descrise acolo: **memoria calculatorului și locație de memorie, tastatura**, cu evidențierea unor **taste funcționale** cu rol special, punerea în funcțiune a calculatorului (legarea la rețeaua electrică, conectarea la ecranul pe care afișăm, eventual racordarea la un casetofon).

Avem în vedere în expunerea de față că se lucrează pe calculatoare din familia Sinclair SPECTRUM într-o configurație minimală (calculator + monitor + casetofon). Cei care beneficiază de dischetă vor face apel la un manual care prezintă cele câteva elemente necesare lucrului cu aceasta. Posesorii de calculatoare compatibile IBM-PC după ce parcurg capitolele de la 3 la 16 vor citi și capitolul 17 care prezintă principalele diferențe dintre limbajul BASIC disponibil pe calculatoare din familia Sinclair SPECTRUM și limbajul GW-BASIC care există pe calculatoarele compatibile IBM-PC.

1

Programe simple în limbajul BASIC

1.1. Exemple de programe

Exemplele simple în limbajul BASIC care vor fi prezentate, au scopul de a ne familiariza cu noțiunea de *program*.

● Unul dintre cele mai simple programe în limbajul BASIC este următorul:

```
P1.1      10 PRINT "Primul program in limbajul BASIC"
```

Programul este format dintr-o singură instrucțiune, PRINT, numită **instrucțiune de scriere**. Efectul acestui program este acela de a afișa pe ecran următorul text:

```
Primul program in limbajul BASIC
```

● Un loc central în limbajul BASIC îl ocupă noțiunea de **variabilă** care este identificată printr-un nume și desemnează un loc în memoria calculatorului.

Următorul program "citește" două valori numerice (așteaptă să fie introduse de la tastatură) pe care le depune în variabilele A și B. Aceste valori sînt apoi afișate pe ecran.

```
P1.2      10 INPUT A, B  
          20 PRINT A, B
```

Instrucțiunea INPUT care citește valorile numerice în A și B se numește **instrucțiune de citire**. Valorile sînt afișate de instrucțiunea PRINT.

● Cu instrucțiunile deja introduse (INPUT și PRINT) vom rezolva o problemă foarte simplă, de calculare a sumei și produsului a două numere, pe care o putem codifica în următoarele acțiuni:

- Se citesc cele două numere în două variabile A și B;
- Se calculează $A+B$ și $A \cdot B$, iar rezultatele sînt depuse în S și P;
- Se afișează valorile conținute în variabilele S și P.

Textul programului care realizează ce am menționat mai sus este următorul:

P1.3

```
10>INPUT A, B
20 LET S=A+B
30 LET P=A*B
40 PRINT S, P
```

Fig. 1.1

Analizînd programul de mai sus facem următoarele *observații*:

- ◆ liniile care compun programul sînt numerotate crescător cu 10, 20, 30, 40
- ◆ instrucțiunea de citire, INPUT, care apare pe linia 10 citește în A și B două valori numerice
- ◆ în liniile 20 și 30 se calculează suma și produsul valorilor aflate în A și B care se depun în S, respectiv P; observăm că operația de înmulțire are simbolul *; instrucțiunea LET de pe aceste linii se numește **instrucțiunea de atribuire**
- ◆ linia 40 conține instrucțiunea de afișare a conținutului variabilelor A și B.

Pentru a putea scrie și executa aceste programe trebuie să mai aveți puțină răbdare pînă la parcurgerea paragrafului următor.

1.2. Introducerea unui program. Taste funcționale

Pentru a lucra un program (a-l executa sau a-l modifica), dialogăm cu calculatorul introducînd de la tastatură comenzi, instrucțiuni, litere, cifre, semne speciale etc.

Pornirea calculatorului constă din următoarele operații:

- ▶ conectarea acestuia la un monitor sau televizor (care în prealabil este pornit și reglat);
- ▶ legarea la rețeaua electrică;
- ▶ apăsarea tastei **CR** (sau **ENTER**). În continuare ne vom referi la această tastă prin **CR**.

Odată pornit calculatorul, acesta se află într-o anumită **stare**, care indică tipul informației ce trebuie să se introducă de la tastatură.

Imediat după pornire, calculatorul trece în starea de **comandă**, care pe ecran este semnalată prin litera **K**. În această stare, apăsarea unei taste, de exemplu cea pe care sînt scrise **I**, **INPUT**, **AT**, va conduce la introducerea instrucțiunii **INPUT**.

Înaintea instrucțiunii **INPUT** se poate introduce un număr, de exemplu 10. Tastînd în ordine 1, 0, **INPUT**, atunci din starea inițială prezentată în figura 1.2 se ajunge la situația prezentată în figura 1.3:

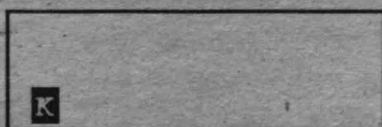


Fig. 1.2

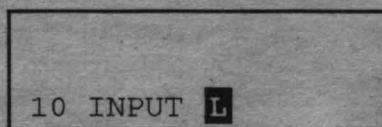


Fig. 1.3

Privind cu atenție figura 1.3, observăm că apare și o literă, **L**, care semnifică o altă stare decât cea de comandă, în care este permisă introducerea unor simboluri precum litere, cifre, semne speciale (<, >, = etc.). Dacă de exemplu se tastează pe **A** atunci pe ecran apare **a**. Pentru a obține **A** atunci trebuie să se țină apăsată tasta **CS** (sau **CAPS SHIFT**) și concomitent să se apese pe **A**. Pentru a tasta un simbol special, de exemplu **+**, se apasă pe **SS** (sau **SYMBOL SHIFT**) și în același timp pe tasta pe care scrie **+**.

Dacă, de exemplu, sîntem în situația prezentată în figura 1.3 și tastăm:

- **CS** și concomitent **A**, atunci apare **A**;
- **SS** și concomitent **N**, atunci apare **,**;
- **CS** și concomitent **B**, atunci apare **B**;

și se obține imaginea din figura 1.4:

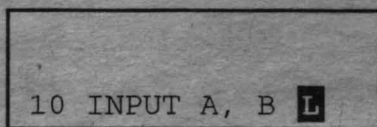


Fig. 1.4

Această linie este transmisă calculatorului atunci cînd se tastează **CR**. Se poate constata că se trece din nou în starea **K**.

Alături de cele două stări **K** și **L** menționate anterior, calculatorul poate să se mai afle în stările **E** sau **G**.

Trecerea din **K** sau **L** în **E** și invers se obține tastând simultan **SS** și **CS**. Trecerea în starea **G** și revenirea apoi în starea inițială se obține tastând simultan **CS** și **9**.

În starea **K** puteți introduce (urmărind tastatura de la stînga la dreapta și de sus în jos):

PLOT, DRAW, REM, RUN, RAND, RETURN, IF, INPUT, POKE, PRINT, NEW, SAVE, DIM, FOR, GO TO, GO SUB, LOAD, LIST, LET, CR, COPY, CLEAR, CONT, CLS, BORDER, NEXT, PAUSE.

În starea **L** se pot introduce:

- cifrele (rîndul de sus);
- literele mici: q, w, e, etc;
- cu **CS** apăsat se obțin literele mari: Q, W, E, etc.

În stările **K** sau **L** ținînd apăsată tasta **SS** se pot introduce:

!, @, #, \$, %, &, ', (,), _, <=, <>, >=, <, >, AND, OR, AT, ;, ", STOP, NOT, STEP, TO, THEN, ^, -, +, :, ?, /, *, ,, .

În starea **E** se obțin informațiile care sînt trecute deasupra tastelor (de exemplu: SIN, COS, TAN, READ etc.).

În starea **E** și ținînd **SS** apăsată se obțin informațiile trecute sub taste (de exemplu: BEEP, INK, PAPER etc.).

Executarea unui program este principalul obiectiv, deoarece în urma acestei acțiuni "culegem" roadele programării.

Pentru a executa programul din figura 1.1 se apasă, în starea **K**, tasta ce desemnează comanda **RUN** și apoi **CR**. Pe ecran apare **L**, semn că trebuie tastat un simbol; în acest caz se așteaptă tastarea unui număr care va fi citit în variabila A. Tastăm de exemplu 10 urmat de **CR**, după care

apare din nou **L**, dar pe mijlocul liniei; în acest caz trebuie tastată o nouă valoare pentru **B**. Tastăm **4** și apoi **CR**. Pe ecran sînt afișate două valori numerice, **14** și **40**, reprezentînd $10+4$ și respectiv $10 \cdot 4$. Revenirea la program se obține tastînd **CR**.

În figura 1.5 este prezentată situația după introducerea numerelor **10** și **4**, iar în figura 1.6 apar rezultatele.

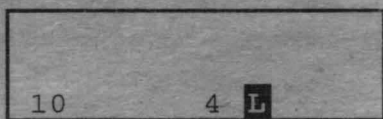


Fig. 1.5

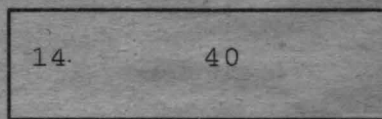


Fig. 1.6

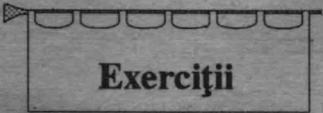
Rețineți !

■ Liniile unui program sînt numerotate crescător cu numere cuprinse între 1 și 9999. Se recomandă ca numerotarea să nu se facă prin numere consecutive (1, 2, 3) ci prin numere de tipul 10, 20, 30, deoarece modificări ulterioare ale programului trebuie să aibă numere de linie disponibile (11, 15, 25 etc.).

■ Operația de înmulțire se reprezintă prin simbolul $*$, iar operația de împărțire prin simbolul $/$.

■ Calculatorul se poate afla în stările **K**, **L**, **E**, **G**.

■ Obținerea literelor mari se realizează ținînd apăsată tasta **CS**, iar obținerea simbolurilor speciale ($;$, $:$, $+$, $-$, $*$ etc.) se realizează acționînd tasta **SS** și cea pe care se află simbolul ales.



Exerciții

1. Tastați următorul program:

```
10 INPUT A, B
30 LET D=A-B
20 LET S=A+B
40 PRINT S, B
```

Există diferențe între programul pe care l-ați tastat și cel care a apărut pe ecranul televizorului ?

2. Executați programul de la exercițiul 1, introducând datele 20 și 10.

3. Scrieți un program similar cu cel din exercițiul 1, dar schimbați -
cu * și + cu /.



2

Comenzi de bază. Ecranul de afișare

Orice activitate de programare necesită anumite acțiuni numite **comenzi**. Astfel de acțiuni se referă la scrierea, corectarea și executarea unui program.

Pentru a introduce oricare din comenzile de mai jos trebuie să aveți calculatorul în starea **K** (comandă).

Comanda de listare - LIST

Pentru a obține pe ecran textul unui program deja scris se acționează tasta **LIST** și apoi **CR**. Pentru a lista un program de la un anumit număr de linie, după comanda **LIST** se introduce numărul de linie al primei instrucțiuni ce se dorește a fi afișată. Dacă textul care se afișează are mai mult de 22 de linii (limita numărului de linii ale ecranului - a se vedea prezentarea care urmează în acest capitol) atunci după ce se afișează 22 de linii apare mesajul

```
scroll?
```

la care se poate răspunde cu **Y** pentru a continua afișarea sau cu **N** pentru a opri listarea. De exemplu, pentru programul:

```
1 LET A=1
2 LET A=2
...
40 LET A=40
41 PRINT A
```


acționarea comenzii **LIST 10** are ca efect afișarea textului începînd cu linia 10. Cînd se ajunge la linia 32 listarea se oprește și este afișat mesajul

```
scroll?
```

Dacă se tastează **Y** atunci sînt afișate și următoarele linii din program.

Comanda de rulare - RUN

Executarea unui program se obține în urma acționării tastei **RUN** și apoi a tastei **CR**. Încercați această comandă pentru programul de mai sus.

Comanda de modificare a unei linii - EDIT

Urmărind textul unui program observați că una din linii este precedată de semnul **>**. Această linie va fi numită **linia curentă**. Comanda de modificare a unei linii, care se obține tastînd **EDIT** (**CS** și **1** tastate simultan) se referă la linia curentă. Dacă dorim să modificăm altă linie decît cea curentă, se acționează săgețile **↓** (**CS** și **6** tastate simultan) și **↑** (**CS** și **7** tastate simultan) pentru a muta simbolul **>** pe linia vizată, care devine astfel linie curentă. Tastarea comenzii **EDIT** are ca efect coborîrea liniei curente în partea de jos a ecranului, pe liniile destinate introducerii comenzilor.

De exemplu, în programul:

```
1 LET A=1
2 LET A=2
3 LET A=4
4>LET A=5
```

dorim să modificăm linia 3. În acest caz tastăm **↑** și apoi **EDIT**. Este coborîtă pentru a fi modificată linia

```
3>LET A=4.
```


Modificarea unei linii se obține acționînd săgețile → (CS și 8 tastate simultan), ← (CS și 5 tastate simultan) și DELETE (CS și 0 tastate simultan). Săgețile stînga și dreapta ne mută la fiecare apăsare cîte un caracter la stînga, respectiv la dreapta. Tastînd DELETE este șters caracterul aflat la stînga poziției curente. Pentru a introduce caractere noi într-o anumită poziție, după ce se ajunge în poziția respectivă, se tastează caracterele vizate.



Important! Orice modificare se termină cu tastarea comenzii **CR**.

Dacă dorim să modificăm linia anterior vizată de comanda **EDIT**, prin înlocuirea lui A cu BC și a lui 4 cu 31, procedăm astfel:

- cu → (CS și 8) se ajunge după A;
- se tastează **DELETE**;
- se apasă apoi pe **B** și **C** (pentru a obține BC);
- se merge spre dreapta pînă în poziția următoare lui 4 și se tastează **DELETE** și apoi 3 și 1;
- ultima comandă este **CR**.

Linia 3 a programului anterior devine:

3 LET BC=31

Comanda de ștergere a unui program din memorie - NEW

Comanda **NEW** are rolul de a elibera memoria calculatorului de programul curent. După executarea acestei comenzi programul existent în memorie este pierdut.

Comanda de întrerupere a unui program - BREAK

Comanda **BREAK** (**CS** și **SPACE** tastate simultan) are scopul de a întrerupe executarea unui program.

Comenzi pentru salvarea și încărcarea programelor - SAVE și LOAD

Programele pe care dorim să le păstrăm trebuie salvate, cel mai adesea și mai comod, pe casete. În prealabil casetofonul trebuie să fie conectat la calculator și pregătit pentru înregistrare. Comanda necesară salvării programului este:

SAVE "nume"

unde *nume* este numele pe care dorim să-l dăm programului care este depus pe casetă. De exemplu dacă dorim să salvăm un program sub numele de **PROG1** atunci tastăm:

SAVE "PROG1"

și pornim casetofonul pentru înregistrare.

Operația inversă, de încărcare a unui program aflat pe casetă, se face tastând:

LOAD "nume"

unde *nume* este ca mai sus. În acest caz casetofonul trebuie să fie conectat la calculator și pregătit pentru redare. De remarcat că dacă *nume* este șirul vid, deci se tastează:

LOAD ""

atunci se încarcă programele în ordinea în care apar pe casetă.

Ecranul de afișare

Ecranele pe care sînt afișate programele, rezultatele acestora, precum și o serie de comenzi, au o structură precizată, deși acestea pot avea mărimi diferite. Astfel, indiferent de mărimea sa ecranul este împărțit în:

- ▶ 22 linii (pe orizontală), numerotate de la 0 la 21 (de sus în jos);
- ▶ 32 coloane (pe verticală), numerotate de la 0 la 31 (de la stînga la dreapta).

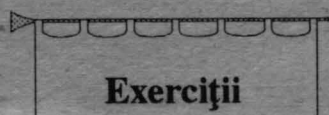
În partea inferioară a ecranului mai sînt două linii (22 și 23) rezervate pentru a se afișa comenzi, date care se tastează la instrucțiunea INPUT, mesaje diverse.

Rețineți că aceste linii (22 și 23) ne sînt inaccesibile, în sensul că nu putem afișa rezultate și nici nu putem lista programe pe aceste linii.

Rezultă astfel că fiecare element afișat pe ecran poate fi identificat prin numărul de linie și cel de coloană. Un element afișat pe ecran îl vom numi **caracter**.

Rețineți !

- **LIST** - listarea programului.
- **RUN** - executarea programului.
- **EDIT** - modificarea liniei curente.
- **NEW** - ștergerea programului din memorie.
- **BREAK** - întreruperea execuției programului.
- **SAVE** - salvarea programului.
- **LOAD** - încărcarea programului.
- Ecranul este împărțit în 22 de linii, numerotate de la 0 la 21 și 32 de coloane, numerotate de la 0 la 31.
- Comenzile și instrucțiunile sînt introduse în aceeași stare, **K**.



Exerciții

1. Scrieți un program cu 5 linii.
2. Listați programul scris la exercițiul 1.
3. Executați programul introdus la exercițiul 1.
4. Modificați programul înlocuind variabila care apare prima într-o instrucțiune de la exercițiul 1 (dacă există), cu alt nume.
5. Rulați programul modificat.
6. Salvați programul inițial pe casetă.
7. Ștergeți programul din memorie și încărcați programul de pe casetă (cel care a fost salvat la exercițiul 6).
8. Fie programul:

```
10 INPUT A
20 PRINT A+1
30 PRINT A+A
```

Faceti următoarele operații cu acest program: a) executați-l; b) modificați în linia 20, + cu -; c) salvați programul modificat sub numele PROGNOU; d) executați programul modificat.

3

Instrucțiunile de citire și scriere. Instrucțiunea STOP

Am văzut în capitolul 1 că instrucțiunile de citire și scriere sînt INPUT și respectiv PRINT.

Instrucțiunea de citire, **INPUT**, are forma:

```
INPUT v1, v2, ..., vn
```

unde v_1, v_2, \dots, v_n sînt zone de memorie ale calculatorului, în care se copiază diferite valori. Numim aceste zone de memorie, **variabile**. Rezultă astfel variabilele v_1, v_2, \dots, v_n . În urma executării acestei instrucțiuni valorile tastate sînt depuse în variabilele din instrucțiunea INPUT.

Introducerea datelor se face tastînd valorile urmate de **CR**. De exemplu pentru a depune în variabilele A, B, C, valorile 10, 20, 30, se execută programul:

```
10 INPUT A, B, C
```

După introducerea comenzii **RUN** programul intră într-o fază de așteptare (pînă la tastarea valorilor numerice ce urmează a fi depuse în variabilele A, B și C). În acest moment se tastează succesiv:

```
10 CR
```

```
20 CR
```

```
30 CR
```

Instrucțiunea opusă acesteia este cea de scriere, care copiază conținutul uneia sau mai multor variabile, constante sau expresii pe ecran.

Instrucțiunea de scriere, **PRINT**, are forma:

```
PRINT e1, e2, ..., en
```

unde e_1, e_2, \dots, e_n sînt variabile, constante sau expresii.

De exemplu, pentru a afișa valorile variabilelor A, B, C, inițializate anterior, se poate completa programul cu:

```
20 PRINT A, B, C
```

În urma executării acestui program (prin acționarea comenzii **RUN**) datele care au fost citite și transferate în variabilele A, B, C, prin instrucțiunea INPUT sînt apoi scrise cu instrucțiunea PRINT.

Instrucțiunea INPUT citește datele care apar pe ecran pe liniile 22 și 23 (în partea de jos a ecranului). Instrucțiunea PRINT afișează datele pe ecran pe una din liniile de la 0 la 21. Afișarea începe de la linia 0.

Exemplul 3.1. Acest program citește valorile unor măsurători în metri și calculează aceste valori în centimetri:

```
P3.1          10 INPUT A
              20 PRINT 100*A
```

Instrucțiunea INPUT (ca și instrucțiunea PRINT) prevede posibilitatea ca valoarea care se citește (care se scrie) să fie precedată de un text care să clarifice modul de introducere al datelor (sau semnificația rezultatelor afișate). Textul care urmează a fi trimis pe ecran prin aceste instrucțiuni este cuprins între simbolurile " și poartă numele de **șir de caractere**.

Exemplul anterior completat cu precizările de mai sus devine:

Exemplul 3.2.

```
P3.2
10 INPUT "Introduceti marimea in metri: ",A
20 PRINT "Valoarea in centimetri este: ",A*100
```

Executînd acest exemplu, mesajele ce însoțesc operațiile de citire și de scriere clarifică execuția programului.

Observații:

◆ Folosind `,` ca delimitator de variabile, datele tastate sau afișate sînt aliniate fie la început de linie, fie la mijlocul acesteia. Dacă utilizăm `;` în loc de `,` observăm că datele apar unele după altele. Sugerăm să modificați exemplul 3.2, substituind `,` cu `;` și apoi să executați programul.

◆ Folosind `;` după ultima expresie dintr-o instrucțiune de scriere, nu se mai trece pe linia următoare la viitoare operație de scriere (a se vedea exemplul 3.5).

Exemplul 3.3. Programul care urmează transformă temperatura exprimată în grade Fahrenheit în grade Celsius prin formula $C = (F - 32) \cdot \frac{5}{9}$

P3.3

```
10 INPUT "Introduceti temp. in grade Fahrenheit:", F
20 PRINT "Temp. in grade Celsius este: ", (F-32)*5/9
```

Exemplul 3.4. Programul care urmează calculează expresiile:

$$a:b+c:d; \quad a:b-c:d; \quad a:b \cdot c:d; \quad (a:b):(c:d),$$

conform cu:

$$(a \cdot d + b \cdot c):b \cdot d; \quad (a \cdot d - b \cdot c):b \cdot d; \quad a \cdot c:b \cdot d; \quad a \cdot d:b \cdot c,$$

presupunînd că sînt îndeplinite condițiile ca aceste fracții să existe (numitorii să fie nenuli).

```
P3.4 10 INPUT "Introduceti fractiile a:b si c:d:
      ";a;b;c;d
      20 PRINT "Suma este: ";a*d+b*c;";";b*d
      30 PRINT "Diferenta este: ";a*d-b*c;";";b*d
      40 PRINT "Produsul este: ";a*c;";";b*d
      50 PRINT "Citul este: ";a*d;";";b*c
```

Următorul exemplu ilustrează posibilitatea de a afișa numere obținute la întâmplare (numite aleatoare), prin utilizarea operației **RND**:

```
10 PRINT RND
```

De menționat că aceste numere sînt cuprinse între 0 și 1. Pentru a tasta RND trebuie să fim în starea **E**.

Instrucțiunea **STOP** are rolul de a opri execuția programului. De exemplu, programul de mai jos:

```
P3.5      10 INPUT A
           20 PRINT A
           30 STOP
           40 INPUT A
```

se execută astfel: se citește în variabila A o valoare numerică, apoi se scrie valoarea citită, după care programul se oprește (instrucțiunea STOP). Instrucțiunea de pe linia 40 nu se mai execută.

Exemplul 3.5. Se citesc trei valori numerice și se scriu pe aceeași linie:

```
P3.6      10 INPUT A
           20 PRINT A;" ";
           30 INPUT A
           40 PRINT A;" ";
           50 INPUT A
           60 PRINT A
           70 STOP
```

Scrierea pe aceeași linie se obține folosind ; după ultima expresie din instrucțiunea PRINT de pe liniile 20 și 40, în cazul de față " ".

Limbajul BASIC prevede posibilitatea de a scrie mai multe instrucțiuni pe aceeași linie, despărțite de simbolul :.

Astfel exemplul 3.5 poate fi transformat în:

Exemplul 3.6.

```
P3.7      10 INPUT A : PRINT A;" ";
          30 INPUT A : PRINT A;" ";
          50 INPUT A : PRINT A
          70 STOP
```

 **Rețineți !**

■ Exemple de instrucțiuni INPUT:

```
INPUT "Introd. un nr."; a
INPUT a; b; c
INPUT a, 'b, c
```

■ Exemple de instrucțiuni PRINT:

```
PRINT a; " este egal cu "; b
PRINT a+b
```

■ Instrucțiunea STOP oprește execuția programului.

■ Instrucțiunile de pe aceeași linie se despart cu :.

■ Generarea numerelor întâmplătoare (aleatoare) cuprinse între 0 și 1 se face utilizând RND.



Exerciții

1. Să se citească în două variabile două mărimi exprimate în kilometri și să se afișeze valorile în metri ale sumei și diferenței celor două mărimi.
2. Să se citească în A și B și în C și D câte două valori exprimate în metri și să se calculeze $A/B+C/D$ în centimetri.
3. Să se citească două valori numerice și să se afișeze produsul acestora.
4. Să se citească patru valori numerice în variabilele A, B, C și D și să se afișeze $A+B$, $A+B+C$, $A+B-C*D$.
5. Să se citească patru valori numerice în variabilele A, B, C și D și să se afișeze $A/C+B/C$ și $A/D-B/D$, presupunând că C și D au valori diferite de zero.
6. Să se citească două valori numerice în A și B și să se afișeze $A+B$, $A-B$, $A*B$ și $A+(B-A)$.
7. Să se citească două valori numerice în A și B și să se afișeze $A+B$, apoi să se citească alte două valori numerice tot în A și B și să se afișeze pe aceeași linie $A-B$, despărțind rezultatele cu simbolul $"/"$.
8. Să se scrie un program care conține 3 instrucțiuni PRINT ce afișează toate rezultatele pe aceeași linie.
9. Să se scrie un program care să scrie trei numere generate întâmplător.

10. Să se scrie un program care citește o valoare numerică în variabila A și afișează suma dintre valoarea conținută în A și un număr generat întâmplător.

11. Să se scrie un program care afișează suma a trei numere generate întâmplător.

12. Să se scrie un program care să transforme în kilometri două numere aleatoare care reprezintă mărimi exprimate în centimetri.

13. Fie programul:

```
10 INPUT A
20 PRINT A+A
30 PRINT A-A
40 STOP
50 PRINT A+RND
60 STOP
```

Să se precizeze ce rezultate afișează și să se explice.

14. Modificați programul de la exercițiul 13 astfel încât să afișeze și valoarea lui A.



4

Constante, variabile numerice, expresii aritmetice. Instrucțiunea de atribuire. Comentariul

Orice limbaj de programare, inclusiv limbajul BASIC, lucrează cu numere, pe care le numim **constante numerice**. Constantele numerice în BASIC pot fi numere **întregi** (de exemplu 1, 2, 1000 etc.) sau **reale** (de exemplu 1.1, 1.2, -7.5).

De menționat că numerele reale se scriu cu "." în loc de ",", și că alături de notațiile de mai sus, constantele numerice pot fi specificate și prin menționarea unei puteri a lui 10 ce înmulțește numărul. De exemplu, notațiile

$$12E+1, 12.12E3, 15.12E-1$$

desemnează $12 \cdot 10^1 = 120$, $12.12 \cdot 10^3 = 12120$ și $15.12 \cdot 10^{-1} = 1.512$.

Exemple:

12, 35, 0, -12, -14, 125 sînt constante întregi, iar
1.2, -1.5, 12.1E-1, -126.125E+2 sînt constante reale.

Variabilele, după cum am văzut în capitolul inițial, desemnează anumite locuri în memoria calculatorului în care se află valori numerice. Variabilele de acest fel se numesc **variabile numerice**, dar pînă vom defini variabile de alt tip, le vom numi pe primele simplu, variabile. Variabilele sînt identificate prin **nume**. Un nume este compus dintr-o succesiune de litere și cifre din care primul caracter este literă.

Exemple de nume ce identifică variabile:

A, B, A100, AB, ION, A0, A1.

Următoarele nume: $A+B$, $1A$, $I?$, $A10_B$ nu identifică variabile deoarece, fie conțin alte simboluri decât cele menționate ($+$, $?$, $_$), fie nu încep cu o literă.

Observație:

◆ Numele de identificatori pot conține atât litere mari cât și litere mici. Nu se face distincție între acestea. De exemplu Ana , ANa , ANA , aNa , reprezintă aceeași variabilă.

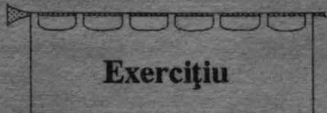
Variabilele și constantele sînt elementele de bază ale **expresiilor**. Expresiile aritmetice pot fi formate din variabile și constante cu ajutorul operațiilor de adunare ($+$), scădere ($-$), înmulțire ($*$), împărțire ($/$) și ridicare la putere (\uparrow). De exemplu $A+B*1.2-3\uparrow 5$ este o expresie aritmetică ce corespunde notației $A+B \cdot 1.2-3^5$.

- Ordinea după care se efectuează operațiile este cea cunoscută din matematică: întîi se efectuează ridicările la putere, apoi înmulțirile și împărțirile, iar la sfîrșit adunările și scăderile. În exemplul de mai sus, ordinea efectuării operațiilor este:

1. $3\uparrow 5$
2. $B*1.2$
3. $A+C$, unde cu C am notat rezultatul de la punctul 2.
4. $D-E$, unde cu D și E am notat rezultatele de la 3 și respectiv 1. Tot expresii pot fi considerate și $3\uparrow 5$, $B*1.5$, constantele și variabilele (3 , 5 , B , A etc).

Exemple de expresii aritmetice:

A , 3 , $3+A$, $3+A*5$, $A+B*C-D\uparrow 2$, $(ION+DAN)*2.0-x1$.



Exercițiu

Să se scrie în limbajul BASIC următoarele expresii matematice:
 $A+B-C$, $A+B \cdot C$, $A-B^2+C^3$, A^2-B^2 .

Am remarcat mai sus că ordinea efectuării operațiilor este identică cu cea cunoscută din matematică. Astfel, știm că pentru a indica anumite calcule ce trebuie să se efectueze înaintea altora, se folosesc parantezele rotunde, drepte, acolade. Acest fapt este valabil și în limbajul BASIC cu remarcă suplimentară că există un singur tip de paranteze, cele **rotunde**: (,).

De exemplu, expresia matematică

$$[(A+1) \cdot B + C] \cdot 3 - 1$$

se transcrie în BASIC prin expresia

$$((A+1) * B + C) * 3 - 1,$$

iar expresia

$$2 \cdot \{ [(A+1)/2 - 3/4] \cdot [A - (B+C) \cdot 4^2] - 3/4 \}$$

se transcrie prin expresia

$$2 * ((A+1) / 2 - 3 / 4) * (A - (B+C) * 4 \uparrow 2) - 3 / 4).$$

Cu ajutorul noțiunilor de **variabilă** și **expresie aritmetică** se construiește o instrucțiune de bază în limbajul BASIC, **instrucțiunea de atribuire**, având forma:

· LET V=E

unde LET desemnează instrucțiunea de atribuire, V este numele unei variabile, iar E este o expresie. Efectul acestei instrucțiuni este următorul: se calculează expresia E și rezultatul este depus în variabila V.

De *exemplu*:

```
LET A=12+13.5  
LET B=A+A*A
```

desemnează instrucțiuni de atribuire.

Observații:

◆ Instrucțiunea de atribuire exprimă puterea de calcul (gamă de operații, precizie) a unui limbaj. Din acest punct de vedere se observă că limbajul BASIC conține operațiile de bază (adunare, scădere, înmulțire, împărțire, ridicare la putere; ulterior vom introduce și radicalul).

◆ Dacă expresiile care apar în instrucțiunea de atribuire, se compun cu ajutorul unor variabile, atunci acestea trebuie să aibă o valoare în momentul execuției.

Exemplu: În momentul executării instrucțiunii

```
30 LET x=a+b
```

trebuiau să se fi executat alte instrucțiuni care să dea valori variabilelor a și b. De exemplu instrucțiunile:

```
10 INPUT a  
20 LET b=10
```

sau

```
10 INPUT a  
20 INPUT b
```

Exemplul 4.1.

Se cere calcularea valorii expresiei

$$E(x, y) = 5 \cdot x \cdot y + x^2 + y^2 - 2 \cdot x + 1,5$$

cu anumite valori specificate ale lui x și y.

P4.1

```

10 INPUT "INTRODUCETI VALORILE LUI X SI Y: "; X;Y
20 LET E=5*X*Y+X↑2+Y↑2-2*X+1.5
30 PRINT "VAL. EXPR. E(X,Y) IN ";X;" SI ";Y;"ESTE ";E

```

Instrucțiunea de atribuire din linia 20 conține expresia $5 * X * Y + X^2 + Y^2 - 2 * X + 1.5$ care se calculează pentru valorile lui X și Y citite de instrucțiunea din linia 10. Rezultatul este depus în variabila E, care este apoi scrisă cu instrucțiunea PRINT de la linia 30.

Instrucțiunea comentariu, REM, are doar efectul unor clarificări în program și nu produce nici un efect la execuție.

Exemplul 4.2.

Programul de mai jos nu are nici un efect la execuție, este un program vid.

```

P4.2 10 REM ACEST PROGRAM NU EXECUTA NIMIC

```

Exemplul 4.3.

```

P4.3 10 REM Acest program calculeaza expresia
      20 REM E(X,Y)=5*X*Y+X+Y pentru valorile
      30 REM X=1 , Y=2
      40 LET X=1 : LET Y=2
      50 LET E=5*X*Y+X+Y
      60 PRINT "VALOAREA EXPRESIEI: "; E

```

Acest program poate fi ușor generalizat înlocuind instrucțiunile de atribuire din linia 40 cu o instrucțiune de citire. Vă propunem această variantă ca exercițiu.

Exemplul 4.4.

Ne propunem să rezolvăm următoarea problemă: trei copii, Ana, Maria și Lucia stau pe trei scaune S1, S2 și S3, în felul următor:

Ana	Maria	Lucia
S1	S2	S3

Cei trei copii se mută ajungând în următoarea situație:

Maria	Lucia	Ana
S1	S2	S3

Care sînt mutările necesare pentru a obține situația de mai sus ?

O variantă ar fi următoarea:

- 1) Ana eliberează scaunul S1 trecind pe un scaun suplimentar S
- 2) Maria trece pe scaunul liber S1
- 3) Lucia trece pe scaunul S2 eliberat de Maria
- 4) Ana trece pe scaunul S3.

Pentru a programa această problemă vom folosi trei variabile S1, S2, S3 care simbolizează scaunele. Cele trei personaje vor fi "codificate" pentru identificare cu 3 numere: Ana cu 1, Maria cu 2 și Lucia cu 3. Vom mai folosi o variabilă suplimentară S care va simboliza scaunul suplimentar pe care trece Ana la pasul 1.

P4.4

```
10 REM Scaunele sint S1, S2 si S3
20 REM Personajele sint Ana=1, Maria=2, Lucia=3
30 REM Ana sta pe S1, Maria pe S2, Lucia pe S3
40 LET S1=1 : LET S2=2 : LET S3=3
50 REM Ana trece pe scaunul suplimentar
60 LET S=S1
70 REM Maria trece pe S1, iar Lucia pe scaunul S2
80 LET S1=S2
90 LET S2=S3
100 REM Ana trece pe scaunul S3
110 LET S3=S
120 PRINT S1; " ";S2; " ";S3
```


Rețineți !

■ Operațiile de adunare, scădere, înmulțire, împărțire, ridicare la putere se reprezintă prin +, -, *, / și respectiv ↑.

■ Numele unei variabile este o succesiune de litere și cifre dintre care prima este literă. Exemple: Ion, I1, IQ25.

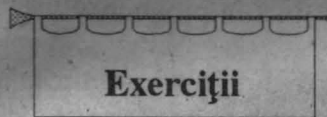
■ Ordinea efectuării operațiilor este cea din matematică: ridicarea la putere, înmulțirea și împărțirea, apoi adunarea și scăderea.

■ Instrucțiunea de atribuire are forma :

LET V=E .

■ Instrucțiunea comentariu are forma :

REM Secvență de caractere .



1. Scrieți un program care să calculeze $1+2\cdot3-4\cdot5+6\cdot7-8\cdot9$ și să afișeze rezultatul.
2. Se dă expresia $E(x)=4\cdot x^3+5\cdot x^2-2\cdot x-1$. Să se calculeze $E(0)$, $E(1)$.
3. Pentru expresia de la exercițiul 2 să se conceapă un program care citind pe x să calculeze pe $E(x)$.
4. Introduceți un comentariu în programul ce rezolvă exercițiul 3 pentru a explica scopul programului.

5. Se dau expresiile

$$E(X,Y)=5\cdot[(X+Y)\cdot 2-X^2]+1$$

și

$$F(X,Y)=2\cdot\{[X+2\cdot(X-Y)-1]\cdot 2+10\}-1$$

Să se scrie expresiile în limbajul BASIC.

6. Se dau expresiile

$$E(X)=X^3+1 \text{ și } F(X)=X\cdot X\cdot X+1$$

- Să se scrie aceste expresii în limbajul BASIC.
- Ce valori iau $E(X)$ și $F(X)$ pentru $X=1$?
- Ce se poate spune despre aceste două expresii atunci când X ia diferite valori?

7. Să se scrie câte un program care să calculeze valorile expresiilor de la exercițiile 5 și respectiv 6 pentru anumite valori ale variabilelor ce intervin în aceste expresii. Valorile variabilelor se specifică în program.

8. Fiind date $E(X,Y)=X+Y+X\cdot Y$ și $F(X,Y)=X-Y-2\cdot X^3$ să se calculeze $E(1,0)$ și $F(1,0)$, adică valorile expresiilor pentru $X=1$ și $Y=0$.

9. Pentru expresiile date în exercițiul 8 să se scrie un program care să calculeze $G(X,Y)=E(X,Y)+F(X,Y)-2$ pentru anumite valori ale lui X și Y citite în program.

10. În două sertare A și B se află 3 caiete și respectiv 3 cărți. Să se scrie un program care să schimbe între ele conținuturile celor două sertare folosind o etajeră.

11. Să se scrie un program care să afișeze valorile variabilelor In , NOI , IOn și VAR care conțin valorile $12E+1$, $1.1E-1$, -3.14 și respectiv $128.123E-2$.

5

Expresii logice. Instrucțiunile IF și GO TO

5.1. Expresii condiționale (condiții)

Am discutat anterior despre expresiile aritmetice, care în urma evaluării (calculării valorii expresiilor pentru anumite valori numerice ale variabilelor și constantelor) conduc la valori numerice. De exemplu, expresia $x \uparrow 2 + 3 * x - a$ are pentru $x=2$ și $a=1$ valoarea 17.

În limbajul BASIC pot fi exprimate pe lângă calcule numerice și anumite **condiții**, de exemplu $x < y$, care înseamnă x mai mic decât y , sau $z \in [1,2]$ care înseamnă că z aparține intervalului $[1,2]$, adică z este mai mare sau egal cu 1 și mai mic sau egal cu 2.

Valoarea unei astfel de condiții poate fi **adevărată**, atunci când valorile variabilelor satisfac condiția (de exemplu pentru $x=3$, $y=4$, $z=1.5$ în cazul de mai sus) sau **falsă**, atunci când nu sînt satisfăcute condițiile (inversați valorile lui x și y specificate mai sus și condiția devine $4 < 3$, adică falsă).

Exprimarea condițiilor se face cu ajutorul următoarelor operații, numite **relaționale**:

<	înseamnă mai mic
>	înseamnă mai mare
=	înseamnă egal
<=	înseamnă mai mic sau egal
>=	înseamnă mai mare sau egal
<>	înseamnă diferit.

Exemple:

Condiții adevărate:

$$1 < 2, 1 <= 2, 1 < > 2, 1 = 1, 1 < = 1, 1 > = 1.$$

Condiții false:

$$3 < 3, 3 < = 2, 3 < > 3, 3 = 4, 3 < = 2, 3 > = 4.$$

Condiția $x <= 2$ este adevărată pentru toate valorile lui x mai mici sau egale cu 2 (de exemplu 2, 1, -1 etc.) și falsă pentru toate valorile lui x mai mari decât 2 (de exemplu 3, 4, 4.5 etc.)

Un caz mai general de condiție este cel în care variabilele sau constantele numerice despărțite de simbolurile operațiilor relaționale sînt înlocuite de expresii aritmetice.

De exemplu:

$$x+1 > 3, x+a+3*b >= 0, a+b < > 2*x-1$$

exprimă condiții. Astfel prima condiție este adevărată atunci cînd valoarea expresiei $x+1$ este mai mare decât 3.

5.2. Expresii logice

Exprimarea matematică " $z \in [1,2]$ " utilizată anterior nu a putut fi transpusă cu ajutorul condițiilor introduse. În continuare vom cunoaște alte operații care ne vor permite să transpunem în limbajul BASIC această expresie matematică. Expresiile condiționale prezentate în paragraful anterior pot fi combinate cu ajutorul operațiilor logice **AND**, **OR** și **NOT** obținînd **expresiile logice**. Expresiile logice la fel ca și cele condiționale pot fi **adevărate** sau **false**.

Două condiții legate cu **AND** formează o expresie logică **adevărată** atunci cînd ambele condiții sînt adevărate. În caz contrar expresia este **falsă**.

De exemplu:

$$x < 2 \text{ AND } x > 0$$

este adevărată atunci când $x < 2$ și $x > 0$ (valori ca 0.5, 1.2, 1.5 satisfac concomitent cele două condiții, deci satisfac și expresia $x < 2 \text{ AND } x > 0$). Matematic această exprimare poate fi scrisă astfel: $x \in [0, 2]$.

Aveți în acest moment posibilitatea de a transcrie în limbajul BASIC expresia matematică $z \in [1, 2]$.

Două condiții legate cu OR formează o expresie logică **adevărată** atunci când cel puțin una din cele două condiții este adevărată. În caz contrar, adică atunci când ambele condiții sînt false, expresia logică este **falsă**.

O condiție precedată de NOT este o expresie logică **adevărată** atunci când condiția este **falsă**.

De exemplu:

$$\text{NOT } x < 3$$

este adevărată atunci când $x \geq 3$. Evident acest exemplu poate fi scris cu ajutorul condiției

$$x \geq 3.$$

Operațiile AND, OR, NOT pot lega nu numai condiții, dar chiar și expresii logice.

Exemplul 5.1. Expresia logică

$$x < 3 \text{ AND } y > 0 \text{ OR } z + 1 < -1$$

se evaluează astfel:

- ▶ întâi se determină condițiile $x < 3$, $y > 0$, $z + 1 < > - 1$;
- ▶ apoi se determină expresia logică $x < 3 \text{ AND } y > 0$, care este adevărată dacă $x < 3$ și $y > 0$ sînt adevărate simultan;
- ▶ ultima expresie determinată este cea indicată prin operația OR și este adevărată dacă expresia logică determinată anterior este adevărată sau condiția $z + 1 < > - 1$ este adevărată.

Observații:

- ◆ Dacă într-o expresie logică apar mai multe operații logice atunci se efectuează mai întâi operațiile NOT, apoi AND și la sfîrșit OR.
- ◆ În orice expresie logică înaintea operațiilor NOT, AND, OR se efectuează operațiile relaționale ce definesc expresiile condiționale.
- ◆ Dacă în condiții apar expresii aritmetice, acestea din urmă se efectuează primele.

Exemplul 5.2. Fie expresia logică

$$x + 3 < a + b - 1 \text{ AND } 2 * x > = y + 1 .$$

Întîi se calculează expresiile aritmetice $x + 3$, $a + b - 1$, $2 * x$, $y + 1$, apoi se efectuează operațiile relaționale ce definesc condițiile $x + 3 < a + b - 1$, $2 * x > = y + 1$. Ultima operație efectuată este operația logică AND.

Observație:

- ◆ În anumite variante ale limbajului BASIC este prezentă o operație logică asemănătoare operației OR, notată de obicei XOR și numită **sau exclusiv**. Două condiții legate cu XOR furnizează o expresie logică adevărată atunci cînd o condiție este adevărată, iar cealaltă este falsă.

5.3. Instrucțiunea IF

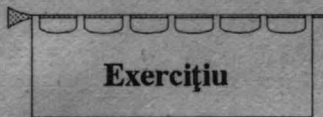
Instrucțiunea **IF** definește o secvență de instrucțiuni care sînt executate numai dac  sînt îndeplinite (adev rate) anumite condiții.

Exemplul 5.3. Următorul program calculeaz  suma a dou  numere pozitive care sînt citite la execuția sa:

P5.1

```
10 REM aduna doua numere pozitive
20 INPUT "Introduceti numerele ";a;b
30 IF a>0 AND b>0 THEN PRINT a+b: STOP
40 PRINT "Nu sint satisfacute conditiile initiale"
```

Instrucțiunile PRINT a+b și STOP sînt executate atunci c nd expresia logic  a>0 AND b>0 este adev rat .



Rescrieți acest exemplu utiliz nd în linia 30 o expresie logic  format  cu ajutorul operației OR.

Forma instrucțiunii IF este următoarea:

```
IF exp_log THEN S1: S2...:Sn
```

unde exp_log indic  o expresie logic  (în particular o condiție), iar S1, S2, ..., Sn, instrucțiunile care se execut  atunci c nd exp_log are valoarea adev rat . Dac  exp_log are valoarea fals  atunci instrucțiunile S1, S2, ..., Sn nu sînt executate și prima instrucțiune care se execut  este cea care urmeaz  dup  instrucțiunea IF.

În cazul exemplului anterior dacă expresia logică $a > 0$ AND $b > 0$ este falsă atunci se execută instrucțiunea de pe linia 40.

5.4. Instrucțiunea GO TO

Instrucțiunea GO TO are ca efect întreruperea execuției secvențiale a programului și transferarea execuției la o altă linie. De exemplu instrucțiunea:

```
10 GO TO 40
```

trece controlul la instrucțiunea de la linia 40 care urmează să fie executată.

Prin urmare instrucțiunea GO TO are forma următoare

```
GO TO nr
```

unde nr indică un număr de linie din program.

Exemplul 5.4.

P5.2

```
10 REM programul afiseaza val. 1 (a=1 in linia 20)
20 LET a=1
30 GO TO 50
40 LET a=2
50 PRINT a
```

Acest program se execută astfel: i se atribuie lui a valoarea 1 și apoi se merge la linia 50 (prin instrucțiunea GO TO) unde se afișează valoarea lui a (egală cu 1). Instrucțiunea de pe linia 40 nu este executată datorită instrucțiunii anterioare, GO TO 50.

Exemplul 5.5. Acest program citește două valori pozitive și apoi calculează suma acestora:

P5.3

```
10 INPUT "Introduceti prima valoare ";a
20 IF a<=0 THEN PRINT "Eroare: val.neg." : GO TO 10
30 INPUT "Introduceti a doua valoare ";b
40 IF b<=0 THEN PRINT "Eroare: val.neg." : GO TO 30
50 REM In acest moment in a si b se afla val. poz.
60 PRINT "Suma este: ";a+b
```

Exemplul 5.6. Ne propunem să calculăm suma

$$1+2+3+4+5+6+7+8+9+10$$

în câteva variante:

Varianta 1.

```
P5.4      10 PRINT 1+2+3+4+5+6+7+8+9+10
```

Varianta 2.

```
P5.5      10 LET a=1+2+3+4+5+6+7+8+9+10
          20 PRINT a
```

Varianta 3.

```
P5.6      10 LET a=0
          20 LET s=0
          30 LET a=a+1
          40 LET s=s+a
          50 IF a<10 THEN GO TO 30
          60 PRINT s
```

Varianta 4.

```
P5.7      10 LET a=1
          20 LET s=0
          30 IF a>10 THEN GO TO 70
```

```
40 LET s=s+a
50 LET a=a+1
60 GO TO 30
70 PRINT s
```

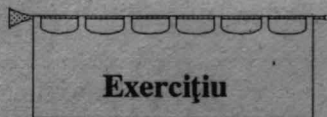
Primele două variante sînt simple, ultimele două sînt puțin mai complicate, deci necesită cîteva explicații. Vom comenta varianta 3, lăsînd ca pe ultima să o înțelegeți singuri.

- ▶ Suma dată se va calcula în s , care inițial are valoarea 0.
- ▶ Instrucțiunile de pe liniile 30 și 40 măresc pe a cu 1 (pentru a lua succesiv valorile 1, 2, 3, ..., 10) iar pe s cu a (luînd succesiv valorile 0, 1, 3, 7, 12 etc.).

Pentru a înțelege mai exact cum evoluează valorile variabilelor a și s vă propunem să introduceți în exemplul anterior instrucțiunea:

```
45 PRINT a, s
```

care ne va indica toate valorile intermediare ale variabilelor a și s .

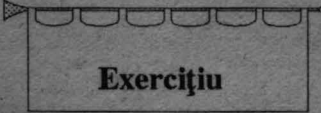


Vă propunem să modificați în varianta 3 liniile 10 și 20 astfel:

```
10 LET a=1
20 LET s=1
```

Care este rezultatul afișat? Explicați rezultatul.

Observație: Variantele 3 și 4, deși mai complicate, pot fi ușor adaptate pentru a calcula sume cu mai mulți termeni. Acest fapt nu este valabil și pentru primele două variante.



Exercițiu

Propunem modificarea programului din varianta 3 sau din varianta 4 pentru a calcula suma $1+2+3+\dots+100$.

Rețineți !

■ Operatorii relaționali care definesc expresiile condiționale sînt:
 $<$, $>$, $=$, $<=$, $>=$, $<>$.

■ Operatorii logici care definesc expresiile logice sînt NOT, AND, OR.

■ Ordinea de efectuare a operațiilor este următoarea:

- ▶ se efectuează operațiile aritmetice (după ordinea cunoscută din matematică)
- ▶ se efectuează apoi operațiile relaționale
- ▶ ultimele se efectuează cele logice în ordinea NOT, AND, OR.

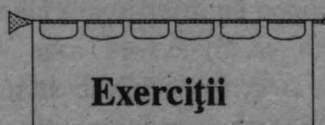
■ Instrucțiunea IF are forma

```
IF exp_log THEN S1: S2: ... :Sn
```

■ Instrucțiunea GO TO are forma

```
GO TO nr
```

unde nr indică numărul unei linii de program.



Exerciții

1. Menționați ce valori de adevăr au condițiile

$$-1=2, 3 < > 4, 1 < = 2, 2 > = 3, 10 < 11, 11 > 10.$$

2. Calculați valoarea condiției $x+1 < 3*a$, pentru următoarele valori ale lui x și a :

$$a) x=3, a=2; b) x=2, a=3; c) x=-1, a=1; d) x=0, a=0.$$

3. Determinați trei valori ale lui x pentru care condiția $x+3 > 10$ este adevărată. Aceeași problemă pentru $3*x < 10$, $x+x < = x$, $x+2*x < > 0$.

4. Fie expresiile logice:

$$(1) a+b > 3 \text{ AND } a-b < 1$$

$$(2) a < = 3-b \text{ OR } a > = b+1.$$

Pentru care din valorile de mai jos sînt adevărate expresiile logice:

$$a) a=5, b=3; b) a=3, b=5; c) a=10, b=-1; d) a=6, b=5; \\ e) a=0, b=-2; f) a=20, b=-20?$$

5. Fie expresiile logice:

$$1. a < 0 \text{ OR } a > = 0;$$

$$2. a < 0 \text{ AND } a > = 0;$$

$$3. \text{NOT } (a < 0 \text{ OR } a > = 0);$$

$$4. a < = 1 \text{ AND } a > -1;$$

$$5. a < 1 \text{ AND } a > 0 \text{ OR } a < 3 \text{ AND } a > 2;$$

$$6. a < 1 \text{ AND } a < 2 \text{ AND } a < 3.$$

Să se specifice pentru ce valori ale lui a sînt adevărate aceste expresii logice.

6. Fie expresia

$$a > -1 \text{ AND } a < 3 \text{ OR } a > 4 \text{ AND } a < 10.$$

Pentru care din valorile de mai jos ale lui a este ea adevărată?

a) $a=20$; b) $a=10$; c) $a=0$; d) $a=2$; e) $a=7$; f) $a=8$.

7. Fie programul:

```

10 LET i=0
20 LET s=0
30 LET i=i+2
40 IF i<=10 THEN LET s=s+i: GO TO 30
50 PRINT s

```

Ce valoare este afișată pentru s ? Explicați rezultatul.

8. Modificați programul de la exercițiul 7 înlocuind linia 10 cu

```
10 LET i=1
```

Ce valoare este afișată pentru s ? Explicați rezultatul.

9. Să se scrie un program care să calculeze suma

$$1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + 4 \cdot 5 + 5 \cdot 6 + 6 \cdot 7 + 7 \cdot 8.$$

10. Să se scrie un program pentru calcularea expresiei:

$$1 - 2 + 3 - 4 + 5 - 6 + 7 - 8 + 9 - 10.$$

11. Să se scrie un program care să citească o valoare n și să calculeze suma $1 - 1 + 1 - 1 + 1 - 1 \dots$, cu n apariții pentru 1.

12. Fie programul

```
10 INPUT A;B;C
20 IF A>= 12.E+1 AND b<2.98E-1 THEN PRINT A+B
30 PRINT A;B;C
40 STOP
```

Dacă la execuția programului se introduc pentru A, B, C, valorile 1000, 0.1 și respectiv 50, atunci precizați ce rezultate sînt afișate. Explicați rezultatele obținute.

13. Să se modifice programul de la exercițiul precedent înlocuind, în linia 20, AND cu OR. Precizați cîteva valori pentru variabilele ce intervin în condiția din linia 20 pentru care instrucțiunea PRINT A+B nu se mai execută.

14. Fie programul

```
10 LET A=1 : LET B=2
20 GO TO 50
30 PRINT A; B
40 STOP
50 GO TO 70
60 PRINT A+B
70 GO TO 30
80 STOP
```

Precizați secvența de instrucțiuni care se execută și rezultatul care se afișează. Scrieți un program mai simplu ca cel de sus care să aibă același efect.



6

Scheme logice

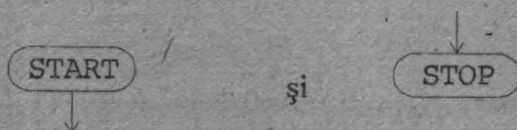
O problemă mai complexă necesită pentru a fi transpusă în program anumite explicații, o prezentare care să se detașeze de limbajul de programare. O astfel de prezentare se poate face în limbajul natural, așa cum am procedat pînă acum, dar acest mod de prezentare deși ușor de urmărit are dezavantajul lipsei de precizie.

Introducem de aceea un „limbaj” mai adecvat de prezentare a rezolvării unei probleme din care programul să poată fi dedus relativ ușor. Acest „limbaj” format din simboluri grafice face obiectul prezentării care urmează. Un astfel de „limbaj” îl vom numi **schemă logică**.

Elementele care compun o schemă logică sînt:

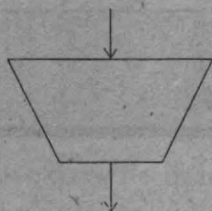
- a) blocurile de început și sfîrșit
- b) blocurile de citire și scriere
- c) blocul de calcul
- d) blocul de decizie.

a) **Blocurile de început și sfîrșit** se pun la începutul și respectiv la sfîrșitul unei scheme logice și au formele

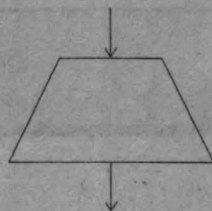


b) **Blocurile de citire și scriere** se folosesc pentru a specifica operații de citire și respectiv de scriere avînd formele:

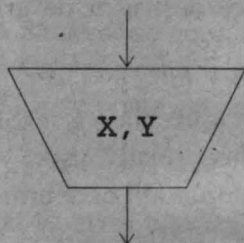
Bloc de citire



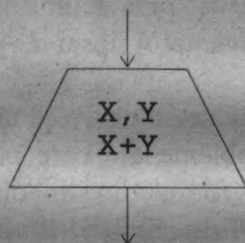
Bloc de scriere



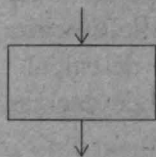
De exemplu pentru a indica faptul că se citește variabilele X și Y și se scriu aceleași variabile și suma acestora $X+Y$, folosim următoarele blocuri



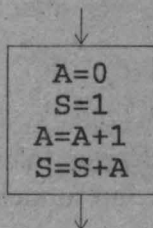
și



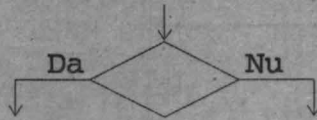
c) **Blocul de calcul** desemnează operații de calcul de tipul atribuirilor și are forma



De exemplu, pentru a indica atribuirile din exemplul 5.6 utilizăm următorul bloc de calcul:

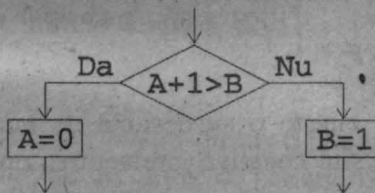


d) **Blocul de decizie** desemnează o condiție sau o expresie logică (cu valoare adevărat sau fals) și alternativele pentru cele două valori ale expresiei. Forma acestui bloc este:



unde alternativa indicată prin DA se alege atunci când pentru valorile asociate variabilelor expresia este adevărată, iar cealaltă atunci când expresia este falsă.

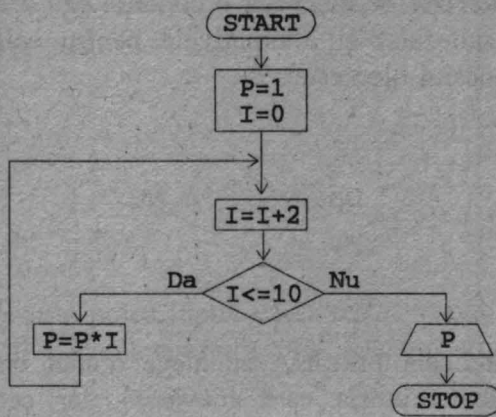
De exemplu, pentru a exprima faptul că în cazul în care expresia $A+1 > B$ este adevărată se execută $A=0$, iar în caz contrar se execută $B=1$, folosim:



în care am utilizat un bloc de decizie și două blocuri de calcul.

Vom ilustra compunerea blocurilor introduse pentru a forma o schemă logică prin

Exemplul 6.1. Pentru a calcula produsul $P=2 \cdot 4 \cdot 6 \cdot 8 \cdot 10$, utilizăm următoarea schemă logică:

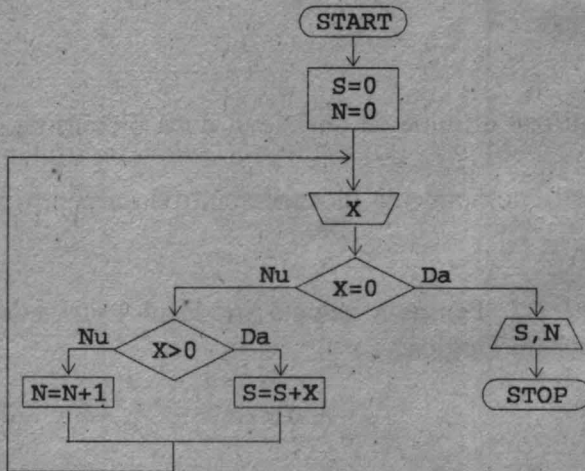


Din schema logică de mai sus putem deduce următorul program în limbaj BASIC:

```

P6.1   10 LET P=1
          20 LET I=0
          30 LET I=I+2
          40 IF I<=10 THEN LET P=P*I: GO TO 30
          50 PRINT P
  
```

Exemplul 6.2. Se citește o succesiune de numere și se adună cele pozitive, se numără câte sînt negative, prelucrarea oprindu-se atunci cînd se introduce 0.



Programul BASIC care se deduce din schema logică anterioară:

```
P6.2   10 S=0 : N=0
        20 INPUT "Introduceți valoarea: ";X
        30 IF X=0 THEN PRINT S,N: STOP
        40 IF X>0 THEN LET S=S+X: GO TO 20
        50 LET N=N+1: GO TO 20
```

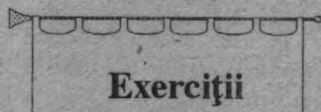
Rețineți !

■ Blocurile ce alcătuiesc o schemă logică sînt:

- ▶ blocul de început/sfîrșit
- ▶ blocul de citire/scriere
- ▶ blocul de calcul
- ▶ blocul de decizie

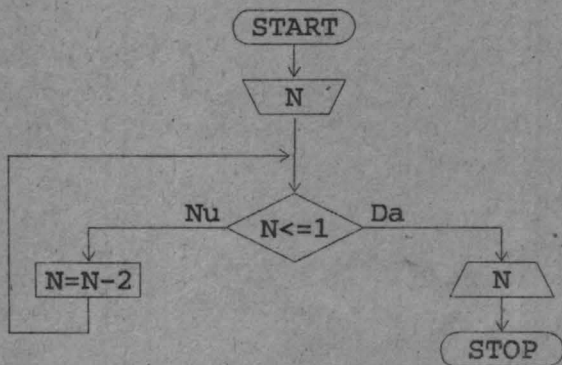
■ Pentru blocurile descrise facem următoarele precizări:

- ▶ în fiecare bloc intră o singură săgeată, excepție face blocul de început
- ▶ din fiecare bloc, fie poate pleca o singură săgeată (în cazul blocurilor de început, de citire, de scriere, de calcul), fie pot pleca două săgeți (din blocul de decizie), fie nu pleacă nici o săgeată (din blocul de sfîrșit).



1. Să se facă schema logică pentru a rezolva problema calculării sumei
 $1+3+5+7+9+11$.

2. Să se facă schema logică pentru calcularea următoarei expresii
 $(2+4+6+8+10) / (1+3+5+7+9+11)$.
3. Să se facă schemele logice pentru a rezolva exercițiile 9 și 10 de la capitolul 5.
4. Să se conceapă schema logică pentru a rezolva problema determinării numărului de elemente mai mari decât o valoare specificată. Elementele se citesc succesiv.
5. Să se conceapă schema logică și programul pentru calcularea sumei
 $1^2+2^2+\dots+8^2$.
6. Să se facă o schemă logică pentru a decide dacă un număr întreg se află între două valori A și B (condiția inițială este $A \leq B$).
7. Se dă schema logică



- a) Ce problemă rezolvă această schemă logică?
- b) Să se scrie programul în limbaj BASIC.
- c) Găsiți o generalizare a acestei probleme.

Instrucțiunea FOR...NEXT. Valoarea absolută, partea întreagă și radicalul (ABS, INT, SQR)

7.1. Instrucțiunea FOR...NEXT

Reamintim programul prezentat în exemplul 5.6, varianta 3, care calculează suma $1+2+3+\dots+10$:

```

10 LET a=0
20 LET s=0
30 LET a=a+1
40 LET s=s+a
50 IF a<10 THEN GO TO 30
60 PRINT s

```

Remarcăm faptul că instrucțiunile de pe liniile 30, 40, 50 se execută de 10 ori, iar variabila a ia succesiv valori întregi cuprinse între 1 și 10. Dintre aceste trei instrucțiuni, ultima, adică instrucțiunea de pe linia 50, are doar rolul de a controla numărul de execuții al grupului de instrucțiuni de pe liniile 30 și 40.

Deoarece asemenea probleme, în care o variabilă ia valori succesive pînă la un anumit prag, apar foarte des, limbajul BASIC prevede o instrucțiune specială numită **FOR...NEXT**, care permite executarea repetată a unui grup de instrucțiuni. O astfel de instrucțiune se numește **instrucțiune de ciclare (iterare)**.

Rescriem exemplul de mai sus utilizînd această instrucțiune.

Exemplul 7.1.

```
P7.1          10 LET s=0
              20 FOR a=1 TO 10
              30   LET s=s+a
              40 NEXT a
              50 PRINT s
```

Observăm că în exemplul 7.1 instrucțiunea FOR...NEXT conține o singură instrucțiune, cea de pe linia 30, care se execută de 10 ori, iar a ia succesiv valorile de la 1 la 10.

Putem deci scrie că "evoluția" instrucțiunii 30 în timpul execuției programului este următoarea:

```
s=s+1      (inițial s a fost 0, deci s=0+1)
s=s+2      (s=0+1+2)
s=s+3      (s=0+1+2+3)
s=s+4      (etc.)
s=s+5
s=s+6
s=s+7
s=s+8
s=s+9
s=s+10
```

deci după cei 10 pași rezultatul cuprins în s va fi $s=0+1+2+3+\dots+10$.

Exemplul 7.2. Ne propunem să calculăm valoarea

$$r=1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 / (1+2+3+4+5+6).$$

Programul în limbajul BASIC este următorul:

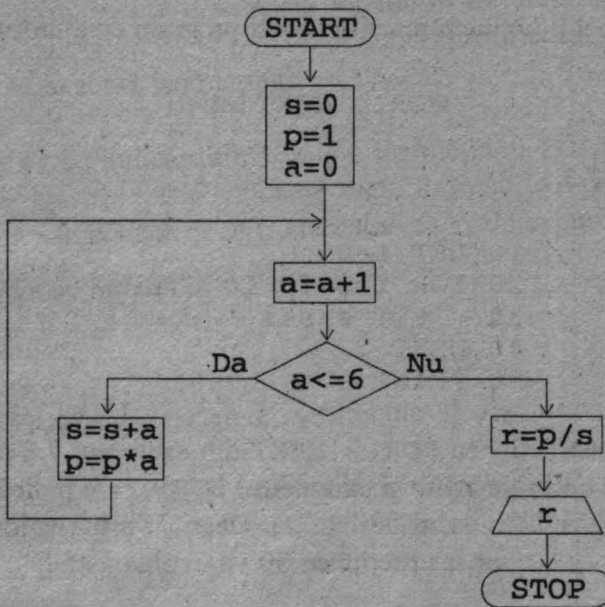
```
P7.2          10 LET s=0
              20 LET p=1
              30 FOR a=1 TO 6
              40   LET s=s+a
```

```

50 LET p=p*a
60 NEXT a
70 LET r=p/s
80 PRINT r

```

O schemă logică pentru rezolvarea problemei este următoarea:



Propunem ca **exercițiu** rescrierea acestui program fără a utiliza instrucțiunea FOR...NEXT.

În exercițiul 7 din capitolul 5 a fost propus următorul program:

```

10 LET i=0
20 LET s=0
30 LET i=i+2
40 IF i<=10 THEN LET s=s+i: GO TO 30
50 PRINT s

```

care execută instrucțiunile 30, 40 pentru valorile lui i egale cu 2, 4, 6, 8, 10, iar pentru valoarea lui i egală cu 12 nu mai execută instrucțiunile care se află în instrucțiunea IF după THEN.

Deducem astfel că acest program calculează suma

$$s=0+2+4+6+8+10.$$

Același efect îl obținem rescriind acest program cu ajutorul instrucțiunii

FOR...NEXT astfel:

Exemplul 7.3.

```
P7.3      10 LET s=0
          20 FOR i=2 TO 10 STEP 2
          30   LET s=s+i
          40 NEXT i
          50 PRINT s
```

Observăm că instrucțiunea FOR...NEXT din exemplul 7.3 are un element nou; în linia de definiție apare și parametrul STEP, care indică valoarea care mărește pe i , pasul (în cazul de față 2). Deci i va lua valorile succesive: 2, 4, 6, 8, 10, iar instrucțiunea 30 va evolua astfel:

```
s=s+2      (s=0+2)
s=s+4      (s=0+2+4)
s=s+6      (etc.)
s=s+8
s=s+10
```

deci în s se va memora suma $2+4+6+8+10$.

Din exemplele prezentate deducem că forma generală a acestei instrucțiuni este:


```

FOR v=vi TO vf STEP p
    S1
    S2
    ...
    Sn
NEXT v

```

unde v_i și v_f sînt expresii aritmetice ce desemnează **valoarea inițială** și respectiv **valoarea finală** între care se desfășoară execuția acestei instrucțiuni. Prin v este desemnată **variabila de ciclare** care ia succesiv valorile $v_i, v_i+p, v_i+2p, \dots, v_f$. Pentru fiecare din valorile lui v de mai sus se execută instrucțiunile S_1, S_2, \dots, S_n .

Prin NEXT se indică ultima linie a instrucțiunii FOR...NEXT. Prin p se definește **pasul**, adică valoarea care îl modifică pe v . Dacă STEP lipsește atunci se consideră că p are valoarea 1, după cum se poate observa și din exemplele 7.1, 7.2.

Observații:

- ◆ Variabila de ciclare, v , trebuie să fie formată dintr-o singură literă.
- ◆ Pasul, p , poate fi orice valoare număr real.
- ◆ Condiția $v_i \leq v_f$ nu este obligatorie după cum rezultă și din exemplul de mai jos:

```

P7.4      10 LET k=0
          20 FOR i=10 TO 1 STEP -1
          30   LET k=k+i
          40 NEXT i
          50 PRINT k

```

care calculează $k=10+9+8+\dots+1$.

◆ Instrucțiunile S_1, S_2, \dots, S_n din interiorul instrucțiunii FOR...NEXT pot fi instrucțiuni FOR...NEXT, în acest caz acestea numindu-se instrucțiuni FOR...NEXT **interioare**. De *exemplu*:

P7.5

```
10 REM Acest program ilustreaza utilizarea
20 REM unei instructiuni FOR...NEXT interioare
30 FOR i=1 TO 2
40   FOR j=1 TO 2
50     PRINT "  j=";j
60   NEXT j
70   PRINT "i=";i
80 NEXT i
```

Instrucțiunile 40-60 definesc o instrucțiune FOR...NEXT interioară celei definite de liniile 30-80.

◆ Deși rezultă din definiția instrucțiunii FOR...NEXT, subliniem totuși faptul că două instrucțiuni FOR...NEXT nu pot fi decât separate sau una interioară celeilalte. Cazul unei "acoperiri parțiale", ca în exemplul care urmează, este **greșit**:

P7.6

```
10 REM utilizare gresita a instructiunilor
FOR...NEXT
20 FOR i=1 TO 2
30 FOR j=1 TO 2
40 PRINT "i="; i
50 NEXT i
60 PRINT "j="; j
70 NEXT j
```

Exemplul 7.4. Următorul program afișează pe ecran valorile 10, 9, 8, 7, 6, 5, 4, fiecare pe câte o linie.

P7.7

```
10 FOR i=10 TO 4 STEP -1
20   PRINT i
30 NEXT i
40 STOP
```

În cazul în care $p > 0$ vorbim de **incrementarea** valorii variabilei de ciclare, iar în cazul în care $p < 0$ spunem că variabila de ciclare este **decrementată**.

Dacă $p=0$ atunci numărul de execuții al instrucțiunilor ce compun o instrucțiune FOR...NEXT este nelimitat.

Exemplul 7.5.

P7.8

```
10 REM Programul executa la nesfirsit instructiunile
20 REM din liniile 40-50. Se opreste cu BREAK.
30 FOR m=1 TO 2 STEP 0
40   INPUT a
50   PRINT a
60 NEXT m
```

7.2. Valoarea absolută, partea întreagă și radicalul - ABS, INT, SQR

Cunoaștem din matematică faptul că **valoarea absolută** sau **modulul** unui număr a este notat cu $|a|$ și definit prin: $|a|=a$ pentru $a \geq 0$, iar $|a|=-a$, pentru $a < 0$.

În limbajul BASIC, dacă a este o variabilă ce conține o anumită valoare atunci prin

ABS a

vom indica modulul valorii ce se află în variabila a .

Exemplul 7.6.

```
P7.9      10 INPUT "Introduceti a=";a
          20 PRINT "|a|=";ABS a
```

Dacă la execuția acestui program pentru instrucțiunea INPUT se tastează 10 atunci pe ecran se afișează valoarea 10, iar dacă se tastează -10, se afișează tot valoarea 10.

pozitivă, atunci prin

SQR a

este desemnată rădăcina pătrată din valoarea conținută în a, adică numărul pozitiv b, cu proprietatea $b^2=a$.

Exemple: SQR 4 = 2, SQR 0.25 = 0.5.

Observații:

◆ Pentru a tasta ABS, INT și SQR se procedează astfel:

▶ se trece în modul E tastând simultan CS (CAPS SHIFT) și SS (SYMBOL SHIFT);

▶ pentru a obține ABS se apasă pe G, pentru a obține INT se apasă pe R, iar pentru a obține SQR se apasă pe H.

◆ ABS, INT și SQR intră în componența expresiilor aritmetice și în acest caz trebuie să se știe care este ordinea efectuării operațiilor.



IMPORTANT: ABS, INT, SQR se calculează înaintea celorlalte operații aritmetice. Pentru a schimba ordinea efectuării operațiilor se utilizează parantezele așa cum am procedat în exemplul 7.7.

Expresia

INT x + y

se efectuează astfel: se calculează partea întreagă a lui x și apoi se adună cu y. Același lucru este valabil și pentru expresia

y + INT x .

Expresia

$$\text{INT} (x + y)$$

se efectuează astfel: se calculează întâi $x + y$ și apoi se calculează partea întregă a rezultatului.

Operația INT poate fi folosită împreună cu RND pentru a obține numere aleatoare (arbitrare) întregi mai mici decât un număr dat. De exemplu, pentru a obține numere aleatoare întregi mai mici ca 6 utilizăm

$$\text{INT} (\text{RND} * 6)$$

7.3. Probleme de divizibilitate

Exemplul 7.8. Dându-se un număr natural n să se determine dacă este prim sau nu.

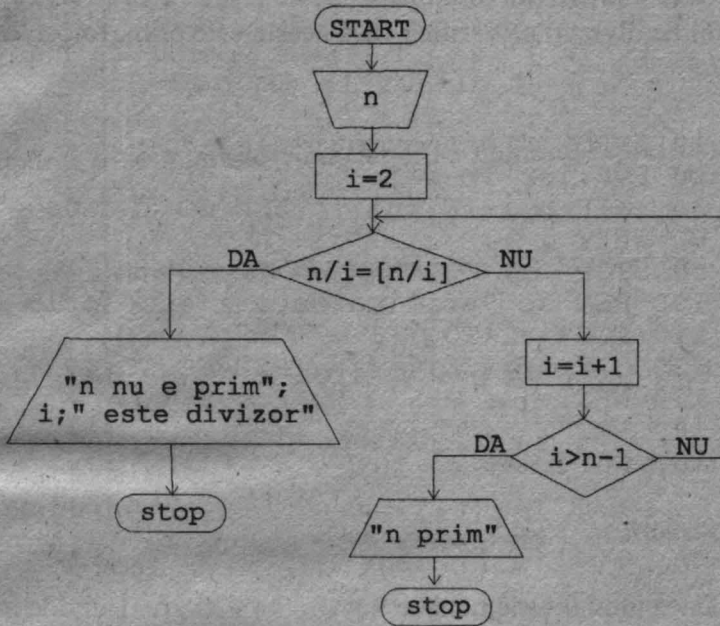
Reamintim că un număr natural n este prim dacă are ca singuri divizori pozitivi pe 1 și pe n . De exemplu

$$2, 3, 5, 7, 11$$

sînt numere prime.

Vom determina că n este prim, verificînd că nici unul din numerele $2, 3, 4, \dots, n-1$ nu divide pe n .

Propunem următoarea schemă logică pentru rezolvarea problemei:



Din această schemă logică deducem următorul program:

P7.11

```

10 REM Program pentru a determina daca n este prim
20 INPUT "Introduceti n>=3 ";n
30 FOR i=2 TO n-1
40   IF n/i=INT(n/i) THEN PRINT i;" divide ";n:STOP
50 NEXT i
60 PRINT n;" prim"
70 STOP
  
```

Vă propunem să găsiți un algoritm mai bun decât cel de sus, care să testeze divizorii pînă la $[n/2]$ și numai pentru numere impare. Încercați să demonstrați că aceste condiții sînt suficiente pentru determinarea calității de număr prim.

Exemplul 7.9. Să se afișeze toți divizorii primi ai unui număr natural n . Metoda de rezolvare constă în următoarele: se verifică dacă n admite divizori numere din mulțimea $\{2, 3, 4, \dots, n-1\}$, apoi pentru fiecare astfel de divizori se verifică dacă acesta este prim, folosind ideea din exemplul 7.8.

```
P7.12  10 INPUT "Introduceti n>=3 ";n
        20 FOR i=2 TO n-1
        30   IF n/i=INT(n/i) THEN GO TO 100
        40 NEXT i
        50 STOP
        100 REM se verifica daca i este prim
        110 FOR j=2 TO i-1
        120   IF i/j=INT(i/j) THEN GO TO 40
        130 NEXT j
        140 PRINT i;" ";
        150 GO TO 40
```

Observații:

◆ Dacă unul din elementele $2, 3, 4, \dots, n-1$ divide pe n atunci se merge la linia 100 unde începe secvența de instrucțiuni ce verifică dacă divizorul lui n este prim.

◆ După verificarea calității de număr prim a divizorului i se revine la instrucțiunea FOR...NEXT cu GO TO 40.

Exemplul 7.10. Să se determine numerele formate din trei cifre abc care satisfac relația $a^2+b^2+c^2=a+b+c$.

```
P7.13  10 FOR a=1 TO 9
        20   FOR b=0 TO 9
        30     FOR c=0 TO 9
        40       IF a*a+b*b+c*c=a+b+c THEN PRINT a;b;c
        50     NEXT c
        60   NEXT b
        70 NEXT a
```


Exemplul 7.11. Este știut din matematică următorul rezultat, cunoscut sub numele de **teorema de împărțire cu rest**: "pentru orice două numere întregi a , b cu $b \neq 0$, există două numere întregi q și r care satisfac relațiile:

$$a = bq + r \text{ și } 0 \leq r < |b|$$

Numărul q se numește *cîtul*, iar r se numește *restul împărțirii* lui a la b .

Programul care urmează determină cîtul și restul împărțirii lui $|a|$ la $|b|$:

P7.14

```
10 INPUT "Introduceti cele doua numere : ";a;b
20 IF b=0 THEN GO TO 10
30 LET a=ABS a : LET b=ABS b
40 REM determinam in q citul, iar in r restul
50 LET q=INT(a/b)
60 LET r=a-q*b
70 PRINT "citul :";q;" restul: ";r
80 STOP
```

Rețineți !

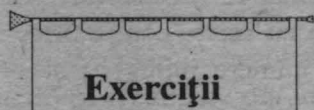
■ Instrucțiunea FOR...NEXT are forma

```
FOR v=vi TO vf STEP p
    S1
    S2
    ...
    Sn
NEXT i
```

■ Instrucțiunile FOR...NEXT nu se pot suprapune parțial.

■ Variabila de ciclare este formată dintr-o singură literă.

■ Determinarea părții întregi, a valorii absolute și a radicalului de ordin doi se face cu INT, ABS și respectiv SQR. Aceste operații sînt prioritare celor matematice uzuale (+, -, *, /, ^).



Exerciții

1. Să se rezolve exercițiile 9 și 10 de la capitolul 5 utilizând instrucțiunea FOR...NEXT.

2. Să se rezolve exercițiile 4 și 5 de la capitolul 6 utilizând instrucțiunea FOR...NEXT.

3. Pentru calcularea sumei

$$1 + 1/(1*2) + 1/(1*2*3) + \dots + 1/(1*2*\dots*10)$$

să se întocmească o schemă logică și un program. Aceeași problemă pentru suma

$$1 + 1/(1+2) + 1/(1+2+3) + \dots + 1/(1+2+\dots+10)$$

Generalizare. Modificați programul pentru a calcula

$$1 + 1/(1*2) + \dots + 1/(1*2*\dots*n),$$

respectiv

$$1 + 1/(1+2) + \dots + 1/(1+2+\dots+n),$$

unde n este un număr natural mai mare sau egal cu 1.

4. Să se completeze programul prezentat în exemplul 7.5 astfel încât instrucțiunile de pe liniile 40 și 50 să se execute de 5 ori.

5. Să se calculeze $\text{INT } x$ și $\text{ABS } x$ pentru x luând valorile 2.4, 2.9, 10.10, -0.1, -0.5, -1.1, -10. Scrieți un program care să verifice răspunsurile date.

6. Să se găsească perechile de numere naturale a, b care satisfac relațiile
 $a+b=1000$, 17 divide a , 19 divide b .

Indicație. Se iau pentru a toate valorile cuprinse între 1 și 999 și se verifică următoarea condiție: 17 divide a și 19 divide $(1000-a)$.

7. Să se genereze toate numerele de 4 cifre de forma $3a2b$ care se divid cu 9.

Indicație. Se consideră toate valorile lui a și b cuprinse între 0 și 9 și se verifică următoarea condiție

$$9 \text{ divide } (3+a+2+b).$$

8. Să se determine numărul numerelor naturale prime mai mici decât n pentru $n \geq 3$ dat.

Indicație. Se consideră numerele $3, 5, 7, 9, \dots, n$; pentru fiecare astfel de număr se verifică dacă este prim, în caz afirmativ se incrementează cu 1 variabila ce numără numerele naturale prime mai mici decât n .

9. Să se calculeze câte din cele n numere care se citesc în variabila x satisfac condiția

$$1 \leq \text{SQR } x \leq 2.$$



8

Instrucțiunile READ, DATA, RESTORE

Am văzut că introducerea datelor într-un program se face cu instrucțiunea INPUT. O altă modalitate de introducere a datelor este oferită de instrucțiunile **READ** și **DATA**. În programul:

```
P8.1      10 READ a,b
          20 LET s=a+b
          30 PRINT a;" ";b;" ";s
          40 DATA 10,20
          50 STOP
```

instrucțiunea de pe linia 10 atribuie lui a și b valorile 10 și respectiv 20 specificate în instrucțiunea DATA. Cele două instrucțiuni READ și DATA funcționează în mod similar instrucțiunii INPUT cu mențiunea că datele care se introduc sînt specificate în program prin instrucțiunea DATA.

Forma instrucțiunii DATA este următoarea :

```
DATA lista
```

unde lista este o listă de valori despărțite cu virgulă.

Exemplu:

```
DATA 10,10,10,20,30.
```

Forma instrucțiunii READ este următoarea:

```
READ lista
```

unde lista este o listă de variabile (nume de variabile despărțite cu virgulă).

Exemplu:

```
READ a,b,c
```

inițializează variabilele a, b, c cu valori din instrucțiunea DATA.

Observații:

- ◆ Într-un program pot exista mai multe instrucțiuni READ și DATA.
- ◆ Instrucțiunea DATA poate sta pe orice linie în program.

Exemplul 8.1.

```
P8.2      10 FOR n=1 TO 6
           20   READ a
           30   PRINT a;" ";
           40 NEXT n
           50 DATA 1,2,3,4
           60 DATA 5,6
           70 STOP
```

Programul citește în variabila a valorile 1, 2, 3, 4, 5, 6 specificate în cele două instrucțiuni DATA și le afișează pe o linie despărțite de câte un spațiu.

Exemplul 8.2. Se dă o valoare cuprinsă între 1 și 365 și se cere stabilirea lunii și a zilei corespunzătoare acestei valori. De exemplu pentru valoarea 65 găsim luna 3 (martie) și ziua 6.

P8.3

```
10 REM Program pentru calcularea lunii și zilei
20 REM corespunzătoare unei valori între 1 și 365
30 INPUT "Introduceti o valoare: "; val
40 IF val<1 OR val>365 THEN GO TO 30
50 FOR i=1 TO 12
60   READ n: REM n contine nr. de zile din luna
70   IF val<=n THEN PRINT "l: ";i;" z: ";val: STOP
```

```
80 LET val=val-n
90 NEXT i
100 DATA 31,28,31,30,31,30,31,31,30,31,30,31
```

Exemplul 8.3. Să se determine în ce zi a săptămînii cade o anumită dată, cunoscînd ziua săptămînii asociată lui 1 ianuarie. De exemplu dacă 1 ianuarie este în ziua de vineri atunci va trebui să obținem că 7 martie este într-o luni.

Pentru a rezolva problema procedăm astfel:

a) codificăm zilele săptămînii astfel: luni=1, marți=2, miercuri=3, joi=4, vineri=5, sîmbătă=6, duminică=7, iar lunile prin ianuarie=1, februarie=2,...,decembrie=12;

b) dacă 1 ianuarie este vineri atunci vom utiliza o instrucțiune DATA cu următoarele valori:

```
DATA 5,6,7,1,2,3,4
```

care indică următoarea ordine a zilelor săptămînii: vineri, sîmbătă,...,joi;

c) se calculează a cîta zi de la începutul anului reprezintă data aleasă (de exemplu 7 martie este a 66-a zi);

d) se calculează restul împărțirii rezultatului de la punctul c) la 7. Restul obținut indică ziua din săptămînă relativă la datele de la punctul b).Astfel dacă restul este 0 atunci ziua este vineri, iar dacă restul este 4 atunci ziua este marți.

P8.4

```
50 INPUT "Introduceti ziua, luna: "; z;l
60 IF l<1 OR l>12 THEN GO TO 50
70 LET nrz=0 : REM nrz=nr. de zile
80 FOR i=1 TO l
90 READ nl : REM nl=nr. de zile al lunii i
```

```
100 LET nrz=nrz+nl
110 NEXT i
115 REM instructiunile 120-150 "consuma" nr. de zile
116 REM pentru lunile ramase (mai mari decit 1)
120 IF l=12 THEN GO TO 160
130 FOR i=l+1 TO 12
140 READ x
150 NEXT i
160 REM nl contine nr. de zile al lunii l, iar nrz
170 REM nr. de zile pina la sfirsitul lunii l
180 IF z<1 OR z>nl THEN GO TO 50
190 LET nrz=nrz-nl+z
200 REM nrz contine nr. de zile pina la data curenta
210 REM r va fi restul impartirii lui nrz la 7
220 LET r=nrz-7*INT(nrz/7)
230 FOR i=0 TO r
240 READ zs : REM zs indica ziua din saptamina
250 NEXT i
260 PRINT z;" ";l;" este in ziua de ";zs
270 STOP
280 REM lunile au urmatoarele valori
290 DATA 31,28,31,30,31,30,31,31,30,31,30,31
300 REM daca 1 ianuarie este vineri atunci avem
310 DATA 5,6,7,1,2,3,4
```

Vă propunem ca **exercițiu** efectuarea schemei logice pentru acest exemplu.

Dacă dorim ca un set de date să fie reluat atunci se folosește instrucțiunea **RESTORE**. Prima instrucțiune READ ce urmează să se execute după RESTORE va lua lista de valori de la prima valoare a primei instrucțiuni DATA.

Exemplul 8.4. Presupunem că avem două secvențe de valori pe care întâi le adunăm, iar apoi le înmulțim:

P8.5

```
10 DATA 1,2,3,4
20 DATA 5,6,7,8
30 LET s=0
40 FOR i=1 TO 8
50   READ x
60   LET s=s+x
70 NEXT i
80 LET p=1
90 RESTORE
100 REM instructiunea DATA va citi valoarea 1
110 FOR i=1 TO 8
120   READ x
130   LET p=p*x
140 NEXT i
150 PRINT s;" ";p
```

Programul va afișa rezultatele: 36 pentru s și 40320 pentru p, adică $s=1+2+3+4+5+6+7+8$ și $p=1*2*3*\dots*8$. Calcularea valorii lui p (liniile de program 80-140) a fost posibilă ca urmare a utilizării instrucțiunii RESTORE în linia 90, care a condus la reluarea secvenței de valori specificate în instrucțiunile DATA de la prima valoare a primei instrucțiuni DATA.

Exemplul 8.5. Aflarea celui mai mare element dintr-un șir de valori specificate cu instrucțiunea DATA. Se consideră un șir avînd 10 elemente.

P8.6

```
10 REM sirul de 10 elemente este
20 DATA 10,20,-10,-20,0,15,100,3,9,10
30 LET max=-1e+10
40 REM se initializeaza max cu o valoare foarte mica
50 FOR i=1 TO 10
60   READ x
70   IF max<x THEN LET max=x
80 NEXT i
90 PRINT max
```


Rețineți !

■ Instrucțiunea DATA are forma:

$$\text{DATA } v_1, v_2, \dots, v_k$$

unde v_1, v_2, \dots, v_k sînt constante numerice.

■ Instrucțiunea READ are forma:

$$\text{READ } a_1, a_2, \dots, a_n$$

și are ca efect trecerea în variabilele a_1, a_2, \dots, a_n a n valori consecutive specificate în instrucțiuni DATA.

■ Instrucțiunea RESTORE are ca efect reluarea setului de date de la prima valoare a primei instrucțiuni DATA.



Exerciții

1. Să se scrie un program care să calculeze numărul de zile ce au trecut de la începutul anului pînă la o zi dintr-o lună, specificate. De exemplu pentru 15 martie numărul de zile de la începutul anului este 74.
2. Să se scrie programele ce rezolvă exercițiile 1 și 2 de la capitolul 3 utilizînd în locul instrucțiunilor INPUT, instrucțiuni READ și DATA.
3. Să se scrie un program ce calculează suma a 6 valori specificate într-o instrucțiune DATA.

4. Să se scrie un program care să calculeze câte elemente mai mari ca 0 se află într-o succesiune de valori specificate într-o instrucțiune DATA.
5. Să se calculeze suma și produsul a 8 valori numerice specificate într-o instrucțiune DATA, parcurgând o singură dată această secvență.
6. Să se modifice programul din exemplul 8.5 pentru a determina cel mai mic element din șir.
7. Să se scrie un program care să determine cel mai mare și cel mai mic element dintr-un șir de valori specificate într-o instrucțiune DATA, folosind eventual instrucțiunea RESTORE.



Șiruri de caractere. Variabile și expresii de tip șir de caractere

Șirul de caractere a fost prezentat o dată cu instrucțiunile de citire și de scriere. Reamintim că un șir de caractere se specifică printr-o succesiune de caractere cuprinse între " (ghilimele).

Exemplu:

```
"Sirul de caractere"; "123".
```

Un șir de caractere poate fi atribuit unei variabile, numită **variabilă de tip șir de caractere** indicată printr-un nume format dintr-o literă urmată de semnul \$. Următoarele nume:

a\$, b\$, x\$

reprezintă variabile de tip șir de caractere.

Exemplul 9.1.

P9.1

```
10 REM acest program va retine numele în variabila n$
20 INPUT "Cum va numiti?: ";n$
30 PRINT "Bucuros de cunostinta ";n$
40 STOP
```

Cu variabile de tip șir de caractere și cu șiruri de caractere se pot efectua o serie de operații ca:

- **atribuiri**
- **comparații**
- **concatenări** (alipiri de șiruri de caractere).

● **Atribuirea** unui șir de caractere sau a unei variabile de tip șir de caractere la o variabilă de tip șir de caractere este ilustrată în exemplele 9.2 (linia 20), 9.8 (liniile 50, 70, 90, 100 etc.).

● **Comparațiile** între șiruri de caractere (variabile de tip șir de caractere) au rezultatul **adevărat** sau **fals** ca și comparațiile între expresii numerice.

În acest capitol vom considera doar șiruri de caractere care conțin **litere mari sau mici** din alfabetul latin. Dacă șirurile conțin **numai litere mari** sau **numai litere mici**, compararea se poate efectua ținând cont că **două litere pot fi comparate pe baza pozițiilor lor în alfabet**. Astfel, comparațiile "a" < "b", "u" > "d", "M" < "Q", "R" < "S" sînt adevărate deoarece "a" se află în fața lui "b", "u" după "d", "M" înaintea lui "Q" și, respectiv "R" în fața lui "S".

? Ce puteți spune despre comparațiile:

"a" < "b", "a" <> "c", "A" >="C", "A" > "C"?

Trecînd acum la șiruri cu mai multe caractere, avem de exemplu:

"ANA" < "ANICA" - este **adevărată** deoarece prima diferență dintre literele ce compun cele două cuvinte apare pe poziția a 3-a unde primul șir are "A", iar al doilea "I", iar "A" < "I" este adevărată (primele două poziții sînt identice).

"BARBU" > "BARBESCU" - este **adevărată**, deoarece pe poziția a 5-a în primul șir apare "U", iar pe aceeași poziție în al doilea apare "E" (pozițiile de la 1 la 4 coincid în ambele cazuri).

"ANA" > "BARBU" - este **falsă** deoarece poziția întâi din primul șir este "A", iar poziția corespunzătoare din al doilea șir este "B".

"casa" <= "masa" - este **adevărată** deoarece primele poziții din cele două șiruri conțin "c" respectiv "m", iar "c" < "m".

"casa" <> "masa" - este **adevărată** deoarece cele două șiruri nu conțin aceleași componente.

? Ce puteți spune despre următoarele comparații:

"plop" < "casa", "cal" < "canibal", "canicula" >= "cal", "PLOP" >= "CASA", "CAL" >= "CANICULA", "COCOR" < "COCOSTIRC"?

Dacă se compară caractere reprezentînd litere mici și caractere reprezentînd litere mari atunci trebuie să se țină cont că:

! **ORICE LITERĂ MARE SE AFLĂ ÎNAINTEA ORICĂREI LITERE MICI.**

Prin urmare comparațiile: "A" < "c", "M" < "m", "S" < "e", "Z" < "a", sînt toate **adevărate**,

iar comparațiile: "A" >= "c", "X" > "a", "A" > "z", "M" = "m", "M" >= "m", sînt toate **false**.

Literele mari /mici apar în următoarea ordine:

A B C D E F G H I J K L M N O P Q R S T U V X Y Z
a b c d e f g h i j k l m n o p q r s t u v x y z

? Puteți spune care din următoarele condiții sînt adevărate:

"A" > "C", "A" <= "a", "X" <> "x", "U" >= "u", "A" <> "Z", "x" = "y"?

Avînd așezarea în ordine crescătoare a literelor mari/mici ale alfabetului, putem spune că:

"Ana" < "Maria" este adevărată ("A" < "M" este adevărată);

"Anica" < "Anisoara" este adevărată ("c" < "s" este adevărată);

"Ana" <= "ANA" este falsă ("n" > "N" este adevărată);

"Popa" <> "PopA" este adevărată ("a" <> "A" este adevărată);

"Uliul" >= "Corbul" este adevărată ("U" >= "C" este adevărată);

"Uliul" >= "corbul" este falsă ("U" < "c" este adevărată).

Dacă două șiruri sînt astfel încît primul se identifică cu primele litere din al doilea șir, atunci primul este mai mic decît al doilea. De exemplu:

"POP" < "POPA", "POPA" < "POPAZU", "DAN" <= "DANIEL"
sînt adevărate.

? Ce puteți spune despre:

"TATA" <= "mama", "Tata" <> "TaTa", "Elev" > "Copil",
"Cap" = "Capitan"?

Șirurile de caractere pot fi, deci, ordonate într-un mod similar celui folosit pentru a întocmi cataloagele cu numele elevilor. Pentru a întocmi un catalog, se compară numele elevilor și acestea se pun unul înaintea celuilalt dacă la compararea acestora cel care se pune primul are pe prima poziție pe

care numele diferă, o literă ce se află în alfabet înaintea celei corespunzătoare din al doilea nume. De exemplu: Popa și Popescu apar în catalog în această ordine, deoarece prima poziție pe care diferă cele două nume este poziția a treia unde primul nume are litera "a", iar al doilea are litera "e" care se află în alfabet după "a".

Ordonarea șirurilor de caractere ține însă seama și de diferența între litere mari și litere mici așa cum am văzut mai devreme.

● Fiind date două șiruri "ab" și "cd" numim șir de caractere obținut prin concatenarea acestora, șirul "abcd". Operația de concatenare se notează cu simbolul +.

Exemplul 9.2.

P9.2

```
10 REM programul ilustreaza utilizarea sirurilor de
15 REM caractere in instructiunea de atribuire
20 LET a$="abc"
30 PRINT a$
```

Șirul "abc" este atribuit variabilei a\$ (instrucțiunea 20), iar apoi acesta este afișat pe ecran (instrucțiunea 30).

Exemplul 9.3.

P9.3

```
10 REM programul ilustreaza utilizarea sirurilor de
15 REM caractere in comparatii
20 INPUT "Cum va numiti?: ";n$
30 IF n$="BASIC" THEN PRINT "Ati tastat BASIC": STOP
40 PRINT "Nu ati tastat numele limbajului"
50 STOP
```

Programul solicită prin instrucțiunea INPUT un șir de caractere care dacă este "BASIC" atunci este executată instrucțiunea PRINT de pe linia 30, altfel se execută instrucțiunea de pe linia 40.

Exemplul 9.4.**P9.4**

```
10 REM programul ilustreaza diverse posibilitati
15 REM de concatenare de siruri
20 LET a$="ab"+"cd"
30 INPUT "Nume:";n$
40 INPUT "Prenume:";p$
50 PRINT a$
60 LET m$=n$+" "+p$
70 PRINT m$
```

Exemplul 9.5. Programul afișează numele fiecărei luni precum și numărul de zile:

P9.5

```
10 DATA "Ianuarie",31,"Februarie",28;"Martie",31
20 DATA "Aprilie",30,"Mai",31,"Iunie",30
30 DATA "Iulie",31,"August",31,"Septembrie",30
40 DATA "Octombrie",31,"Noiembrie",30,"Decembrie",31
50 FOR i=1 TO 12
60   READ l$,n
70   PRINT "Luna "+l$;" nr.zile ";n
80 NEXT i
90 STOP
```

Exemplul 9.6. Programul modifică exemplul 8.3 astfel încât să se renunțe la codificarea zilelor săptămânii.

P9.6

```
50 INPUT "Introduceti ziua, luna: ";z;l
60 IF l<1 OR l>12 THEN GO TO 50
70 LET nrz=0
80 FOR i=1 TO l
90   READ nl
100  LET nrz=nrz+nl
110 NEXT i
115 REM instructiunile 120-150 consuma numarul de
116 REM zile pentru lunile ramase
120 IF l=12 THEN GO TO 180
```



```

130 FOR i=1+1 TO 12
140 READ x
150 NEXT i
180 IF z<1 OR z>n1 THEN GO TO 50
190 LET nrz=nrz-n1+z
220 LET r=nrz-7*INT(nrz/7)
230 FOR i=0 TO r
240 READ s$:REM s$ indica ziua din saptamina
250 NEXT i
260 PRINT "Ziua din saptamina este "+s$
270 STOP
280 DATA 31,28,31,30,31,30,31,31,30,31,30,31
290 DATA "vineri", "simbata", "duminica", "luni"
300 DATA "marti", "miercuri", "joi"

```

Exemplul 9.7. Dându-se numele unei luni a anului, să se determine a câta lună din an este și câte zile are:

P9.7

```

10 DATA "Ianuarie",31,"Februarie",28,"Martie",31
20 DATA "Aprilie",30,"Mai",31,"Iunie",30
30 DATA "Iulie",31,"August",31,"Septembrie",30
40 DATA "Octombrie",31,"Noiembrie",30,"Decembrie",31
50 INPUT "Introduceti luna: ";l$
60 FOR i=1 TO 12
70 READ n$,nr
80 IF n$=l$ THEN PRINT "luna: ";i;" nr zile: ";nr
: STOP
90 NEXT i
100 STOP

```

Reluăm exemplul 4.4 care rezolva problema mutării a trei copii Ana, Maria și Lucia de pe trei scaune S1, S2 și respectiv S3, pe scaunele S3, S1 și respectiv S2.

Exemplul 9.8.

P9.8

```

10 REM scaunele sint x$, y$, z$ care inlocuiesc pe
15 REM S1, S2, S3, din exemplul 4.4
20 REM Ana se afla pe x$,Maria pe y$ iar Lucia pe z$

```

```

50 LET x$="Ana":LET y$="Maria":LET z$="Lucia"
60 REM Ana trece pe scaunul suplimentar s$
70 LET s$=x$
80 REM Maria trece pe scaunul x$, iar Lucia pe y$
90 LET x$=y$
100 LET y$=z$
110 REM Ana trece pe scaunul z$
120 LET z$=s$
130 PRINT x$;" ";y$;" ";z$

```

Exemplul 9.9. Programul caută șirul de caractere care este cel mai mare într-o mulțime de șiruri de caractere specificate cu DATA.

P9.9

```

10 DATA "ANA", "MARIA", "ANICA", "MONICA", "LUIZA"
20 REM se afiseaza multimea sirurilor de caractere
30 FOR i=1 TO 5
40   READ n$
50   PRINT n$
60 NEXT i
70 RESTORE
80 REM se cauta cel mai mare sir de caractere
90 READ n$
100 LET m$=n$
110 FOR i=2 TO 5
120   READ n$
130   IF m$<n$ THEN m$=n$
140 NEXT i
150 PRINT "Sirul maxim este "+m$

```

Observație: Într-un program pot fi utilizate variabile numerice și variabile de tip șir de caractere cu același nume. Acestea sînt totuși distincte!

Exemplul 9.10.

P9.10

```

10 REM a si a$ desemneaza variabile numerice si
20 REM respectiv sir de caractere
30 LET a=100
40 LET a$="a"
50 PRINT a;" "+a$

```

Programul va afișa în urma execuției:

100 a

ceea ce arată că variabila a conține 100 iar variabila a\$ conține "a".

Rețineți !

■ Un șir de caractere este o succesiune de caractere cuprinse între simbolurile ".

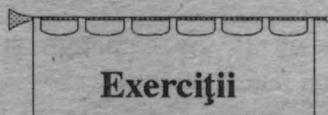
■ Numele unei variabile de tip șir de caractere este formată dintr-o literă și din simbolul \$.

■ Operații permise cu șiruri de caractere:

▶ atribuirea

▶ comparațiile (<, >, <=, >=, =, <>)

▶ concatenarea (+).



1. Să se scrie un program care citește două șiruri de caractere în două variabile și afișează aceste șiruri despărțite de simbolul "+".

2. Să se scrie un program care citește două șiruri de caractere și afișează întâi pe al doilea și apoi pe primul.

3. Să se scrie un program care citește două șiruri de caractere și menționează care din relațiile =, <, >, <=, >=, <> sînt satisfăcute de cele două șiruri.

4. Arătați care din condițiile de mai jos sînt adevărate:

"A" <> "C", "U" < "u", "X" = "x", "U" >= "a", "Ana" <= "ANA", "AN" <= "ANUAL", "AN" <= "ANUL", "UNU" <> "unu", "doi" <= "trei", "doi" <= "TREI".

5. Se dau următoarele instrucțiuni:

```
10 DATA "Ianuarie", "Februarie", "Martie"
20 DATA "Aprilie", "Mai", "Iunie"
30 DATA "Iulie", "August", "Septembrie"
40 DATA "Octombrie", "Noiembrie", "Decembrie"
50 DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
```

care specifică lunile anului și numărul de zile. Folosind aceste instrucțiuni să se conceapă un program care citind o lună (ca un șir de caractere) să afișeze numărul de zile ale lunii.

6. Dîndu-se o mulțime de șiruri de caractere:

"a", "b", "ab", "aab", "bb",

să se conceapă un program care să citească un șir de caractere și să precizeze dacă acesta se află în mulțimea de mai sus.

7. Se dă o mulțime de șiruri de caractere. Să se scrie un program care să determine cel mai mic șir de caractere și cel mai mare șir de caractere.

Indicație. Mulțimea de șiruri de caractere se poate specifica printr-o instrucțiune DATA. Încercați programul pentru următoarele șiruri: "A", "AN", "ANA", "M", "MA", "MAMA", "EU".

Probleme rezolvate în limbajul BASIC

În acest capitol vom aborda cîteva probleme, avînd în general un substrat matematic, pe care le vom transpune în programe scrise în limbajul BASIC.

Vor fi prezentate probleme privind rezolvarea ecuației de gradul I, determinarea unor elemente direct proporționale cu anumite valori date, reprezentarea la scară, determinarea unor mărimi utilizînd exprimări procentuale, calcularea celui mai mare divizor comun.

La rezolvarea problemelor din acest capitol vom utiliza o bună parte din cunoștințele dobîndite pînă acum referitoare la limbajul BASIC.

10.1. Rezolvarea ecuației de gradul I

Ecuația de gradul I are forma:

$$ax+b=0$$

Se știe că aceasta are soluție unică, $x=-b/a$ dacă $a \neq 0$. Pentru $a=0$ sînt două cazuri: dacă $b=0$ atunci ecuația are mai multe soluții, iar dacă $b \neq 0$ atunci ecuația nu are soluție.

P 10.1

```
10 REM se citesc a si b
20 INPUT "Introduceti coeficientii ec.: ";a;b
30 IF a=0 THEN GO TO 60
```

```
40 PRINT "Solutia: x= ";-b/a
50 STOP
60 IF b=0 THEN PRINT " Ec. cu infin. de sol.":STOP
70 PRINT "Ec. nu are solutie"
80 STOP
```

10.2. Mărimi direct proporționale

Să se determine trei numere x , y , z direct proporționale cu a , b , c și a căror sumă este s . Matematic problema se scrie astfel:

$$\begin{cases} \frac{x}{a} = \frac{y}{b} = \frac{z}{c} \\ x+y+z=s \end{cases}$$

Din relațiile de mai sus obținem:

$$y = (b/a) \cdot x \quad (1)$$

$$z = (c/a) \cdot x \quad (2)$$

$$x(1+b/a+c/a) = s \quad (3)$$

Din aceste relații rezultă că numerele x , y , z se pot determina în mod unic, dacă ecuația (3) are soluție unică.

P 10.2

```
10 INPUT "Introduceti factorii de proport.: ";a;b;c
20 INPUT "Introduceti suma: ";s
30 IF 1+b/a+c/a=0 THEN GO TO 110
40 REM se determina x conform cu rel. (3)
50 LET x=s/(1+b/a+c/a)
60 REM se determina y,z conform cu rel. (1) si (2)
70 LET y=(b/a)*x
80 LET z=(c/a)*x
```

```

90 PRINT "x=";x;" y=";y;" z=";z
100 STOP
110 PRINT "x, y, z nu se determină unic"

```

10.3. Reprezentarea la scară

Pentru a desena harta unei țări sau localități este nevoie de o reprezentare a acestui loc la o anumită scară. Reprezentarea la scară, s , înseamnă că unei unități u de pe teren să-i corespundă pe hartă o altă unitate u' ($s=u'/u$). Cele două unități se exprimă de obicei în sisteme de măsură diferite.

De exemplu, presupunem că unitatea u se exprimă în kilometri, u' în centimetri, iar scara este $s=1/500000$. Dacă între două puncte A și B de pe teren este o distanță de 100 de kilometri, atunci pentru a afla distanța dintre punctele corespunzătoare de pe hartă, calculăm:

$$s=1/500000=x/100; \quad x=100 \cdot s. \quad \text{Deci } x=0,0002 \text{ km}=20 \text{ cm.}$$

În general, dacă scara este s , distanța între puncte pe teren este d (în kilometri), atunci distanța d' pe hartă (în centimetri) este:

$$d'=s \cdot d \cdot 100000$$

P 10.3

```

10 INPUT "Introduceti factorul de scara: ";s
20 INPUT "Introduceti distanta ";d
30 REM calculam distanta pe harta. Presupunem ca
40 REM distanta pe teren este in km, iar pe harta
50 REM este in cm.
60 LET dp=s*d*100000
70 PRINT "Distanța pe harta: ";dp;" cm"
80 STOP

```

10.4. Determinarea unor mărimi când se cunoaște procentul

Dându-se o mărime x și un procent $p\%$ din x rezultă că mărimea corespunzătoare este dată de:

$$y = p \cdot x / 100$$

```

P 10.4  10 INPUT "Introduceti marimea: ";x
        20 INPUT "Introduceti procentul: ";p
        30 REM determinarea a p% din x
        40 LET y=(p*x)/100
        50 PRINT "p% din x=";y
        60 STOP

```

10.5. Cel mai mare divizor comun a două numere întregi

În capitolul 7 am discutat problema determinării cîtului și a restului împărțirii a două numere întregi (exemplul 7.11), folosind teorema de împărțire cu rest. Același rezultat se poate folosi și pentru a determina cel mai mare divizor comun astfel:

Dându-se a, b numere întregi putem scrie că

$$a = b \cdot q_1 + r_1 \text{ și } r_1 < b \quad (1)$$

același rezultat se poate apoi aplica pentru b și r_1

$$b = r_1 \cdot q_2 + r_2 \text{ și } r_2 < r_1 \quad (2)$$

și relația poate continua dacă $r_2 > 0$ prin

$$r_1 = r_2 \cdot q_3 + r_3 \text{ și } r_3 < r_2 \quad (3)$$

pînă cînd

$$r_{k-2} = r_{k-1} \cdot q_{k-1} + r_k \text{ și } r_k < r_{k-1} \quad (k)$$

iar $r_k > 0$, deci

$$r_{k-1} = r_k \cdot q_k \quad (k+1)$$

Vă propunem să demonstrați că șirul de relații (1), (2), (3),..., (k) este finit și că r_k este cel mai mare divizor comun al lui a și b.

Trebuie remarcat că r_k poate fi și 1, caz în care a și b sînt prime între ele. Numărul r_k se notează cu **cmmdc(a,b)**.

Această metodă de determinare a celui mai mare divizor comun este cunoscută sub numele de **algoritmul lui Euclid**.

P 10.5

```
10 INPUT "Introduceti a=";a
20 INPUT "Introduceti b=";b
30 LET a=ABS a: LET b=ABS b
40 LET c=INT(b/a): REM c=citul
50 LET r=b-a*c: REM r=restul
60 IF r=0 THEN PRINT "cmmdc(";a;", ";b;")=";a:STOP
70 LET b=a
80 LET a=r
90 GO TO 40
```

10.6 Joc cu întrebări și răspunsuri

Încercați programul următor, remarcînd utilizarea șirurilor de caractere:

P 10.6

```
10 PRINT "Raspundeti cu da sau nu"
20 PRINT "la urmatoarele 5 intrebari"
30 REM rasp=va indica nr.de raspunsuri corecte
```

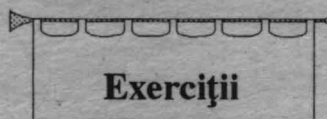
```

40 REM intrebarile si raspunsurile sint in DATA
50 LET rasp=0
60 FOR k=1 TO 5
70   READ i$,r$
80   PRINT i$;
90   INPUT x$:PRINT x$
100  IF x$=r$ THEN LET k=k+1:PRINT "Corect":GO TO 120
110  PRINT "Incorect"
120 NEXT k
130 PRINT "S-a raspuns corect la ";k;" intrebari"
140 DATA "Este adevarata conditia 1+2+3=6?:","da"
150 DATA "Este corecta instructiunea LET a=1?:","da"
160 DATA "IF a=1 THEN LET a=0 este atribuire?:","nu"
170 DATA "REM este instructiune de scriere?:","nu"
180 DATA "PRINT a citeste un numar?:","nu"
190 STOP

```

Programul citește întrebarea în $i\$$ și răspunsul corect în $r\$$; răspunsul dat de la tastatură se află în $x\$$. Dacă răspunsul este corect (adică $x\$=r\$$) atunci se afișează pe ecran "Corect", în caz contrar se afișează "Incorect".

Instrucțiunile DATA de pe liniile 140-180 conțin întrebările și răspunsurile.



1. Arătați care din următoarele secvențe de instrucțiuni sînt corecte:

- a) 10 LET A=10²
- b) 10 IF A=1 THEN A=0
- c) 10 LET a\$="abc"
- d) 10 LET a=1: LET a=2
- e) 10 IF a="abc" THEN a=1
- f) 10 IF a+1=b AND x\$="\$" THEN LET a=1

- g) 10 GO TO 10
 h) 10 IF a=0 THEN GO TO 10

2. Fie programul:

```
10 LET a=1:LET s=0
20 LET s=s+a
30 IF s<=10 THEN LET a=a+1:GO TO 20
40 PRINT s
```

Ce valoare afișează instrucțiunea PRINT s? Explicați rezultatul.

3. Modificați programul prezentat în paragraful 10.6 pentru a răspunde la 10 întrebări pe care să le furnizați.

4. Să se scrie un program care să determine x , y , z care satisfac relațiile:

$$ax=by=cz \text{ și } x+y+z=s \text{ pentru } a, b, c, s \text{ date.}$$

5. Folosind programul prezentat în paragraful 10.5 să se scrie un program pentru a determina cel mai mic multiplu comun al numerelor a și b (notat $\text{cmmmc}(a, b)$) știind că

$$\text{cmmmc}(a, b) = a \cdot b / \text{cmmdc}(a, b)$$

6. Dându-se valorile y și p , să se determine x astfel încât y să reprezinte $p\%$ din x . Să se scrie un program care să calculeze x având ca date de intrare y și p .

7. Dacă prin $\max(a, b)$ înțelegem cel mai mare număr dintre a și b , iar prin $\min(a, b)$ înțelegem cel mai mic număr dintre a și b , să se scrie un program care să calculeze

$$\max(0, a) + \min(0, b)$$

pentru a și b citite la execuție.

8. Dându-se a, b, c trei numere întregi să se scrie un program care să calculeze:

$$\max(|-7|+a-\min(a-b,0), c+|7-b|).$$

9. Să se scrie un program care să calculeze:

$$1:\{1:3+[1:3+1:(1:7+2:14)]\cdot 12:13\}.$$

10. Să se scrie câte un program care să determine:

- toate numerele de forma $\overline{4a3a}$ divizibile cu 2;
- toate numerele de forma $\overline{120a}$ divizibile cu 3;
- toate numerele de forma $\overline{4xy}$ divizibile cu 18;
- toate numerele de forma \overline{xyz} știind că $\overline{xyz}-\overline{zyx}=693$.

Indicație:

- a poate lua valorile 0, 2, 4, 6, 8;
- a se determină din relația $1+2+0+a$ divizibil cu 3;
- x și y satisfac relațiile

$$y=0, 2, 4, 6, 8 \quad \text{și} \quad 4+x+y \text{ se divide cu } 9.$$

d) relația se mai scrie $100 \cdot x + 10 \cdot y + z - 100 \cdot z - 10 \cdot y - x = 693$, adică $99 \cdot (x - z) = 693$ sau $x - z = 6$.

11. Știind că $\text{cmmdc}(a, b, c) = \text{cmmdc}(a, \text{cmmdc}(b, c))$ (adică cel mai mare divizor comun al numerelor a, b, c se determină ca fiind cel mai mare divizor comun al lui a și al celui mai mare divizor comun al lui b și c) să se scrie un program care să determine $\text{cmmdc}(a, b, c)$ pentru a, b, c , trei numere naturale citite la execuție.

12. Să se scrie un program care să determine:

- ce distanță parcurge un vehicul ce merge cu viteza v un timp t ;
- în cât timp parcurge un vehicul distanța d dacă merge cu o viteză v .

Să se verifice programul pentru:

- $v=40$ km/h și $t=1/2$ h;
- $v=40$ km/h și $d=100$ km.

13. Să se facă schema logică pentru rezolvarea problemei din paragraful 10.5.

14. Să se calculeze suma:

$$S = x_6 \cdot 2^6 + x_5 \cdot 2^5 + \dots + x_1 \cdot 2 + x_0$$

știind că numerele x_6, x_5, \dots, x_0 sînt prezente într-o instrucțiune DATA astfel:

$$\text{DATA } x_0, x_1, \dots, x_6.$$

Se va face programul pentru următoarele valori: $x_0 = x_6 = 1$, iar celelalte valori ale lui x_i vor fi egale cu 0.

15. Se dă un șir de 15 numere întregi. Să se calculeze următoarele sume de numere:

- a celor care se află înaintea primului element egal cu 0;
- a celor care se află între două elemente egale cu 0, între care nu se mai află alt element egal cu 0;

c) a celor care urmează după ultimul element egal cu 0.

De exemplu pentru șirul 1 2 -1 0 2 3 0 2 -2 1 0 1 2 1 2, se calculează sumele:

a) $1+2-1$;

b) $2+3$ și $2-2+1$;

c) $1+2+1+2$.

(problema este o generalizare a problemei P46 din G.I.nr.4,1992)

16. Completați enunțul problemei anterioare cu cerința ca sumele care se vor afișa să fie divizibile cu 2.

17. Se dă un șir de 10 numere întregi. Se cere să se calculeze suma valorilor absolute ale celor negative și suma radicalilor de ordin 2 din cele pozitive.

18. Să se determine toate numerele de 3 cifre divizibile cu 3.



Variabile structurate

Pînă acum am folosit variabile ce conțineau valori numerice sau șiruri de caractere. Aceste variabile le numim **simple** deoarece conțin la un moment dat o singură valoare numerică sau un singur șir. Pentru a reține mai multe valori avem nevoie de mai multe variabile.

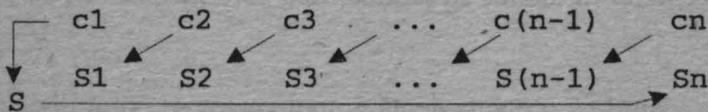
Reamintim exemplul 4.4 care schimba locurile ocupate de Ana, Maria și Lucia:

```

40 LET S1=1: LET S2=2: LET S3=3
60 LET S=S1
80 LET S1=S2
90 LET S2=S3
110 LET S3=S
120 PRINT S1;" ";S2;" ";S3

```

Generalizăm problema de mai sus astfel: se dau $S_1, S_2, S_3, \dots, S_n$ scaune ocupate de $c_1, c_2, c_3, \dots, c_n$ copii și dorim să facem următoarele schimbări c_2 să ocupe scaunul S_1 , c_3 să ocupe scaunul S_2, \dots, c_n să ocupe scaunul $S_{(n-1)}$, iar c_1 să ocupe scaunul S_n . Mutările sînt sugerate de următoarea schemă:



în care mutările s-ar putea face în următoarea ordine:

- c_1 trece pe S
- c_2 trece pe S_1 , c_3 trece pe S_2, \dots, c_n trece pe $S_{(n-1)}$
- c_1 (aflat pe S) trece pe S_n .

O generalizare directă a programului anterior ar însemna practic rescrierea acestuia la fiecare modificare a lui n . Astfel pentru $n=5$ avem nevoie de variabilele S_1, S_2, S_3, S_4, S_5 , iar pentru $n=7$ de $S_1, S_2, S_3, S_4, S_5, S_6, S_7$ etc. Numărul instrucțiunilor de atribuire se modifică și acesta la fiecare schimbare a lui n .

Pentru a putea realiza această generalizare a problemei de mai sus utilizăm o **variabilă structurată**, adică o variabilă avînd **mai multe componente**, fiecare componentă conținînd cîte o valoare.

O variabilă structurată trebuie să fie **declarată**, adică trebuie să apară într-o instrucțiune **DIM**, care are forma:

$$\text{DIM } a(n)$$

unde a este o variabilă structurată cu n componente; n are în mod necesar o valoare în momentul apariției în această instrucțiune.

Observații:

◆ Numele unei variabile structurate trebuie să fie format dintr-o singură literă.

◆ O variabilă structurată ca cea de mai sus va mai fi numită și **vector**.

Componentele variabilei structurate sînt referite prin $a(1), a(2), a(3)$ etc.

Pentru a identifica o componentă a unui vector se pot utiliza și expresii de tipul $a(1+i), a(i+1), a(2*i)$; expresiile ce indică o componentă trebuie să aibă o valoare la momentul utilizării.

Exemplul 11.1. Programul citește 10 valori numerice într-un vector a și calculează suma acestor valori, aflate în componentele vectorului.

P11.1

```

10 DIM a(10): REM se declara a cu 10 componente
20 FOR i=1 TO 10
30   INPUT a(i): REM se citește în componența a(i)
40 NEXT i
50 LET s=0
60 FOR i=1 TO 10
70   LET s=s+a(i)
80 NEXT i
90 PRINT s

```

Instrucțiunea de pe linia 30 citește valori numerice în variabilele $a(1)$, $a(2)$, ..., $a(10)$ (componențele variabilei structurate a).

Instrucțiunea de pe linia 70 calculează suma

$$a(1) + a(2) + \dots + a(10)$$

Acest program poate fi ușor modificat pentru a funcționa pentru vectori având un număr arbitrar, dar precizat de componente. Prezentăm în continuare programul cu aceste modificări:

Exemplul 11.2.**P11.2**

```

10 INPUT "Introduceti numarul componentelor: ";n
20 DIM a(n)
30 FOR i=1 TO n
40   INPUT a(i)
50 NEXT i
60 LET s=0
70 FOR i=1 TO n
80   LET s=s+a(i)
90 NEXT i
100 PRINT s

```

După executarea instrucțiunii din linia 10 se precizează valoarea lui n , deci instrucțiunea din linia 20 se execută pentru o valoare a acestei variabile.

Revenim acum la problema amintită la începutul capitolului, pentru a rezolva cazul în care numărul de scaune și de copii sînt arbitrare. Codificăm copiii c_1, c_2, \dots, c_n cu $1, 2, 3, \dots, n$.

P11.3

```
10 INPUT "Introduceti nr. de scaune:";n
20 DIM S(n)
30 REM initializare: pe Si se aseaza ci, 1<=i<=n
40 FOR i=1 TO n
50   LET S(i)=i
60 NEXT i
70 REM urmeaza schimbarile de locuri
80 LET SP=S(1): REM c1 se aseaza pe SP
90 FOR i=2 TO n
100  LET S(i-1)=S(i): REM ci trece pe S(i-1)
110 NEXT i
120 LET S(n)=SP: REM c1 trece pe Sn
130 FOR i=1 TO n
140  PRINT S(i)
150 NEXT
```

Observați tehnica de declarare a vectorului S.

Pentru a înțelege mecanismul referirii la componentele unei variabile structurate, dezvăluim modul în care se fac atribuirile (linia 100) din instrucțiunea FOR...NEXT (liniile 90-110):

- pentru $i=2$: $S(1)=S(2)$
- pentru $i=3$: $S(2)=S(3)$
- ...
- pentru $i=n$: $S(n-1)=S(n)$

Exemplul 11.3. Introducerea unui element într-un șir. Dindu-se un șir x_1, x_2, \dots, x_n și un element a , se cere să se construiască un nou șir $x_1', x_2', \dots, x_{(n+1)'}$, astfel:

- dacă elementul a nu este egal cu nici unul din elementele șirului dat atunci se pune la sfârșitul acestuia, deci șirul de $n+1$ elemente este următorul:

$$x_1, x_2, \dots, x_n, a$$

- dacă elementul a este egal cu un anumit element x_i din șir, atunci se pune după acesta, deci șirul de $n+1$ elemente arată astfel:

$$x_1, x_2, \dots, x_i, a, x_{(i+1)}, \dots, x_n.$$

P11.4

```

10 INPUT "Introduceti nr. de elemente: ";n
20 DIM x(n+1)
30 REM se citesc elementele sirului
40 PRINT "Se introduc elementele sirului"
50 FOR i=1 TO n
60   INPUT x(i)
70 NEXT i
80 REM se citește elementul care se caută în șir
90 INPUT "Introduceti elementul cautat: ";a
100 FOR i=1 TO n
110  IF x(i)=a THEN GO TO 150
120 NEXT i
130 REM în acest caz a nu se află în șir
140 LET x(n+1)=a: GO TO 220
150 REM în acest caz a se află în șir pe poziția i
160 REM elementele de la i+1 se mută către dreapta
170 IF i=n THEN GO TO 210
180 FOR j=n TO i+1 STEP -1
190  LET x(j+1)=x(j)
200 NEXT j
210 LET x(i+1)=a
220 REM scrie vectorul obținut
230 FOR i=1 TO n+1
240  PRINT x(i)
250 NEXT i
260 STOP

```

Observație: Programul conține o secvență pe care este indicat să o rețineți (liniile 180-200), deoarece execută deplasarea către dreapta cu o poziție a elementelor $x_{(i+1)}, \dots, x_{(n)}$.

Exemplul 11.4. Calcule elementare cu mulțimi (varianta 1).

Vom construi un program care pentru două mulțimi A și B va calcula diferența, reuniunea și intersecția acestor mulțimi notate cu $A-B$, $A+B$, respectiv $A*B$.

Mulțimile vor fi reprezentate ca vectori conținând elementele mulțimilor. De exemplu, dacă $A = \{1, 2, 5, 7\}$, atunci vom reprezenta această mulțime ca un vector A, definit cu $DIM A(4)$, astfel încât $A(1)=1$, $A(2)=2$, $A(3)=5$, $A(4)=7$.

Înainte de a trece la prezentarea programului reamintim definițiile matematice ale operațiilor mai sus menționate.

$$\begin{aligned} A - B &= \{x \mid x \in A \text{ și } x \notin B\} \\ A + B &= \{x \mid x \in A \text{ sau } x \in B\} \\ A * B &= \{x \mid x \in A \text{ și } x \in B\} \end{aligned}$$

Programul care urmează va citi întâi mulțimile A și B, apoi le va afișa și în final va calcula $A-B$, $A+B$, $A*B$.

P11.5

```

10 INPUT n,m:REM nr.de elemente pentru A respectiv B
20 DIM A(n): DIM B(m)
25 REM urmeaza citirea multimii A
30 FOR i=1 TO n
40   INPUT A(i)
50 NEXT i
55 REM urmeaza citirea multimii B
60 FOR i=1 TO m
70   INPUT B(i)
80 NEXT i
85 REM este afisata multimea A
90 PRINT "A={" ; A(1) ;
100 FOR i=2 TO n
110   PRINT " ," ; A(i) ;
120 NEXT i

```



```
130 PRINT "}"
135 REM este afisata multimea B
140 PRINT "B={" ; B(1) ;
150 FOR i=2 TO m
160   PRINT " , " ; B(i) ;
170 NEXT i
180 PRINT "}"
185 REM se calculeaza A-B
190 PRINT "A-B={" ;
200 FOR i=1 TO n
210   FOR j=1 TO m
220     IF A(i)=B(j) THEN GO TO 250
230   NEXT j
240   PRINT A(i) ; " , " ;
250 NEXT i
260 PRINT "}"
265 REM se calculeaza A+B
270 PRINT "A+B={" ;
280 FOR i=1 TO n
290   PRINT A(i) ; " , " ;
300 NEXT i
310 FOR j=1 TO m
320   FOR i=1 TO n
330     IF B(j)=A(i) THEN GO TO 360
340   NEXT i
350   PRINT B(j) ; " , " ;
360 NEXT j
370 PRINT "}"
375 REM se calculeaza A*B
377 PRINT "A*B={" ;
380 FOR i=1 TO n
390   FOR j=1 TO m
400     IF A(i)=B(j) THEN PRINT A(i) ; " , " ; : GO TO 420
410   NEXT j
420 NEXT i
430 PRINT "}"
440 STOP
```

Programul de mai sus a fost realizat de Ion Diamandi.

Exemplul 11.5. Calcule elementare cu mulțimi (varianta 2).

Vom relua problema din exemplul anterior, dar de această dată vom alege un alt mod de a reprezenta mulțimile. Vom presupune că lucrăm cu mulțimi avînd elemente cuprinse între 1 și n . În acest caz mulțimea A va fi reprezentată de vectorul A definit cu $\text{DIM } A(n)$, în care $A(i) = 1$, dacă $i \in A$ și $A(i) = 0$, în caz contrar.

De exemplu, dacă $n=10$ și $A = \{1, 3, 7, 8\}$ atunci vectorul A va avea următoarele componente:

$$A(1) = A(3) = A(7) = A(8) = 1$$

și

$$A(2) = A(4) = A(5) = A(6) = A(9) = A(10) = 0$$

P11.6

```

10 INPUT n: REM limita domeniului multimilor A,B
20 DIM A(n): DIM B(n)
30 REM Aceasta varianta de BASIC prevede ca
35 REM toate componentele lui A si B sa fie 0.
40 REM In lipsa acestei facilitati componentele
50 REM lui A si B trebuie facute 0
60 REM Se citeste A, introducind pe rind elementele
70 REM sale; dupa ultimul element se introduce 0.
80 INPUT "Elementul lui A=";x
90 IF x=0 THEN GO TO 120
100 LET A(x)=1
110 GO TO 80
120 REM Se citeste B
130 INPUT "Elementul lui B=";x
140 IF x=0 THEN GO TO 170
150 LET B(x)=1
160 GO TO 130
170 REM A-B
175 PRINT "A-B={";
```

```

180 FOR i=1 TO n
190   IF A(i)=1 AND B(i)=0 THEN PRINT i;",";
200 NEXT i
210 PRINT "}"
220 REM A+B
225 PRINT "A+B={"
230 FOR i=1 TO n
240   IF A(i)=1 OR B(i)=1 THEN PRINT i;",";
250 NEXT i
260 PRINT "}"
270 REM A*B
275 PRINT "A*B={"
280 FOR i=1 TO n
290   IF A(i)=1 AND B(i)=1 THEN PRINT i ;",";
300 NEXT i
310 PRINT "}"
320 STOP

```

Exemplul 11.6. Generarea submulțimilor unei mulțimi.

Fiind dată o mulțime A, de exemplu $A = \{1, 2, 4\}$, submulțimile acesteia sînt:

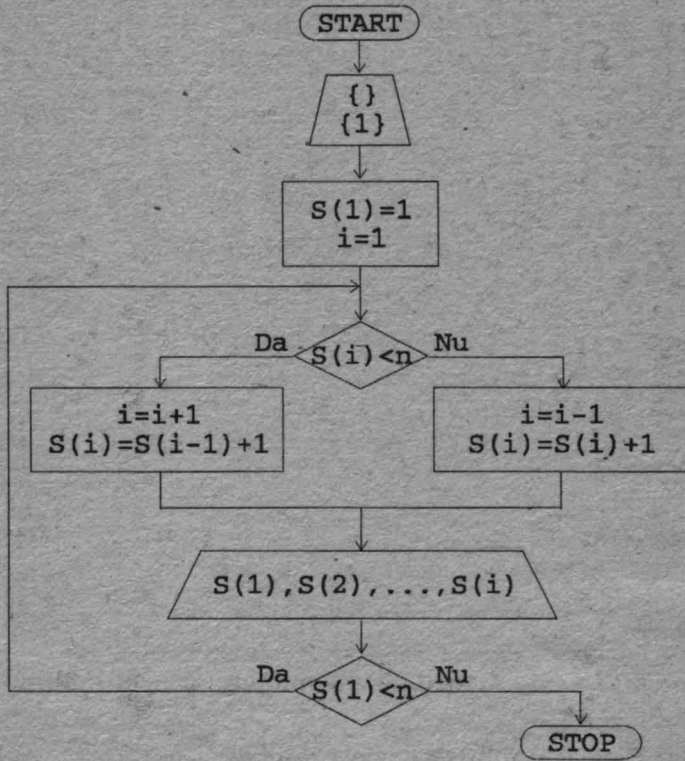
```

{} - mulțimea vidă
{1}, {2}, {3}, {4}
{1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}
{1,2,3}, {1,2,4}, {1,3,4}, {2,3,4}
{1,2,3,4} = A.

```

Mulțimile vor fi reprezentate ca în varianta 1 a operațiilor cu mulțimi (exemplul 11.4).

Pentru a genera submulțimile unei mulțimi A care conține elementele 1, 2, 3, ..., n, considerăm următoarea schemă logică:



Vă propunem să verificați că algoritmul prezentat de schema logică de mai sus calculează pentru $A = \{1, 2, 3, 4\}$, următoarea succesiune de submulțimi:

$\{\}, \{1\}, \{1,2\}, \{1,2,3\}, \{1,2,3,4\}, \{1,2,4\}, \{1,3\}, \{1,3,4\}, \{1,4\}, \{2\}, \{2,3\}, \{2,3,4\}, \{2,4\}, \{3\}, \{3,4\}, \{4\}$.

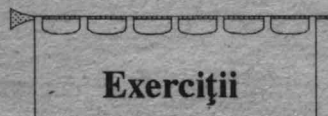
Rețineți !

■ Declarația unei variabile structurate, a, se face prin:

DIM a(n)

■ Numele unei variabile structurate este compus dintr-o singură literă

■ Referirea componentelor unei variabile structurate, a , se face prin $a(1)$, $a(2)$, $a(3)$, etc., sau prin $a(i)$, $a(i+1)$, $a(i+j)$, etc.



1. Să se modifice programul din exemplul 11.1 pentru a calcula suma componentelor vectorului A odată cu citirea acestora.

2. Se dau n scaune S_1, S_2, \dots, S_n ocupate de n copii c_1, c_2, \dots, c_n . Să se scrie un program care să realizeze următoarele mutări:

- c_1 trece pe scaunul S_2
- c_2 trece pe scaunul S_3
- ...
- $c_{(n-1)}$ trece pe scaunul S_n
- c_n trece pe scaunul S_1 .

3. Se citește un șir de n numere într-un vector. Să se determine cel mai mare și cel mai mic element din șirul dat.

4. Se citește un șir de n numere într-un vector. Să se determine cel mai mare element și cel mai mic element din șirul dat; să se indice în plus pozițiile din șir în care apar aceste elemente. Să se execute programul pentru $n=5$ și pentru șirul $1, 2, 1, 2, 1$. Trebuie să se obțină maximul egal cu 2 și poziția a 2-a, iar minimul egal cu 1 și poziția întâi.

5. Se citește un șir de n numere într-un vector. Să se determine cel mai mare și cel mai mic element din șirul dat. Să se schimbe locurile acestor două elemente între ele. Să se afișeze șirul după modificările efectuate.

6. Se citește un șir de n numere într-un vector și un element a . Dacă a este egal cu un element din șir atunci acest element se elimină din șir, altfel șirul rămîne nemodificat.

7. Să se scrie un program care să determine cîte elemente sînt mai mici ca 0 într-un șir care se află memorat într-un vector.

8. Să se scrie un program care pentru două mulțimi A și B să determine dacă A este o submulțime a lui B .

9. Se dă un șir x_1, x_2, \dots, x_n memorat într-un vector X , cu n număr par. Să se scrie un program care să calculeze într-un vector Y următorul șir:

$$Y(1) = X(1) + X(n), \quad Y(2) = X(2) + X(n-1),$$

$$Y(3) = X(3) + X(n-3), \dots$$

Să se precizeze cîte elemente are Y și care este formula de calcul pentru ultimul termen al lui Y ?

10. Să se scrie un program după schema logică prezentată în exemplul 11.6.

11. Să se scrie un program pentru a genera submulțimile de cîte m elemente ale unei mulțimi de n elemente. Să se verifice programul pentru $m=2, n=4$ (soluția: $\{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}$).

12. Să se scrie un program care să genereze submulțimile unei mulțimi X care conțin o mulțime Y . Se va verifica programul pentru $X=\{1,2,3,4,5\}, Y=\{1,3\}$ (soluția: $\{1,3\}, \{1,3,2\}, \{1,3,4\}, \{1,3,5\}, \{1,3,2,4\}, \{1,3,2,5\}, \{1,3,4,5\}, \{1,3,2,4,5\}$).

13. Se dă un vector cu n componente ce conține $n-1$ elemente și un loc liber pe poziția i .

a) Să se transforme vectorul astfel încât poziția liberă să fie j .

b) Să se transforme vectorul astfel încât poziția liberă să fie prima poziție (ultima poziție).

Exemplu: Pentru $i=3$, $j=6$, vectorul:

$$1, 2, \quad, -1, 2, 4, 3, 4$$

se transformă în

a) $1, 2, -1, 2, 4, \quad, 3, 4$

b) $\quad, 1, 2, -1, 2, 4, 3, 4$

$$1, 2, -1, 2, 4, 3, 4, \quad.$$

14. Se dă un vector cu n componente având ocupate primele r componente și ultimele $n-r-1$ componente, componenta $r+1$ fiind liberă. Se cere să se obțină următoarea aranjare a elementelor în vector: ultimele $n-r-1$ componente inițiale se pun pe primele $n-r-1$ poziții, iar primele r componente inițiale se pun pe ultimele r poziții, iar poziția liberă va deveni $n-r$. Mutările de componente se fac fără a utiliza variabile în care să se memoreze temporar componente ale vectorului. De exemplu pentru $n=8$, $r=3$ și vectorul

$$-1, -1, -2, \quad, 1, 2, 0, 1$$

se ajunge la vectorul:

$$1, 2, 0, 1, \quad, -1, -1, -2.$$

Subrutine

Există situații în care anumite părți de program se repetă de două sau mai multe ori. De exemplu dorim să scriem un program care citește două numere, verifică dacă sînt pozitive și le adună, apoi citește alte două numere verifică dacă sînt pozitive și le înmulțește. O primă transcriere în BASIC a algoritmului anterior ar fi următoarea:

```
P12.1      10 INPUT a;b
           20 IF a<=0 OR b<=0 THEN GO TO 10
           30 LET s=a+b: PRINT s
           40 INPUT a;b
           50 IF a<=0 OR b<=0 THEN GO TO 40
           60 LET p=a*b: PRINT p
           70 STOP
```

Este evident că secvențele de program 10-20 și 40-50 sînt identice.

Limbajul BASIC oferă posibilitatea de a concepe separat secvențele de program care se repetă, sub forma **subrutinelor**. O **subrutină** este o secvență de instrucțiuni care se termină cu:

RETURN

Apelarea unei subrutine într-un program se face cu instrucțiunea:

GO SUB n

unde n este numărul de linie al primei instrucțiuni din subrutină.

Exemplul 12.1. Problema prezentată mai sus poate fi rescrisă astfel:

P12.2

```

10 REM apelam subrutina pentru a citi a si b si
20 REM pentru a verifica daca sint pozitive
30 GO SUB 100
40 LET s=a+b: PRINT s
50 REM apelam din nou subrutina
60 GO SUB 100
70 LET p=a*b: PRINT p
80 STOP
100 REM subrutina
110 INPUT a;b
120 IF a<=0 OR b<=0 THEN GO TO 110
130 RETURN

```

Subrutina (liniile 100-130) citește a și b și verifică dacă acestea sînt pozitive. Subrutina este apelată în liniile 30 și 60. Programul se execută astfel:

- se apelează subrutina (linia 30) care citește a, b și verifică dacă acestea sînt pozitive;
- se calculează suma s (linia 40);
- se apelează din nou subrutina (linia 60);
- se calculează produsul p (linia 70);
- programul conține STOP (linia 80) care este necesar în acest caz deoarece în lipsa acestuia ar fi fost executate, după calcularea produsului, instrucțiunile subrutinei.

Exemplul 12.2. Dîndu-se un șir de numere x_1, x_2, \dots, x_n , să se determine pozițiile pe care se află primele două elemente în ordine descrescătoare din șir. De exemplu pentru șirul:

1, -1, 3, 5, 7, -2, 10, 3

răspunsul este 7, 5 (poziția 7 îl indică pe 10, iar poziția 5 îl indică pe 7). Pentru a rezolva această problemă vom concepe o subrutină care calculează poziția celui mai mare element într-un șir. Vom apela această

subrutină, vom pune apoi pe poziția celui mai mare element o valoare foarte mică și vom apela din nou subrutina, obținând astfel poziția celui mai mare element dintre cele ale noului șir.

P12.3

```

10 REM se citește șirul
20 INPUT "Cite elemente are șirul?:";n
30 DIM X(n)
40 FOR i=1 TO n
50   INPUT X(i)
60 NEXT i
70 GO SUB 1000
80 LET MAX1=k: LET X(k)=-1.0e7
90 REM în poziția maximului, k, s-a pus -10000000
100 GO SUB 1000
110 LET MAX2=k
120 PRINT MAX1;" ";MAX2
130 STOP
1000 REM subrutina care determină poziția
1010 REM celui mai mare element
1020 LET MAX=X(1): LET k=1
1030 FOR i=2 TO n
1040   IF MAX<X(i) THEN LET MAX=X(i): LET k=i
1050 NEXT i
1060 REM poziția maximului este k
1070 RETURN

```

Exemplul 12.3. Căutarea unui element într-un șir.

Dându-se un șir x_1, x_2, \dots, x_n să se determine dacă un element a se află în acest șir. Programul pentru rezolvarea acestei probleme îl vom concepe astfel:

- se citește șirul x_1, x_2, \dots, x_n într-un vector x
- se citește elementul a
- se caută elementul a în șir, indicând și poziția (pentru aceasta vom construi o subrutină).

Tehnica după care se face căutarea elementului a în șir va fi următoarea:

- elementul se pune pe poziția $n+1$ în șir
- se compară apoi elementele șirului cu a .

În urma acestei căutări a va trebui să se afle printre cele $n+1$ elemente, dar el se află în șirul inițial dacă a a fost egal cu unul din primele n elemente.

P12.4

```

10 INPUT "Introduceti nr. de elemente ale sirului:";n
20 DIM X(n+1)
30 FOR i=1 TO n
40   INPUT X(i)
50 NEXT i
60 INPUT "Introduceti elementul cautat:";a
70 GO SUB 1000
80 IF k=n+1 THEN PRINT a;" nu este in sir": STOP
90 PRINT a;" este in sir": STOP
1000 REM subrutina de cautare
1010 LET X(n+1)=a
1020 FOR i=1 TO n+1
1030   IF X(i)=a THEN LET k=i: RETURN
1040 NEXT i
1050 RETURN

```

Observație: Instrucțiunea RETURN este ultima instrucțiune care se execută într-o subrutină, dar nu trebuie să fie totdeauna pe ultima poziție în subrutină. A se vedea pentru exemplificare instrucțiunea RETURN de pe linia 1030 din exemplul anterior.

Exemplul 12.4. Ordonarea crescătoare a elementelor unui șir.

Programul care rezolvă această problemă are următoarele etape:

- se citesc elementele șirului în vectorul $X(n)$
- se consideră apoi toate valorile lui i de la 1 la $n-1$ și se determină

cel mai mare element dintre $X(i+1)$, $X(i+2)$, ..., $X(n)$, care se pune pe poziția i , iar elementul de pe poziția i trece pe poziția maximului.

Remarcăm faptul că determinarea maximului și a poziției sale se fac printr-o subrutină.

P12.5

```

10 INPUT "Introduceti nr. de elemente ale sirului:";n
20 DIM X(n)
30 FOR i=1 TO n
40   INPUT X(i)
50 NEXT i
60 FOR i=1 TO n-1
70   GO SUB 1000
80 NEXT i
90 FOR i=1 TO n
100  PRINT X(i)
110 NEXT i
120 STOP
1000 REM subrutina pentru determinarea maximului
1001 REM si punerea elementului maxim pe pozitia i
1010 LET a=X(i): LET k=i: REM k=poz.MAX
1020 FOR j=i+1 TO n
1030   IF X(j)>a THEN LET k=j: LET a=X(j)
1040 NEXT j
1050 LET X(k)=X(i)
1060 LET X(i)=a
1070 RETURN

```

Exemplul 12.5. Program cu mai multe subrutine.

Următorul program calculează:

$$E = a + b + a * b + a / b + a - b$$

utilizând subrutine pentru a calcula suma, diferența, cîmul și produsul elementelor a și b (se presupune $b \neq 0$).

```

P12.6          10 INPUT a,b
                  20 GO SUB 100
                  30 GO SUB 200

```



```

40 GO SUB 300
50 GO SUB 400
60 LET E=S+P+C+D: PRINT E
70 STOP
100 LET S=a+b
110 RETURN
200 LET P=a*b
210 RETURN
300 LET C=a/b
310 RETURN
400 LET D=a-b
410 RETURN

```

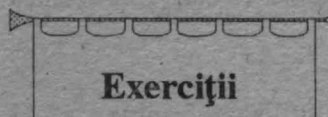
Rețineți !

- Apelarea unei subrutine se face cu

GO SUB n

- Ultima instrucțiune care se execută într-o subrutină trebuie să fie RETURN.

- Dacă o subrutină este scrisă după program atunci acesta trebuie să se termine cu STOP.



1. Dându-se două șiruri a_1, a_2, \dots, a_n și b_1, b_2, \dots, b_n , să se conceapă un program care să calculeze un șir c_1, c_2, \dots, c_n , astfel încât $c_1 = a_1 + b_1, c_2 = a_2 + b_2, \dots, c_n = a_n + b_n$.

Programul va utiliza o subrutină pentru a calcula șirul c_1, c_2, \dots, c_n .

2. Aceeași problemă de la 1 cu mențiunea că șirul c_1, c_2, \dots, c_n este dat de $c_1 = a_1 + b_n, c_2 = a_2 - b_{n-1}, c_3 = a_3 + b_{n-2}, c_4 = a_4 - b_{n-3}, \dots$

3. Fie programul:

```
10 LET A=1
20 GO SUB 40
30 PRINT A
40 LET A=A+1
50 RETURN
```

Este corect acest program?

4. Fie programul:

```
10 INPUT A
20 GO SUB 40
30 PRINT A
35 STOP
40 LET A=A+1
50 RETURN
```

În ce caz va afișa acest program valoarea 2?

5. Scrieți un program pentru realizarea operațiilor elementare cu mulțimi (reuniunea, intersecția, diferența) concepînd pentru fiecare din acestea câte o subrutină.

6. Rescrieți exemplele 12.3 și 12.4 fără a mai utiliza subrutine.

7. Se dă un vector de elemente numerice. Să se scrie un program care să elimine toate elementele egale cu zero ale vectorului. Se va scrie o subrutină care să elimine, în momentul în care a fost găsit, un zero. De exemplu vectorul X conține 1, 2, 0, 3, 0, 1 și după prelucrare devine 1, 2, 3, 1.

8. Se dă un șir de elemente x_1, \dots, x_n și o valoare a . Să se scrie un program care să transforme acest șir astfel: toate elementele mai mici decât a rămân neschimbate, iar cele mai mari decât a se schimbă cu a . De exemplu pentru șirul $1, 2, -1, 10, 20, 1$ și $a=5$ se obține $1, 2, -1, 5, 5, 1$.

9. Să se scrie un program care să completeze un vector x astfel:

- componentele de indice par cu 0 ($x(2) = x(4) = \dots = 0$)

- componentele de indice impar cu suma indicilor pînă la cel curent ($x(1) = 1, x(3) = 6, x(5) = 14$ etc).

Se va concepe o subrutină pentru a calcula suma indicilor.

10. Să se scrie un program pentru a calcula:

$$(1+2+\dots+8) / (1 \cdot 2 \cdot \dots \cdot 8)$$

folosind cîte o subrutină pentru a calcula suma și produsul.



13

Instrucțiunea BEEP

Calculatorul efectuează așa cum am văzut pînă acum o serie de operații (adunări, scăderi, înmulțiri, comparații etc.) cu ajutorul cărora am construit programe pentru rezolvarea anumitor probleme. Calculatorul însă poate să emită și...sunete!. În continuare vom vedea cum putem folosi (programa) această facilitare.

Dacă nu ați încercat pînă acum, vă propunem cîteva note muzicale!

```
10 BEEP 1,0  
20 BEEP 1,2  
30 BEEP 1,3  
40 BEEP 1,5
```

Dacă aveți puțină ... ureche muzicală și dacă difuzorul calculatorului nu este dereglat, ați recunoscut primele note ale gamei do major (do, re, mi, fa) și faptul că toate au aceeași durată.

Din exemplul discutat deducem că forma generală a acestei instrucțiuni (este totuși o instrucțiune!) este:

```
BEEP d,i
```

unde d desemnează durata, iar i înălțimea (cea care corespunde unei anumite note).

Pentru a putea transcrie o melodie (de pe un portativ) într-o secvență de instrucțiuni **BEEP** este necesar să cunoaștem ceva mai mult despre legătura dintre notele muzicale și valorile înălțimii (parametrul al doilea din BEEP).

Iată câteva valori ale înălțimii:

note	do	re	mi	fa	sol	la	si	do
valori	0	2	3	5	7	8	10	
valori	12	14	15	17	19	20	22	*24

Dacă dorim note muzicale sub nota do din gama cu același nume, atunci luăm valori negative:

note	do	re	mi	fa	sol	la	si	do
valori	-12	-10	-9	-7	-5	-4	-2	0

Ați observat că diferența a două înălțimi consecutive este de 1 sau 2 după cum notele corespunzătoare determină un semiton, respectiv un ton.

Celălalt parametru al instrucțiunii BEEP, durata, se exprimă în secunde (1 - indică o secundă). Asocierea unei unități la un tip de notă (notă întreagă, doime, pătrime etc.) este la alegerea noastră. Putem de exemplu, pentru pătrime să luăm o secundă; rezultă că pentru doime luăm două secunde, iar pentru optime o jumătate de secundă și pentru șaisprezecime un sfert de secundă.

Fie următorul portativ:

do re mi re do do re sol

Programul care-l traduce în instrucțiuni BEEP este următorul:

```
P13.1      10 BEEP 1,0: BEEP 1,2: BEEP 0.5,3
           20 BEEP 0.5,2: BEEP 1,0
           30 BEEP 1,0: BEEP 1,2: BEEP 2,7
```

O tehnică mai elegantă și mai generală de a transpune un portativ într-un program este aceea dată de utilizarea instrucțiunilor READ, DATA, eventual RESTORE.

Transcriem cu această tehnică portativul anterior și în același timp repetăm primele cinci note.

Exemplul 13.1.

```
P13.2
10 DATA 1,0,1,2,0.5,3,0.5,2,1,0,1,0,1,2,2,7
20 FOR i=1 TO 5: REM se "cînta" primele 5 note
30   READ d,in
40   BEEP d,in
50 NEXT i
60 RESTORE: REM reluam primele 5 note
70 FOR i=1 TO 8: REM se "cînta" toate notele
80   READ d,in
90   BEEP d,in
100 NEXT i
110 STOP
```

Știm de la muzică faptul că o melodie se poate cînta într-un ritm mai rapid sau mai lent. Acest lucru este posibil să-l cerem și calculatorului nostru! Pentru aceasta înlocuim liniile 40 și 90 cu

BEEP d*p,in.

Valoarea lui p se stabilește la începutul programului (de exemplu printr-un INPUT). Dacă $p < 1$ atunci melodia va avea un ritm mai vioi (încercați pentru $p=0.5$, $p=0.2$), iar dacă $p > 1$, atunci melodia va fi mai lentă (încercați pentru $p=2$, $p=5$). Evident, $p=1$ corespunde cazului inițial.

Instrucțiunea BEEP poate fi folosită în programe și pentru a semnala sonor faptul că se execută sau nu anumite secvențe de instrucțiuni. De exemplu, dorim să introducem două numere pozitive pe care să le comparăm pentru a-l afișa pe cel mai mare; dacă la citire numerele nu satisfac condiția de a fi pozitive, atunci se emite un semnal sonor și se reia citirea:

Exemplul 13.2.

P13.3

```

10 GO SUB 100
15 REM se citeste in a primul numar pozitiv
20 LET x=a
30 GO SUB 100
35 REM se citeste in a al 2-lea nr. pozitiv
40 LET y=a
50 IF x<y THEN LET x=y
60 PRINT "Cel mai mare este: ";y
70 STOP
100 REM subrutina de citire a unui numar pozitiv
110 INPUT "a=";a
120 IF a<0 THEN BEEP.0.5,10: GO TO 110
130 RETURN

```

Revenind la tehnicile de a transpune un portativ în instrucțiuni, remarcăm necesitatea existenței unei instrucțiuni care să modeleze pauzele ce apar între anumite note muzicale. Această instrucțiune este

PAUSE n

unde n indică numărul de unități. Reținem că o secundă are circa 50 de unități. Deci dacă pătrimea o asimilăm cu o secundă, atunci o pauză de o pătrime se va genera cu

PAUSE 50 .

Exemplul 13.3. Programul următor "interpretează" o melodie numită "Banul Mărăciine" și utilizează un parametru *p* care, dacă este modificat la diversele rulări schimbă ritmul melodiei (încercați pentru valorile 1, 0.5, 2). Parametrul *p* este citit în linia 20.

P13.4

```

10 PRINT "      **** Banul Maracine ****"
20 INPUT p
30 FOR i=1 TO 22
40   READ d,in
50   BEEP d*p,in
60 NEXT i
70 PAUSE 50*p: REM pauza de 1 sec.=1 patrime
80 FOR i=1 TO 22
90   READ d,in
100  BEEP d*p,in
110 NEXT i
120 DATA 0.5,7,0.5,8,1,7,0.5,5,0.5,7,1,5,0.5,3,0.5,2
130 DATA 0.5,3,0.5,5,2,7,0.5,7,0.5,8,1,7,0.5,5,0.5,7
140 DATA 1,5,0.5,3,0.5,0,0.5,2,0.5,-1,1,0
150 REM intre aceste note apare pauza de o patrime
160 DATA 0.5,-1,0.5,0,1,2,0.5,2,0.5,3,1,5,0.5,3
170 DATA 0.5,2,0.5,3,0.5,5,2,7,0.5,-1,0.5,0,1,2
180 DATA 0.5,2,0.5,3,1,5,0.5,3,0.5,0,0.5,2,0.5,-1,1,0

```

 **Rețineți !**

- Instrucțiunea BEEP are forma

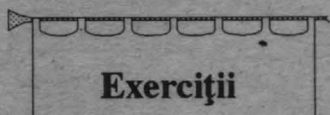
BEEP d,in

unde *d* este durata, iar *in* este înălțimea.

- Pauza între note se realizează cu

PAUSE n

unde *n* indică numărul de unități (1 sec.=50 unități).



Exerciții

1. Scrieți gama do major (do, re, ..., do) luând ca durată unitatea (BEEP 1, n).
2. Scrieți gama utilizând un parametru pentru durată, pe care să-l introduceți cu INPUT.
3. Rescrieți exemplul 13.1 parametrizând durata.
4. Scrieți un program în care înainte de fiecare afișare pe ecran să emiteți câte un semnal sonor.
5. Alegeți o melodie și transcrieți-o cu ajutorul instrucțiunilor BEEP, READ, DATA, eventual PAUSE și RESTORE.



Punctul de pe ecran în care ne aflăm la un moment dat se numește **poziție curentă**.

Pregătirea ecranului pentru a desena pe acesta, ștergînd toate informațiile anterior scrise sau desenate, se face cu instrucțiunea CLS. După executarea acestei instrucțiuni, ecranul este șters, iar poziția curentă este în punctul de coordonate $(0, 0)$, adică în stînga jos.

Instrucțiunile de grafică ale limbajului BASIC ne permit să desenăm **puncte, linii și cercuri**.

Desenarea punctelor se face cu instrucțiunea:

PLOT x, y

care pune un punct pe ecran avînd coordonatele (x, y) .

Exemplul 14.1. Următorul program trasează o linie punctată orizontală între punctele avînd coordonatele $(0, 50)$ și $(150, 50)$.

```
P14.1      10 CLS: REM sterge ecranul
           20 FOR i=0 TO 50
           30   PLOT 3*i,50
           40 NEXT i
```

Se observă că acest program "pune" puncte pe ecran avînd coordonatele $(0, 50)$, $(3, 50)$, $(6, 50)$, ..., $(150, 50)$.

Trasarea unei linii se face cu instrucțiunea

DRAW x, y

care desenează o linie între poziția curentă și punctul ale cărui coordonate se obțin adunînd la cele ale poziției curente x , respectiv y .

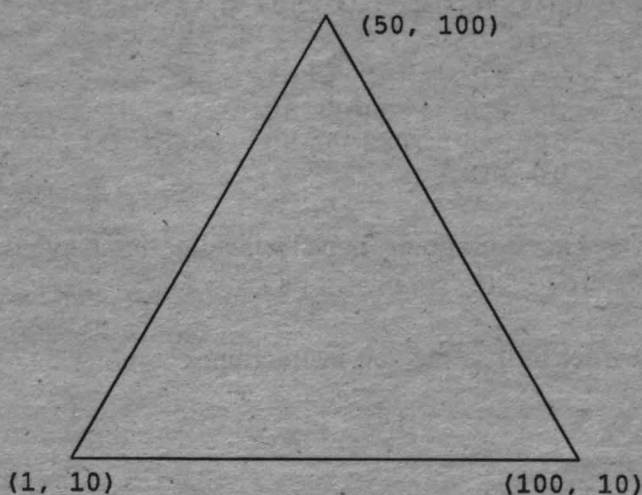
Exemplul 14.2. Programul trasează linii între punctele de coordonate (10,10) - (50,10) (linie orizontală), (100,100) - (100,50) (linie verticală) și (20,20) - (50,50) (linie oblică). Pentru a trasa aceste linii trebuie ca înainte de utilizarea instrucțiunilor DRAW să folosim instrucțiunea PLOT pentru a obține poziția curentă convenabilă.

P14.2

```
10 CLS
20 PLOT 10,10: REM pozitia curenta (10,10)
30 DRAW 40,0: REM linie:(10,10) -> (50,10)
40 PLOT 100,100: REM pozitia curenta (100,100)
50 DRAW 0,-50: REM linie:(100,100) -> (100,50)
60 PLOT 20,20: REM pozitia curenta (20,20)
70 DRAW 30,30: REM linie:(20,20) -> (50,50)
```

Cu ajutorul instrucțiunii DRAW se pot trasa diferite figuri geometrice ca: triunghiuri, dreptunghiuri, pătrate etc.

Exemplul 14.3. Pentru a desena triunghiul de mai jos

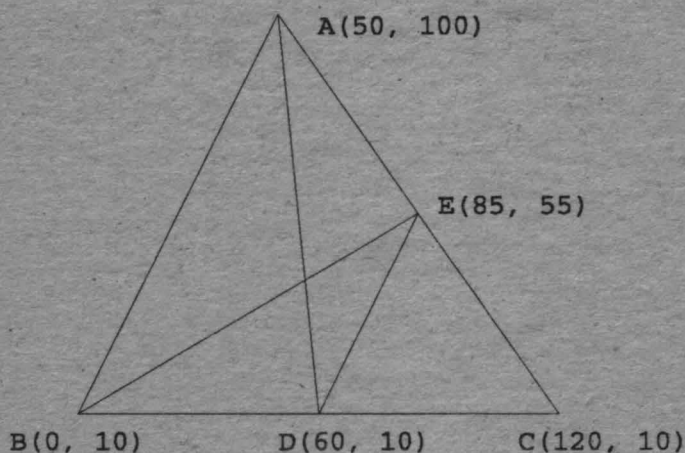


utilizăm următorul program:

P14.3

```
10 CLS
20 PLOT 50,100
30 DRAW -49,-90: REM linie:(50,100) -> (1,10)
40 DRAW 99,0: REM linie:(1,10) -> (100,10)
50 DRAW -50,90: REM linie:(100,10) -> (50,100)
```

Exemplul 14.4. În continuare ne propunem să realizăm următorul desen:



Pentru aceasta vom trasa următoarea succesiune de segmente: AB, BC, CA, AD, DE, EB, cu ajutorul instrucțiunii DRAW. Parametri necesari trasării vor fi calculați și depuși într-o instrucțiune DATA (linia 20). Acești parametri indică deplasările ce trebuie calculate pentru a trasa segmentele în ordinea indicată mai sus.

P14.4

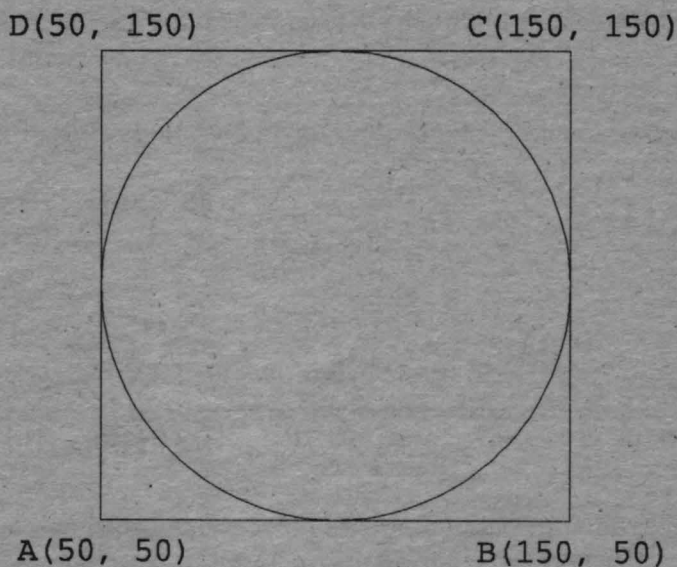
```
10 DATA -50,-90,120,0,-70,90,10,-90,25,45,-85,-45
20 CLS
30 PLOT 50,100
40 FOR i=1 TO 6
50   READ x,y
60   DRAW x,y
70 NEXT i
```

Trasarea cercurilor se execută cu ajutorul instrucțiunii:

CIRCLE x, y, r

unde x și y indică poziția centrului cercului, iar r raza acestuia.

Exemplul 14.5. Următorul program trasează un pătrat având coordonatele din figura de mai jos și în interior un cerc de centru $(100, 100)$ și rază 50.



Programul trasează pătratul desenând următoarea succesiune de segmente: AB, BC, CD, DA, iar apoi desenează cercul din interior. Coordonatele necesare instrucțiunii DRAW sînt specificate într-o instrucțiune DATA (linia 10).

P14.5

```
10 DATA 100,0,0,100,-100,0,0,-100
20 CLS
30 PLOT 50,50
35 REM se traseaza patratrul
40 FOR i=1 TO 4
```

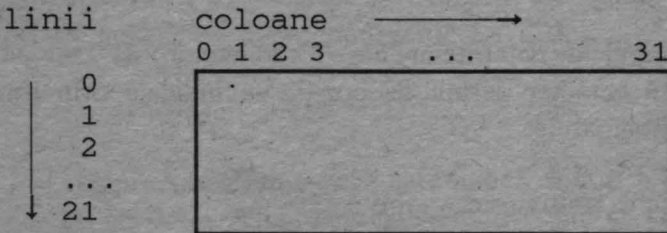
```

50 READ x,y
60 DRAW x,y
70 NEXT i
75 REM se traseaza cercul
80 CIRCLE 100,100,50

```

Trasarea figurilor geometrice devine mai sugestivă dacă poate fi completată cu posibilitatea de a nota aceste figuri ca în geometrie. Pentru aceasta trebuie făcute câteva precizări în legătură cu ecranul și cu modul de scriere și de desenare pe acesta.

Am văzut în acest capitol că instrucțiunile PLOT, DRAW, CIRCLE se referă la ecranul împărțit în 256x176 de puncte. În cazul instrucțiunilor de scriere, PRINT, ecranul este divizat în caractere în număr de 32x22 (32 coloane și 22 linii). Corespondența dintre caractere și puncte se face ținând cont de faptul că pentru a reprezenta un caracter sînt necesare 8x8 puncte. Pentru a depista corespondența exactă dintre caractere și puncte menționăm că identificarea unui caracter se face printr-o valoare indicînd linia și printr-o valoare indicînd coloana. Valorile liniilor și coloanelor sînt prezentate mai jos:

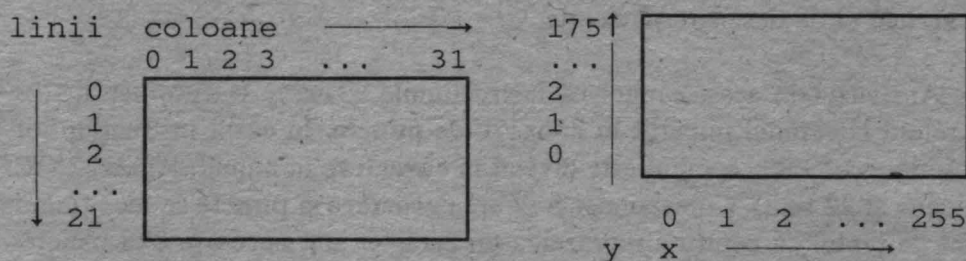


Rezultă astfel că pentru a determina poziția unui caracter, specificată printr-un număr de linie, l , și unul de coloană, c , se procedează astfel:

- ▶ din colțul din stînga sus se coboară l poziții;
- ▶ din poziția curentă se merge spre dreapta c poziții.

De exemplu, caracterul de pe linia 3 și coloana 2 se identifică prin coborîrea cu trei poziții și deplasarea la dreapta cu încă două poziții.

Se observă că indicarea pe verticală a pozițiilor caracterelor diferă de modul de indicare a acestor poziții (pe verticală) pentru puncte. În figura de mai jos specificăm alăturat cele două sisteme de identificare (a caracterelor și a punctelor):



De exemplu, caracterul de pe linia 5 și coloana 10 va conține punctele avînd coordonatele y cuprinse între 128 și 135, iar coordonatele x cuprinse între 80 și 87.

Formulele generale de determinare ale coordonatelor x și y ale punctelor ce corespund unui caracter definit de poziția sa indicată prin linia l și coloana c , sînt următoarele:

- coordonatele y sînt cuprinse între

$$8 \cdot (l-1) \quad \text{și} \quad 8 \cdot (l-1) + 7$$

- coordonatele x sînt cuprinse între

$$8 \cdot c \quad \text{și} \quad 8 \cdot c + 7.$$

Vă propunem ca **exercițiu** să determinați formulele ce indică linia și coloana unui caracter care acoperă un punct de coordonate x și y date.

Referindu-ne la exemplul 14.4 putem deduce că vîrfurile A, B, C ale triunghiului au proprietatea că aparțin caracterelor ale căror poziții indicate prin valorile liniilor și coloanelor sînt:

- ▶ pentru A (50, 100), linia=9, coloana=6;
- ▶ pentru B (0, 10), linia=20, coloana=0;
- ▶ pentru C (120, 10), linia=20, coloana=15.

Prin urmare figura desenată în exemplul 14.4 ar putea fi notată cu literele A, B, C în vîrfuri, dacă aceste caractere ar putea fi scrise pe pozițiile avînd următoarele coordonate:

- ▶ linia=8, coloana=6;
- ▶ linia=21, coloana=0;
- ▶ linia=21, coloana=15.

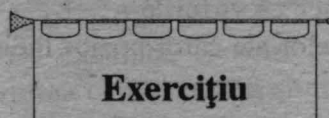
Facilitatea de a scrie direct la o poziție pe ecran ne este oferită de atributul AT asociat cu instrucțiunea PRINT astfel:

```
PRINT AT l,c; sir
```

unde l și c sînt expresii numerice care indică linia și respectiv coloana, iar sir este un șir de caractere care se scrie începînd cu poziția indicată.

Programul prezentat în exemplu 14.4 poate fi completat astfel:

```
80 PRINT AT 8,6;"A"
90 PRINT AT 21,0;"B"
100 PRINT AT 21,15;"C"
```

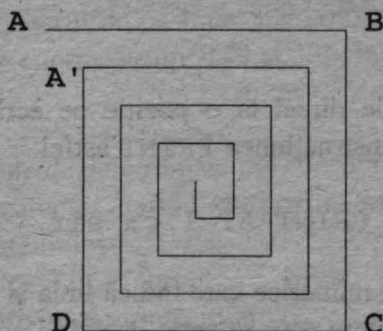


a) Verificați că punctele de coordonate $(50, 100)$, $(1, 10)$, $(100, 10)$ din exemplul 14.3. fac parte din caracterele indicate de următoarele valori de linii și coloane:

- ▶ linia=9, coloana=6;
- ▶ linia=20, coloana=0;
- ▶ linia=20, coloana=12.

b) Completați programul din exemplul 14.3 notînd vîrfurile prezentate la punctul a) în ordinea A, B și respectiv C.

Exemplul 14.6. Pentru a realiza desenul din figura de mai jos



plecînd din punctul A și urmînd sensul indicat de săgeată facem următoarele observații:

- a) punctul A are coordonatele (x, y) ; lungimea inițială a liniei este d , iar valoarea care o micșorează este p ;
- b) liniile: orizontală sus (AB) și verticală dreapta (BC) au aceeași lungime d ;

c) liniile: orizontală jos (CD) și verticală stînga (DA') au aceeași lungime $d-p$ (d fiind lungimea anterioară);

d) după ce s-a parcurs un ciclu de tipul descris la punctele b) și c), valoarea d se modifică astfel: $d=d-p$. Dacă $d > 2 \cdot p$, atunci se reia de la b), altfel STOP.

Algoritmul descris mai sus se traduce în următorul program:

P14.6

```

10 INPUT "Introduceti coordonatele ";x;y
20 INPUT "Introduceti distanta initiala ";d
30 INPUT "Introduceti pasul de micșorare ";p
40 CLS
50 PLOT x,y
60 IF d<=2*p THEN STOP
70 GO SUB 1000
80 LET d=d-p
90 GO TO 60
1000 REM subrutina ce traseaza cele 4 linii descrise
1010 REM in pasii b) si c) din algoritm
1020 DRAW d,0
1030 DRAW 0,-d
1040 LET d=d-p
1050 DRAW -d,0
1060 DRAW 0,d
1070 RETURN

```

Instrucțiunile 10-30 citesc datele de intrare x , y , d și p . Instrucțiunea 50 fixează coordonatele punctului de plecare. Instrucțiunea 70 este un apel de subrutină. Subrutina trasează 4 linii conform pașilor b) și c) din algoritm. Trasarea se face cît timp condiția $d \leq 2 \cdot p$ este adevărată.



Completați acest program astfel încît să apară pe desen și notațiile pentru vîrfurile A, B, C și D.

Realizarea desenelor devine și mai atractivă dacă acestea pot fi făcute în culori. Limbajul BASIC oferă posibilitatea de a desena colorat, iar dacă dispuneți și de un monitor color, atunci ne putem apuca de ... colorat!

Paleta de culori disponibilă este compusă din următoarele culori, în paranteză fiind trecute codurile ce vor identifica în limbaj aceste culori:

- negru (0)
- albastru (1)
- roșu (2)
- violet sau magenta (3)
- verde (4)
- albastru deschis sau cyan (5)
- galben (6)
- alb (7).

Pe ecranele alb/negru aceste culori vor fi nuanțe de gri.

Trei sînt elementele care pot fi colorate:

- ▶ **fondul**
- ▶ **bordura** (marginea ecranului)
- ▶ **desenele.**

Pentru a colora **fondul** se utilizează instrucțiunea:

PAPER nr

unde nr este o valoare între 0 și 7.

Bordura se colorează prin invocarea instrucțiunii:

BORDER nr

unde nr este o valoare între 0 și 7.

Desenele sînt executate cu instrucțiuni precum PLOT, DRAW, CIRCLE. Pentru a le trasa colorat se utilizează înaintea instrucțiunilor de mai sus, instrucțiunea:

INK nr

care prin nr indică o valoare între 0 și 7.

Exemplul 14.7. Desenăm pe o foaie galbenă, cu bordură roșie o linie albastră și una neagră:

P14.7

```
10 CLS
20 BORDER 2: REM bordura rosie
30 PAPER 6: REM fondul (foaia de desen) galben
40 INK 1: REM desen albastru
50 DRAW 100,100
60 INK 0: REM desen negru
70 DRAW 100,0
```



ATENȚIE !

După rularea unui program care stabilește anumite culori pentru desene, **TEXTUL PROGRAMULUI** (care reapare după ce se tastează CR) **APARE SCRIS IN CULOAREA DESENELOR** (ultima opțiune) **PE FOND COLORAT IN CULOAREA FONDULUI.**

Este recomandabil să nu dați culori identice sau foarte apropiate ca nuanțe pentru fond și desen, deoarece după execuția programului obțineți un text al acestuia pe care nu-l mai distingeți de fond!

Exemplul 14.8. În continuare prezentăm două variante de colorare a interiorului unui dreptunghi. Prima variantă colorează dreptunghiul prin puncte, iar a doua prin linii. Dreptunghiul are coordonatele: (10, 100), în colțul din stînga sus și (55, 10) în colțul din dreapta jos.

Varianta 1.**P14.8**

```
10 CLS
20 BORDER 2: PAPER 6: INK 1
30 PLOT 10,100: REM coltul din stanga sus
40 REM se traseaza dreptunghiul
50 DRAW 45,0
60 DRAW 0,-90
70 DRAW -45,0
80 DRAW 0,90
90 INK 0: REM culoarea pentru umplerea dreptunghiului
100 FOR x=11 TO 54
110   FOR y=11 TO 99
120     PLOT x,y
130   NEXT y
140 NEXT x
```

Varianta 2. Liniile 10-90 sînt identice cu cele din varianta 1, celelalte fiind:

```
P14.9           100 FOR x=11 TO 54
                  110   PLOT x,99
                  120   DRAW 0,-88
                  130   NEXT x
```

Observații:

◆ Remarcați diferența între vitezele de execuție ale celor două programe.

◆ Programul din varianta 1 oferă totuși mai multe posibilități de a umple (hașura) interiorul dreptunghiului. Varianta prezentată umple compact dreptunghiul. Vă propunem ca **exercițiu** să umpleți interiorul dreptunghiului cu alte hașuri. De exemplu, pe liniile pare puneți puncte numai pe pozițiile pare, iar pe liniile impare numai pe pozițiile impare.

◆ Modificați ambele variante astfel încît să umpleți interiorul dreptunghiului după cum urmează: vor fi trasate numai punctele de pe liniile pare.

◆ Modificați atît variantele 1 și 2 de mai sus cît și cele pe care le-ați conceput conform observațiilor anterioare, alegînd alte culori pentru bordură, fond și desene.

Rețineți !

■ Ecranul grafic are 256x176 puncte (pe verticală și pe orizontală).

■ Inițializarea ecranului se face cu CLS.

■ Trasarea punctelor se face cu

PLOT x, y .

■ Trasarea liniilor se realizează cu

DRAW x, y .

■ Cercurile se desenează cu

CIRCLE x, y, r .

■ Scrierea la o anumită poziție, a unui șir se face cu

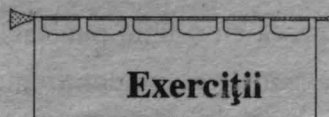
PRINT AT $l, c; \text{șir}$

■ Stabilirea culorilor se face prin instrucțiunile

BORDER nr bordura;

PAPER nr fondul;

INK nr desenul.



1. Se dau punctele $A(10, 20)$, $B(20, 20)$, $C(20, 30)$, $D(10, 30)$.
Să se scrie un program care să traseze liniile AB, BC, CD, DA.

2. Se dau punctele de coordonate $(40, 100)$, $(60, 100)$, $(80, 100)$, $(100, 100)$, $(120, 100)$, $(140, 100)$, $(160, 100)$. Să se traseze câte un cerc cu centrul în punctele specificate și de rază 10.
3. Să se traseze un triunghi dreptunghic, un triunghi obtuzunghic și unul ascuțitunghic.
4. Să se traseze un pătrat de latură 50 și să se ducă diagonalele acestuia.
5. Să se traseze un triunghi și să se unească apoi vîrfurile triunghiului cu mijloacele laturilor opuse.
6. Să se traseze un triunghi și să se ducă dintr-un vîrf înălțimea și să se unească acest vîrf cu mijlocul laturii opuse.
7. Să se traseze un pătrat, să se împartă apoi în patru pătrate egale și să se umple interiorul pătratelor din stînga sus și din dreapta jos.
8. Să se modifice programul prezentat în exemplul 14.6 astfel încît lungimea fiecărei linii să fie jumătate din lungimea celei trasate anterior. Lungimea primei linii trasate este specificată la execuție.
9. Să se traseze un pătrat și apoi să se unească mijlocul unei laturi cu vîrfurile opuse.
10. Trasați șapte cercuri de culori diferite.
11. Să se traseze un triunghi și să se construiască cu o linie punctată una din înălțimi.



Variabile structurate de tip tablou. Variabile structurate șiruri de caractere

15.1. Tablouri

În capitolul 11 am discutat noțiunea de variabilă structurată sau vector. În acest capitol vom cunoaște o structură de date mai generală pe care o vom numi **tablou** și care poate fi considerată ca fiind formată din mai multe date de tip vector așezate unele lângă altele. Dacă de exemplu, avem doi vectori de aceeași dimensiune și îi așezăm unul sub celălalt obținem un tablou cu două **linii**. Fie vectorii având componentele 1, 2, 3, 4, 5 și -1, -2, -3, -4, -5, atunci putem considera următorul tablou cu două linii:

1	2	3	4	5
-1	-2	-3	-4	-5

În tabloul de mai sus am pus deja în evidență **liniile** formate din componentele vectorilor. Putem însă să remarcăm și elementele așezate pe verticală: 1 și -1, 2 și -2, 3 și -3 etc. Aceste elemente formează câte o **coloană** în tablou.

Un alt tablou de elemente este și următorul:

1	1	2	3	4
0	2	3	4	5
1	0	2	0	1

Din exemplele și explicațiile de mai sus deducem următoarele caracteristici ale unui tablou:

- ▶ elementele așezate pe orizontală formează **liniile** tabloului
(0 2 3 4 5 formează linia a doua în tablou);
- ▶ elementele așezate pe verticală formează **coloanele** tabloului
(2 3 2 formează coloana a treia în tablou);
- ▶ toate **liniile** au același număr de elemente (în cazul ultimului exemplu, numărul de elemente este 5);
- ▶ toate **coloanele** au același număr de elemente (în cazul ultimului exemplu, numărul de elemente este 3).

Din prezentarea făcută rezultă că un vector poate fi considerat un caz particular de tablou, un tablou cu o singură linie. De exemplu:

1 2 3 -1 2

este un tablou cu o linie, sau un vector.

Numele unui tablou ca și al unui vector este format dintr-o literă și se declară cu **DIM**.

Dacă dorim ca elementele tabloului anterior să fie identificate prin numele t atunci declarăm:

DIM $t(3,5)$

Componentele lui t care corespund valorilor din exemplu sînt identificate prin:

$t(1,1), t(1,2), \dots, t(1,5)$
 $t(2,1), t(2,2), \dots, t(2,5)$
 $t(3,1), t(3,2), \dots, t(3,5)$

Exemplul 15.1. Programul citește iar apoi scrie elementele tabloului t.

P15.1

```

10 REM se citesc elementele tabloului t
20 DIM t(3,5)
30 FOR i=1 TO 3
40   FOR j=1 TO 5
50     INPUT t(i,j): REM se citesc pe linii
60   NEXT j
70 NEXT i
80 REM se scriu elementele tabloului t
90 FOR i=1 TO 3
100  FOR j=1 TO 5
110    PRINT t(i,j);"/";
120  NEXT j
130 PRINT : REM se trece pe rindul urmator
140 NEXT i

```

Următorul exemplu citește elementele unui tablou (ale cărui dimensiuni se specifică la execuție) și însumează elementele de pe linii afișând la sfârșit suma acestora.

Exemplul 15.2.

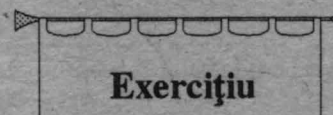
P15.2

```

10 INPUT "Introduceti nr. de linii: ";n
20 INPUT "Introduceti nr. de coloane: ";m
30 DIM x(n,m)
40 FOR i=1 TO n
50   FOR j=1 TO m
60     PRINT "x(";i;",";j;")=";:INPUT x(i,j)
65     PRINT x(i,j)
70   NEXT j
80 NEXT i
90 REM se calculeaza sumele pe linii si se afiseaza
100 FOR i=1 TO n
110  LET s=0
120  FOR j=1 TO m
130    LET s=s+x(i,j)
140  NEXT j
150  PRINT "Suma pe linia ";i;"=";s
160 NEXT i

```

Observație: Instrucțiunea de pe linia 60 va afișa $x(i, j) =$, apoi în partea de jos a ecranului - pe liniile 22, 23 ale acestuia - apare numărul ce se introduce, iar pentru ca acesta să apară după $x(i, j) =$, am inserat instrucțiunea `PRINT x(i, j)`.



Vă propunem să rețineți sumele calculate pe linii într-un vector având numărul de componente egal cu numărul de linii ale lui x , iar după ce sînt toate sumele calculate să fie afișate.

În *exemplul* următor vom muta elementele unui tablou $X(n, m)$ într-un vector $V(n*m)$ astfel:

◆ prima linie din X (elementele $X(1, 1), \dots, X(1, m)$) în primele m elemente din V ;

◆ a doua linie din X (elementele $X(2, 1), \dots, X(2, m)$) în următoarele m elemente din V ($V(m+1), \dots, V(m+m)$) etc.

Deducem din cele de mai sus că la un moment dat vom transfera elementele liniei i , $1 \leq i \leq n$, din X (elementele $X(i, 1), \dots, X(i, m)$) în elementele $V((i-1)*m+1), \dots, V((i-1)*m+m)$.

Exemplul 15.3.

P15.3

```
10 INPUT "Introduceti nr. de linii ";n
20 INPUT "Introduceti nr. de coloane ";m
30 DIM X(n,m): DIM V(n*m)
40 REM se citesc elementele lui X
```

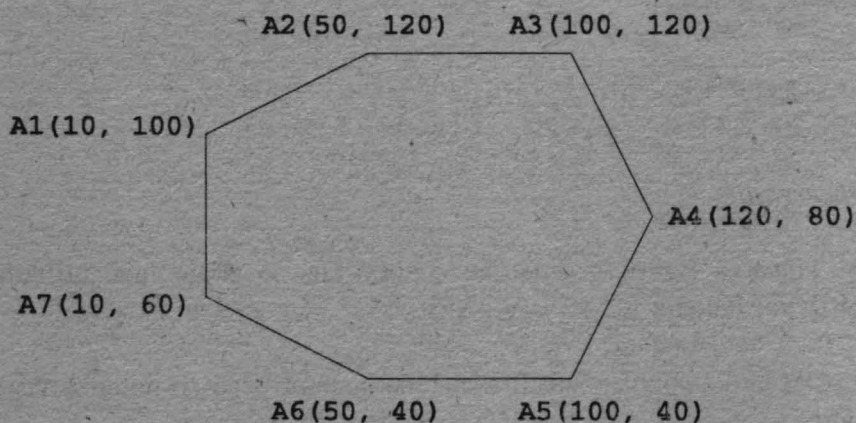


```

50 FOR i=1 TO n
60   FOR j=1 TO m
70     INPUT X(i,j)
80   NEXT j
90 NEXT i
100 REM se face transferul lui X in V
110 FOR i=1 TO n
120   FOR j=1 TO m
130     LET V((i-1)*m+j)=X(i,j)
140   NEXT j
150 NEXT i
160 REM se afiseaza V; cite m elemente pe linie
170 LET k=0
180 FOR i=1 TO n*m
190   LET k=k+1
200   PRINT V(i);"/";
210   IF k=m THEN LET k=0: PRINT
220 NEXT i

```

Vom desena acum o figură geometrică ale cărei vîrfuri au coordonate într-un tablou cu două linii; pe coloana i vom găsi pe prima linie coordonata x , iar pe a doua, coordonata y , pentru al i -lea punct. De exemplu, figura prezentată în continuare este caracterizată de succesiunea de vîrfuri A_1, A_2, \dots, A_7 , care au coordonatele trecute în paranteză.



Tabloul care conține coordonatele acestor puncte este următorul:

10	50	100	120	100	50	10
100	120	120	80	40	40	60.

Programul care trasează figura A1A2...A7 este:

P 15.4

```

10 REM se defineste tabloul f si se citesc datele
20 INPUT "Introduceti nr. de puncte ";n
30 DIM f(2,n)
40 FOR i=1 TO n
50   PRINT "x pentru A(";i;")="";:INPUT f(1,i)
55   PRINT f(1,i)
60   PRINT "y pentru A(";i;")="";:INPUT f(2,i)
65   PRINT f(2,i)
70 NEXT i
80 CLS
90 REM se traseaza figura A1A2A3...An
100 REM pentru a trasa segmentul AiA(i+1) calculam
110 REM parametrii x si y pentru DRAW astfel:
120 REM x=f(1,i+1)-f(1,i);y=f(2,i+1)-f(2,i)
130 PLOT f(1,1),f(2,1): REM se fixeaza A1
140 FOR i=1 TO n-1
150   LET x=f(1,i+1)-f(1,i)
160   LET y=f(2,i+1)-f(2,i)
170   DRAW x,y
180 NEXT i
190 REM se traseaza AnA1
200 DRAW f(1,1)-f(1,n),f(2,1)-f(2,n)

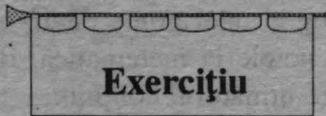
```

Observatii:

◆ Trasarea figurii geometrice de mai sus se poate face utilizând și facilitățile de culoare (INK, PAPER).

◆ Desenul obținut poate fi mai bine sugerat dacă se notează vîrfurile acestuia.

- ◆ Desenul poate fi completat prin trasarea unor diagonale (A1A3, A1A4, A1A5, A1A6, A2A4 etc.).
- ◆ Se poate umple interiorul dreptunghiului A2A3A5A6.
- ◆ Programul poate fi modificat prin înlocuirea tabloului f cu un șir de date specificate într-o instrucțiune DATA.



Transpuneți în programe observațiile de mai sus.

15.2. Variabile structurate șiruri de caractere

În capitolul 9 am introdus noțiunile de șir de caractere și variabilă de tip șir de caractere. Reamintim că o astfel de variabilă este formată dintr-o literă urmată de semnul \$. De exemplu a\$, b\$, x\$, desemnează variabile de tip șir de caractere.

La fel ca în cazul numeric și în cazul șirurilor de caractere se pot considera variabile structurate (compuse) formate din mai multe componente, fiecare de tip șir de caractere. În acest caz trebuie însă precizată **lungimea maximă** a șirurilor ce alcătuiesc componentele.

De exemplu pentru a specifica o variabilă structurată de tip șir de caractere avînd trei componente de maximum 10 caractere fiecare, trebuie folosită o instrucțiune:

```
DIM a$(3,10)
```

Dacă dorim ca în aceste trei componente să punem șirurile "Ana", "Andrei", "Maria" atunci scriem

```
LET a$(1) = "Ana"
LET a$(2) = "Andrei"
LET a$(3) = "Maria"
```

Observați că fiecare șir are cel mult 10 caractere.

Exemplul 15.4. Vom întocmi un catalog al unei grupe de 10 copii care conține numele și prenumele acestora și notele la matematică, fizică și informatică (în această ordine). Catalogul are următorul conținut:

1.	BARBULESCU CORNEL	10	9	9	(17)
2.	CORNEA VIOREL	7	7	10	(13)
3.	ENACHE PAUL	7	7	8	(11)
4.	GANEA VICTOR	10	9	10	(12)
5.	IONESCU ANDREI	10	10	10	(14)
6.	MANEA CATALIN	7	10	10	(13)
7.	NICOLAE MIHAI	9	9	10	(13)
8.	RADUCANU ION	7	10	7	(12)
9.	SABIN LIVIU	8	9	10	(11)
10.	TUDOR OCTAVIAN	10	10	9	(14)

Între paranteze, pe ultima coloană, este trecută dimensiunea șirului de caractere format cu nume și prenume (s-a considerat și poziția reprezentând spațiul dintre nume și prenume).

Acest catalog poate fi memorat în două variabile:

► o variabilă structurată șir de caractere, n\$, care să conțină numele și prenumele în ordinea de mai sus;

► o variabilă de tip tablou, s, cu 10 linii și 3 coloane care să conțină notele la matematică (în coloana 1), la fizică (în coloana 2), la informatică (în coloana 3).

Următorul program citește informațiile din catalog (nume, prenume, note la matematică, fizică și informatică) și apoi scrie acest catalog astfel încât să apară ca mai sus.

P15.5

```

10 INPUT "Introduceti dim.max. a numelui/prenum.:";d
20 INPUT "Introduceti nr. de elevi din grupa:";ne
30 DIM n$(ne,d): DIM s(ne,3)
40 REM urmeaza citirea datelor
50 FOR i=1 TO ne
60 PRINT "nume/prenume ";i;"=";:INPUT n$(i)
65 PRINT n$(i)
70 PRINT "nota mat.pt. ";n$(i);:INPUT s(i,1)
75 PRINT s(i,1)
80 PRINT "nota fiz.pt. ";n$(i);:INPUT s(i,2)
85 PRINT s(i,2)
90 PRINT "nota inf.pt. ";n$(i);:INPUT s(i,3)
95 PRINT s(i,3)
100 NEXT i
110 REM urmeaza afisarea catalogului
120 CLS
130 PRINT "****Catalogul****"
140 FOR i=1 TO ne
150 PRINT i;". ";n$(i)
160 PRINT " ";s(i,1);" ";s(i,2);" ";s(i,3)
170 NEXT i

```

Programul poate fi completat pentru a rezolva probleme ca:

- afișarea elevilor care au nota 10 la o materie (matematică sau informatică);
- afișarea elevilor care au nota 10 la matematică și la informatică;
- afișarea elevilor care au numai note de 9 și 10;
- calcularea mediei pe baza notelor la cele 3 obiecte;
- afișarea elevilor care au medii peste 8.

În continuare vom rezolva ultimele două probleme sub forma unor subrutine pe care le vom apela în programul anterior. Restul problemelor vă rămân ca **exerciții**.

P15.6

```
1000 REM subrutina pentru calcularea mediei
1010 REM introducem variabila de tip vector, m, in
1020 REM care vom calcula media
1030 DIM m(ne)
1040 FOR i=1 TO ne
1050     LET m(i)=(s(i,1)+s(i,2)+s(i,3))/3
1060 NEXT i
1070 RETURN
```

Programul **P15.5** va fi completat cu următoarele linii:

```
180 REM calcularea si afisare mediilor
190 PRINT "***Medii***"
200 GO SUB 1000
210 FOR i=1 TO ne
220     PRINT i;". ";m(i)
230 NEXT i
240 STOP
```

și, evident, cu liniile programului **P15.6**.

Afișarea elevilor care au medii peste 8 se face în corelație cu rezolvare problemei anterioare. Avem nevoie de subrutina **P15.6** de la punctul anterior, iar programul **P15.5** se completează cu următoarele linii:

```
180 REM afisarea elevilor cu medii peste 8
190 PRINT "***Lista elevilor cu medii >=8***"
200 LET nr=0
210 FOR i=1 TO ne
220     IF m(i)>=8 THEN LET nr=nr+1:PRINT nr;". ";n$(i)
230 NEXT i
240 STOP
```

și cu liniile programului **P15.6**

15.3. Alte operații cu șiruri de caractere

Operațiile de comparare a șirurilor de caractere apar foarte frecvent și de aceea este necesar să înțelegem exact cum se efectuează. Să ne reamintim că am învățat deja cum se compară șiruri de caractere formate numai din litere mari sau numai din litere mici. Mulțimea caracterelor din care se pot selecționa elemente pentru a forma șirurile conține pe lângă litere și cifre, semne speciale (+, -, *, / etc.).

Pentru a efectua comparații între șiruri de caractere trebuie să știm să facem comparații între caractere (șiruri de caractere formate dintr-un element).

Știm că se pot efectua comparații între litere deoarece acestea apar într-o anumită ordine. De exemplu:

$$"A" <= "C"$$

deoarece A se află în fața lui C;

$$"m" > "e"$$

deoarece m se află după e, iar

$$"M" < "c"$$

deoarece literele mari le preced pe cele mici. Acest ultim rezultat este o **convenție impusă de calculator**.

Celelalte simboluri care desemnează caractere (cifre, simboluri speciale) sînt și acestea ordonate. Următorul tabel conține în ordine principalele caractere utilizate în șirurile de caractere din limbajul BASIC:

!	"	\$	%	'	()	*	+	,
-	.	/							
0	1	2	3	4	5	6	7	8	9
..	;	<	=	>	?				
A	B	C	D	E	F	G	H	I	J
K	L	M	N	O	P	Q	R	S	T
U	V	W	X	Y	Z				
[]								
a	b	c	d	e	f	g	h	i	j
k	l	m	n	o	p	q	r	s	t
u	v	w	x	y	z				
{		}							

Din acest tabel deducem că următoarele comparații sînt adevărate:

"!" < "\$", "%>" < "(", "*" < "/", "<" < "=", "=" < ">", "A" < "[", "I" < "c", "m" < "{", "o" > "+", "p" > ";".

Ce puteți spune despre următoarele comparații ?

"." < "0", "7" > "(", "(" > "(", "A" < "8", "a" > "9", "{" < "}" .

Putem să considerăm și comparații între șiruri de caractere pe baza relațiilor stabilite pînă acum. Astfel:

"A1" > "A\$" - este **adevărată** deoarece "1" > "\$" este adevărată;

"Pop1" <= "Pop2" - este **adevărată** deoarece "1" < "2" este adevărată;

"123" <= "123{" - este **adevărată** deoarece primul șir coincide cu primele trei componente ale celui de al 2-lea șir;

"{{1}" >= "{{2}" - este **falsă** deoarece "1" < "2" este adevărată.

Rezultă astfel că efectuarea comparațiilor între șiruri de caractere se face după aceleași principii stabilite pentru șirurile de caractere ce conțineau numai litere, adică două șiruri se pun unul înaintea celuilalt dacă prima poziție ce conține elemente diferite în cele două șiruri, are în șirul de caractere care se pune primul, un caracter care se află în tabelul anterior înaintea caracterului corespunzător din al doilea șir de caractere.

Exemplele de mai sus evidențiază aceste observații. Pe baza acestor ordonări ale șirurilor de caractere se pot stabili valorile de adevăr ale comparațiilor.

Observații:

◆ Caracterul spațiu poate face parte din șiruri de caractere. De exemplu: "A B", " 12", "aA B".

Trebuie remarcat că acest caracter se află **înaintea tuturor** elementelor din tabelul prezentat. Astfel următoarele comparații sînt adevărate:

" " < "!", " " < "A", " " <= "1", "A " <= "AA", "A b" < "Ab".

◆ Analizînd tabelul caracterelor observăm că acesta conține și caracterul " care este folosit pentru a delimita șirurile de caractere. Atunci cînd se dorește utilizarea caracterului " într-un șir de caractere este obligatorie **dublarea sa**. De exemplu pentru a construi șirul A"B, luăm "A""B", iar pentru A""B, luăm "A""""B".

În exemplele de mai sus am utilizat în șirurile de caractere cifrele de la 0 la 9. Distingem astfel

numerele: 123, 12, 3

și

șirurile de caractere: "123", "12", "3".

Să ne reamintim că numerele pot fi adunate, înmulțite, comparate, atribuite, citite, scrise, pe când șirurile de caractere pot fi doar atribuite, comparate, concatenate, citite, scrise.

Exemplul 15.5. Programul ilustrează utilizarea variabilelor numerice și a celor de tip șir de caractere.

P15.7

```
10 PRINT "123"
20 PRINT 123
30 LET a=0: LET b=1
40 LET a=a+b+1
50 LET x$="12": LET y$="A"
60 PRINT a+b;" "+x$+y$
70 IF a<b AND x$<y$ THEN PRINT "OK"
```

Există posibilitatea de a trece de la un număr la un șir de caractere și invers.

Trecerea de la un număr la un șir de caractere se face cu

STR\$ nr

unde nr este un număr, iar rezultatul este șirul de caractere corespunzător. De exemplu:

```
STR$ 125 = "125"
STR$ 3 = "3"
STR$ 2+"1" = "21"
"1"+STR$ 45+"+" = "145+"
```

Operația inversă, de obținere a valorii numerice a unui șir de caractere (având drept componente caracterele "0" la "9") se realizează cu:

VAL sir

unde sir este un șir de caractere. De exemplu:

```

VAL "15" = 15
VAL "15"+1 = 16
1+VAL "125"-2 = 124.

```

O altă informație utilă referitoare la un șir de caractere este lungimea acestuia care poate fi aflată prin:

```
LEN sir
```

unde `sir` este un șir de caractere, iar rezultatul este lungimea acestuia. De exemplu:

```

LEN "ABC" = 3
LEN "125AB" = 5

```

Observații:

◆ `LEN`, `VAL` și `STR$` se tastează trecând în modul `E` (`CS` și `SS` tastate simultan).

◆ `LEN`, `VAL` și `STR$` se aplică și variabilelor, după cum rezultă și din exemplul următor:

Exemplul 15.6. Programul calculează lungimea unui șir care se citește; dacă șirul este format dintr-o cifră atunci, dacă este "0" sau "9", se scrie, iar dacă nu, se scrie șirul format din cifra anterioară și cea care urmează (de exemplu pentru "3" se scrie "24"). Pentru celelalte cazuri se scrie șirul introdus.

P15.8

```

10 REM se citeste sirul de caractere
20 INPUT a$
30 REM se testeaza daca lungimea este 1
40 IF LEN a$=1 THEN GO TO 70
50 PRINT "Sirul "+a$+" de lungime ";LEN a$
60 STOP
70 IF a$>"0" AND a$<"9" THEN GO TO 100

```

```
80 PRINT a$+" Are lungimea 1"  
90 STOP  
100 REM a$ contine o cifra diferita de "0" sau "9"  
110 LET b$=STR$(VAL a$-1)+STR$(VAL a$+1)  
120 PRINT b$
```

Transformarea unui număr într-un șir de caractere și invers se poate face într-un mod mai general decât cel de sus.

Să presupunem că pentru fiecare caracter există o valoare numerică ce-i corespunde și care are legătură cu poziția caracterului în tabelul caracterelor pe care le-am prezentat. Astfel dacă în acel tabel un caracter x se află înaintea altui caracter y , atunci valoarea numerică asociată lui x este mai mică decât cea asociată lui y .

Fiind dat un caracter x , valoarea sa numerică v , este dată prin următoarea operație:

$$v = \text{CODE } x .$$

De exemplu:

$$\text{CODE "a"} = 97$$

$$\text{CODE "b"} = 98$$

$$\text{CODE "A"} = 65$$

$$\text{CODE " "} = 32$$

Se poate face și o operație inversă, de determinare a caracterului x , ce corespunde unei valori numerice v , prin:

$$x = \text{CHR\$ } v .$$

De exemplu:

$$\text{CHR\$ } 97 = \text{"a"}$$

$$\text{CHR\$ } 98 = \text{"b"}$$

$$\text{CHR\$ } 32 = \text{" "}$$

Observații:

◆ CODE și CHR\$ se tastează trecînd în modul E (CS+SS tastate simultan).

◆ CODE și CHR\$ se aplică și variabilelor de tip șir de caractere și respectiv numerice, după cum rezultă și din exemplul anterior:

Exemplul 15.7. Următorul program afișează literele de la "a" la "f".

P15.9

```
10 LET vi=CODE "a": LET vf=CODE "f"
20 FOR i=vi TO vf
30 PRINT CHR$ i
40 NEXT i
```

 Rețineți !

■ Tablourile sînt variabile structurate avînd un număr specificat de linii și coloane. Un tablou *t* cu *n* linii și *m* coloane se introduce prin

$$\text{DIM } t(n, m)$$

■ Variabilele structurate cu elemente șiruri de caractere sînt caracterizate de numărul de elemente și de lungimea maximă a șirului. O variabilă structurată *a\$* avînd *n* șiruri de caractere, fiecare de lungime maximă *m*, se introduce prin

$$\text{DIM } a\$(n, m)$$

■ Lungimea unui șir de caractere *s*, se obține prin

$$\text{LEN } s$$

■ Valoarea numerică a unui șir de caractere conținând caractere de tip cifre se calculează cu

VAL s

■ Șirul de caractere corespunzător unui număr n se obține cu

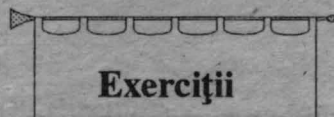
STRȘ n

STRȘ și VAL sînt funcții inverse una alteia.

■ Asocierea dintre un caracter x și o valoare numerică v se face cu

$v = \text{CODE } x$ și $x = \text{CHRȘ } v$.

CODE și CHRȘ sînt funcții inverse una alteia.



1. Să se scrie un program care să citească un tablou $t(3, 5)$ pe linii și să-l scrie pe coloane (fiecare coloană pe cîte un rînd).

2. Să se scrie un program care să citească două tablouri $t(2, 2)$, $p(2, 2)$ și să scrie tabloul $o(2, 2)$ obținut din cele două astfel: $o(i, j) = t(i, j) - p(i, j)$, dacă $t(i, j) > p(i, j)$ și $o(i, j) = 0$, în caz contrar.

3. Să se scrie un program care citește un tablou $t(2, 3)$ și scrie un tablou $p(3, 2)$ care are următoarea proprietate: liniile din p sînt coloane în t , iar coloanele în p sînt linii în t . De exemplu dacă

t: 0 1 2
1 2 2

atunci

$$p: \begin{matrix} 0 & 1 \\ 1 & 2 \\ 2 & 2 \end{matrix}$$

4. Să se scrie un program care citește un tablou $x(4,5)$ și mută elementele sale într-un vector $v(20)$ astfel: se mută întâi prima coloană, apoi a doua etc. Să se afișeze x și v .

5. Să se scrie un program care citește un tablou $p(5,5)$ și determină cel mai mare element al său și poziția sa în tablou. Aceeași problemă pentru cel mai mic element din tablou.

6. Să se scrie un program care citește un tablou $t(n,m)$ cu n și m introduse într-o instrucțiune DATA, și determină câte elemente mai mici ca 0 se află în tabloul t .

7. Să se scrie un program care să citească un tablou $t(4,4)$ și să scrie liniile tabloului t astfel: prima linie și a treia de la stînga la dreapta, iar a doua și a patra în sens invers. Generalizare pentru un tablou $t(2n,k)$, cu n și k citite cu INPUT.

8. Să se scrie un program care citește un tablou $t(m,n)$ cu elemente pozitive și cu n, m citite cu INPUT. Să se treacă într-un vector v următoarele elemente ale lui t : de pe liniile pare cele care se află pe coloane numere pare, iar de pe liniile impare cele care se află pe coloane numere impare. Tabloul se parcurge pe linii începînd cu prima. Dimensiunea vectorului v se va calcula în funcție de m și n . De exemplu pentru tabloul $t(3,5)$

$$\begin{matrix} 1 & 2 & 5 & 7 & 1 \\ 2 & 1 & 7 & 2 & 2 \\ 3 & 1 & 4 & 5 & 3 \end{matrix}$$

vectorul v va conține următoarele elemente: 1 5 1; 1 2; 3 4 3.

9. Să se scrie un program care, pentru un tablou dat, să calculeze:

- a) suma elementelor de pe liniile pare (notată sa)
- b) suma elementelor de pe liniile impare (notată sb)

iar pentru $sb \neq 0$ să se calculeze sa/sb .

10. Să se scrie un program care să afișeze pozițiile elementelor dintr-un tablou (linia și coloana) care sînt nule. Parcurgerea se va face pe linii.

11. Fiind dat un tablou t cu n linii și n coloane, elementele $t(1,1)$, $t(2,2)$, ..., $t(n,n)$ definesc **diagonala principală** a tabloului. Să se scrie un program care să afișeze elementele care se află într-un tablou deasupra diagonalei principale. Parcurgerea se va face pe linii. Aceeași problemă pentru elementele de sub diagonala principală. De exemplu pentru tabloul

```

1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5
1 2 3 4 5

```

diagonala principală conține elementele 1 2 3 4 5, iar lista elementelor care se află deasupra acesteia este următoarea:

2 3 4 5; 3 4 5; 4 5; 5

12. Să se scrie un program care, pentru un tablou dat, să calculeze următoarele sume:

- a) pentru elementele pozitive de pe linii (notată lp);
- b) pentru elementele negative de pe linii (notată ln);
- c) pentru elementele pozitive de pe coloane (notată cp);
- d) pentru elementele negative de pe coloane (notată cn);

iar dacă $cp+cn \neq 0$, atunci să se calculeze și $(lp+ln)/(cp+cn)$.

13. Să se scrie un program pentru întocmirea unui catalog care conține numele și prenumele a 5 elevi precum și notele la 4 materii.

a) Să se afișeze elevii și notele acestora.

b) Să se afișeze elevii care au note maxime la cel puțin 2 materii.

14. Să se afle cel mai mare element dintr-o mulțime de șiruri de caractere de lungime maximă 20; mulțimea are 10 elemente.

15. Să se scrie un program care să citească o succesiune de șiruri de caractere și să afișeze lungimea acestora.

16. Să se scrie un program care citește două caractere și afișează apoi toate caracterele care au valorile numerice asociate cuprinse între valorile numerice ale caracterelor citite. De exemplu dacă se citesc "a" și "c", atunci se afișează "a" "b" "c".

17. Să se scrie un program care citește două caractere pentru care diferența valorilor numerice asociate este cel mult 5 și afișează șirul de caractere format din caracterele care au valorile numerice cuprinse între valorile numerice ale celor două caractere citite. De exemplu, dacă se citesc "1" și "4" atunci se afișează "1234".

18. Rezolvați:

LEN "LEN"=?; LEN "1" + 1=?; VAL "125" + 1=?; STR\$ 100 + 1=?; STR\$ (100+1)=?; "1" + STR\$ 1=?; CHR\$ CODE "a"=?; CODE CHR\$ 97=?.



16

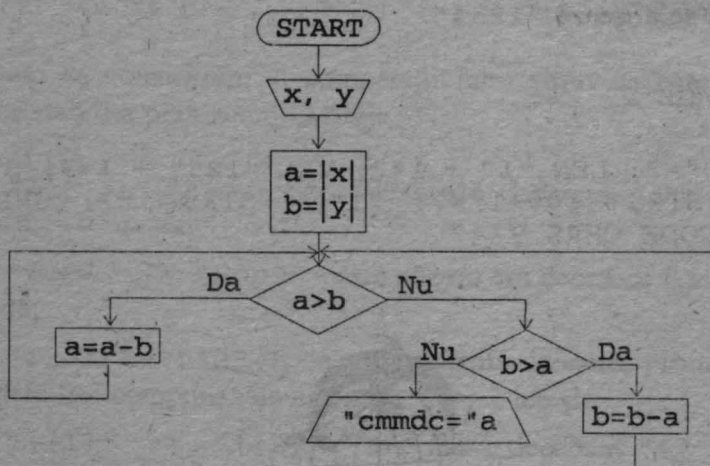
Probleme rezolvate și propuse

În acest capitol vom rezolva câteva probleme la care vom recapitula cunoștințele de BASIC învățate pînă acum și vom utiliza rezultate matematice dobîndite pînă în prezent.

16.1. Determinarea celui mai mare divizor comun

Problema determinării celui mai mare divizor comun pentru două numere întregi x și y (pe scurt $\text{cmmdc}(x, y)$) a mai fost discutată și un program în limbajul BASIC a fost conceput în paragraful 10.5 folosind algoritmul lui Euclid.

Prezentăm acum schema logică a unui alt algoritm de determinare a $\text{cmmdc}(|x|, |y|)$, pentru două numere întregi x și y :



Acest algoritm de o mare simplitate și eleganță poate fi "tradus" într-un program (renunțăm la prezentarea argumentelor matematice care justifică faptul că algoritmul conduce la $\text{cmmdc}(|x|, |y|)$). Deoarece vom mai utiliza acest algoritm și în alte probleme din acest capitol îl vom scrie sub forma unei subrutine astfel:

P 16.1

```
1000 REM Subrutina de determinare a cmmdc(|x|, |y|)
1010 LET a=ABS x : LET b=ABS y
1020 IF a>b THEN LET a=a-b : GO TO 1020
1030 IF b>a THEN LET b=b-a : GO TO 1020
1040 cmmdc=a
1050 RETURN
```

Un program în care folosim această subrutină este următorul:

P16.2

```
10 INPUT x; y
20 GO SUB 1000
30 PRINT cmmdc
40 STOP
```

16.2. Operații cu numere raționale

Sînt cunoscute din matematică regulile după care se adună, scad, înmulțesc și împart numerele raționale. În acest paragraf vom concepe programe pentru operațiile amintite cu numere raționale și de asemenea pentru simplificarea fracțiilor (punerea numerelor raționale sub forma unei fracții ireductibile). Obținerea formei ireductibile a unui număr rațional x/y se realizează prin simplificare cu $\text{cmmdc}(|x|, |y|)$.

Următorul program realizează operațiile de adunare, scădere, înmulțire și împărțire a două numere raționale x/y și x_p/y_p . Rezultatele vor fi puse sub formă ireductibilă. Programul solicită una din următoarele opțiuni:

a - adunare
s - scădere
m - înmulțire
d - împărțire
f - stop

P 16.3

```
10 REM Se introduce optiunea pentru operatii
20 INPUT "Optiunea (a;s;m;d;f): ";o$
30 IF o$="a" THEN GO TO 200
40 IF o$="s" THEN GO TO 300
50 IF o$="m" THEN GO TO 400
60 IF o$="d" THEN GO TO 500
70 IF o$="f" THEN PRINT "STOP" : STOP
80 PRINT "Eroare optiune" : GO TO 20
200 REM Adunare
210 GO SUB 2000 : REM Subrutina de introducere nr.
220 LET x=x*yp+xp*y : REM Numaratorul rezultatului
230 LET y=y*yp : REM Numitorul rezultatului
240 GO SUB 1000 : REM Subrutina pentru cmmdc
250 PRINT x/cmmdc;"/";y/cmmdc
260 GO TO 20
300 REM Scadere
310 GO SUB 2000
320 LET x=x*yp-xp*y
330 LET y=y*yp
340 GO SUB 1000
350 GO TO 250 : REM Afisare rezultat
400 REM Inmultire
410 GO SUB 2000
420 LET x=x*xp
430 LET y=y*yp
440 GO SUB 1000
450 GO TO 250 : REM Afisare rezultat
500 REM Impartire
510 GO SUB 2000
520 LET x=x*yp
530 LET y=xp*y
540 GO SUB 1000
550 GO TO 250 : REM Afisare rezultat
```



```

1000 REM Subrutina pentru cmmdc
... a se vedea paragraful anterior
2000 REM Introducerea datelor
2010 INPUT "Numaratorul primului nr: ";x
2020 INPUT "Numitorul primului nr: ";y
2030 INPUT "Numaratorul pentru al 2-lea nr: ";xp
2040 INPUT "Numitorul pentru al 2-lea nr: ";yp
2050 RETURN

```

Vă propunem ca **exercițiu** să modificați acest program astfel încât să testeze dacă numitorii sînt nenuli (variabilele y , yp) și afișarea să fie efectuată de o subrutină (afișarea de la linia 250).

16.3. Valoarea unui polinom. Schimbarea bazei de numerație

Pentru un polinom

$$5 \cdot X^3 + 2 \cdot X^2 - X + 1 \quad (1)$$

a-i calcula valoarea pentru $X=2$ înseamnă a înlocui în expresia care definește polinomul pe X cu 2, deci:

$$5 \cdot 2^3 + 2 \cdot 2^2 - 2 + 1 = 47$$

Același polinom se poate pune sub forma

$$((5 \cdot X + 2) \cdot X - 1) \cdot X + 1 \quad (2)$$

iar valoarea sa în 2 se va calcula conform expresiei

$$((5 \cdot 2 + 2) \cdot 2 - 1) \cdot 2 + 1$$

Algoritmul pentru calcularea valorii polinomului într-un punct va pleca de la forma (2).

Calculule care se fac pentru această formă și pentru $X=2$ pot fi descrise prin următoarele relații:

$$y=5 \cdot 2+2 \quad (2.1)$$

$$z=y \cdot 2-1 \quad (2.2)$$

$$u=z \cdot 2+1 \quad (2.3)$$

În general, un polinom se scrie sub forma

$$a_n X^n + a_{n-1} X^{n-1} + \dots + a_1 X + a_0 \quad (1)$$

iar forma echivalentă este

$$(\dots ((a_n X + a_{n-1}) \cdot X + \dots + a_1) \cdot X + a_0 \quad (2)$$

Valoarea acestuia pentru $X=v$ se obține înlocuind pe X cu v .

Relațiile după care se calculează valoarea acestui polinom pentru $X=v$ sînt următoarele:

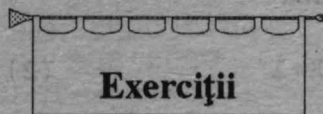
$$y_1 = a_n \cdot v + a_{n-1} \quad (2.1)$$

$$y_2 = y_1 \cdot v + a_{n-2} \quad (2.2)$$

$$y_3 = y_2 \cdot v + a_{n-3} \quad (2.3)$$

...

$$y_n = y_{n-1} \cdot v + a_0 \quad (2.n)$$



1. Considerați polinomul:

$$5X^4 - 2X^3 + X^2 - X + 1 \quad (1')$$

și puneți-l în forma (2') așa cum polinomul anterior a fost adus de la forma (1) la forma (2).

2. Scrieți relații similare cu (2.1), (2.2), (2.3), pentru forma (2') a polinomului de mai sus și pentru $X = -1$.

Cea mai mare putere a lui X într-un polinom, numită **gradul polinomului**, ne indică numărul de relații ce definesc valoarea polinomului adus la forma (2), pentru o anumite valoare v a lui X .

Analizând forma (2) a polinomului și relațiile (2.1) - (2.3) ca și forma și relațiile echivalente ale polinomului din exercițiul de mai sus, putem ușor înțelege programul care urmează:

P 16.4

```
10 REM Calcularea valorii unui polinom
20 INPUT "Introduceti gradul polinomului: ";n
30 INPUT "Introduceti valoarea lui x: ";x
40 READ a
50 FOR i=n-1 TO 0 STEP -1
60   READ b
70   LET a=a*x+b : REM rel. (2.1) - (2.n)
80 NEXT i
90 PRINT a
100 DATA 5, 2, -1, 1
```

Coefficienții polinomului (numerele care înmulțesc pe X^3 , X^2 , X , X^0) apar în instrucțiunea DATA. Evident schimbarea polinomului implică și modificarea acestei instrucțiuni.

Programul anterior se poate utiliza și pentru a trece un număr k scris într-o bază b , în baza 10.

Într-adevăr, se știe că un număr k având cifrele k_n , k_{n-1} , \dots , k_1 , k_0 , în această ordine, se poate scrie astfel:

$$k = k_n \cdot b^n + k_{n-1} \cdot b^{n-1} + \dots + k_1 \cdot b + k_0.$$

Este evidentă asemănarea scrierii lui k în baza b , cu determinarea valorii unui polinom, pentru valoarea b . Vă lăsăm ca **exercițiu** să scrieți polinomul din care se obține k pentru $X=b$.

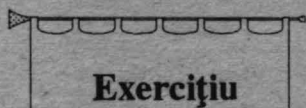
Executînd programul anterior pentru un anume k (număr în baza b), valoarea a , care se afișează reprezintă scrierea lui k în baza 10.

În informatică se folosește destul de frecvent scrierea unui număr în baza 2. Din acest motiv, dar și pentru faptul că este în sine o problemă interesantă, vom aborda chestiunea trecerii unui număr scris în baza 10 la scrierea sa în baza 2.

De *exemplu*: $5_{10}=101_2$; $10_{10}=1010_2$.

Un algoritm simplu care rezolvă problema scrierii lui x (număr în baza 10) ca număr în baza 2 este următorul:

- a) Se face $i=1$;
- b) Se împarte x la 2 și rezultă un cît, c și un rest, r_i ;
- c) Dacă $c > 2$, atunci se face $x=c$ și $i=i+1$, apoi se merge la b);
- d) În caz contrar (adică $c < 2$) obținem reprezentarea în baza 2 ca fiind formată din următoarele cifre: $c, r_1, r_{1-1}, \dots, r_1$.



Aplicați acest algoritm pentru $x=10$.

Pentru a reține cifrele scrierii lui x în baza 2, avem nevoie de un vector notat r . Pentru a determina dimensiunea acestui vector folosim următoarea

Observație: Numărul n de componente ale vectorului r , satisface relația:

$$2^{n-1} \leq x < 2^n.$$

De exemplu:

$$2^2 \leq 5 < 2^3, \quad 5_{10} = 101_2; \quad 2^3 \leq 10 < 2^4, \quad 10_{10} = 1010_2.$$

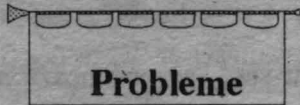
Cu precizările de mai sus, algoritmul prezentat poate fi tradus în următorul program:

P 16.5

```

10 REM Trecerea lui x in baza 2
20 INPUT "Introduceti x: ";x
30 INPUT "Introduceti n: ";n
40 DIM r(n)
50 LET i=1
60 LET r(i)=x-INT(x/2)*2 : REM Restul
70 LET x=INT(x/2) : REM Citul
80 IF x>=2 THEN LET i=i+1: GO TO 60
90 REM Ultimul cit este mai mic ca 2
100 LET i=i+1
110 LET r(i)=x
120 FOR j=i TO 1 STEP -1
130 PRINT r(j);
140 NEXT j

```



1. Să se rezolve exercițiul din paragraful 16.2.
2. Să se modifice programul de calculare a valorii unui polinom (paragraful 16.3) astfel încât coeficienții polinomului să fie introduși cu instrucțiunea INPUT.
3. Să se scrie un program care să calculeze valoarea unui polinom folosind forma (1) a acestuia (paragraful 16.3).

4. Să se modifice programul de trecere a unui număr din baza 10 în baza 2, pentru a trece în baza 3. Modificați programul astfel încât baza în care se trece să fie introdusă prin INPUT.

5. Se dă un șir de numere întregi. Să se aranjeze acest șir astfel încât pe primele poziții să apară numere pare, iar apoi cele impare. De exemplu, 1 2 -1 0 3 4, se transformă în 2 0 4 1 -1 3.

6. Se dă un șir de numere întregi. Să se ordoneze șirul astfel încât primele poziții să conțină numere pare ordonate crescător, iar apoi să conțină numere impare ordonate descrescător. De exemplu, 1 2 -1 0 3 -2, se transformă în -2 0 2 3 1 -1.

7. Se dă un șir de numere întregi. Să se aranjeze acest șir astfel încât primele poziții să conțină numere negative, iar următoarele, numere pozitive. De exemplu, 1 2 -1 3 0 -2 4, se transformă în -1 -2 1 2 3 0 4.

8. Se dă un șir a_1, a_2, \dots, a_n și o valoare k , $1 < k \leq n$. Să se ordoneze șirul astfel:

$$a_k \ a_{k+1} \ \dots \ a_n \ a_1 \ a_2 \ \dots \ a_{k-1}.$$

De exemplu, șirul 1 2 3 -4 0 2, pentru $k=3$, se ordonează astfel: 3 -4 0 2 1 2.

9. Să se genereze toate numerele de 5 cifre scrise în baza 2 (numerele de 3 cifre în baza 2 sînt: 100, 101, 110, 111).

10. Să se generalizeze problema 9 pentru numere de n cifre în baza 2.

11. Se dă un tablou cu 5 linii și 5 coloane. Să se afișeze elementele acestui tablou astfel: întâi linia de sus, apoi coloana din dreapta, urmează linia de jos (de la dreapta la stînga) și la sfîrșit coloana din stînga (de jos în sus);

tabloul cu 4 linii și 4 coloane ce nu a fost listat încă este afișat în același mod; apoi cel de 3 linii și 3 coloane, pînă se afișează toate elementele tabloului. De exemplu tabloul de 4 linii și 4 coloane

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

se listează astfel:

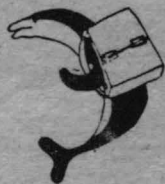
1 -1 2 3; 4 5 6; -1 2 1; 2 3; 7 4; 2 1

12. Să se generalizeze problema pentru un tablou cu n linii și n coloane.

13. Se dă șirul $1\ 2\ 3\ \dots\ n\ -1\ -2\ \dots\ -n$. Să se ordoneze șirul astfel încît să devină $1\ -1\ 2\ -2\ 3\ -3\ \dots\ n\ -n$

14. Se dă șirul $1\ 2\ 3\ \dots\ n\ -1\ -2\ \dots\ -n$. Să se ordoneze șirul astfel încît să devină $1\ -n\ 2\ -(n-1)\ \dots\ n\ -1$

15. Se dă un șir de numere întregi aflat într-un vector. Se cere să se obțină șirul numerelor pozitive. De exemplu șirul $-1\ 2\ 3\ -2\ -3\ 4$ se transformă în $2\ 3\ 4$. Găsiți o soluție care nu folosește un alt vector.



Limbajul GW-BASIC

Diferențe față de limbajul prezentat

Acest capitol este dedicat prezentării limbajului GW-BASIC disponibil pe calculatoarele compatibile IBM-PC (pe scurt PC-uri) sub sistemul de operare MS-DOS. Prezentarea de față este înțeleasă numai dacă au fost parcurse capitolele precedente. Limbajul pe care-l prezentăm se numește GW-BASIC, iar cel pe care l-am utilizat pînă aici îl vom identifica prin BASIC.

17.1. Principalele comenzi

■ Lansarea limbajului GW-BASIC se face prin comanda:

GWBASIC

După lansare, pe ecran apare în parte de jos un **meniu** care cuprinde principalele comenzi, iar pe prima linie va fi scris cuvîntul

OK

care se numește **prompt**.

■ Pentru a tasta un program facem următoarele precizări:

Tastatura este alcătuită din următoarele elemente:

- litere mari și mici
- cifrele 0 la 9
- simboluri speciale (<, >, =, :, ! etc.)
- tasta **SPACE**, pentru introducerea caracterului spațiu

Introducerea numelor de variabile, a instrucțiunilor, etichetelor, se face element cu element; de *exemplu*, pentru a introduce:

```
10 INPUT x
```

se apasă succesiv pe 1, 0, **SPACE**, I, N, P, U, T, **SPACE**, x, **CR** (unde prin **CR** am identificat tasta funcțională **CR** sau **ENTER** sau **RETURN**).

- taste funcționale:

CR (sau **ENTER** sau **RETURN**)

F1, F2, ... , F10 (eventual **F11, F12**)

SHIFT (cînd este ținută apăsată transmite litere mici și caracterele scrise în partea de sus a tastelor)

INS, DEL

CTRL (sau **CONTROL**)

BREAK

ALT

săgețile (→, ←, ↑, ↓)

Introducerea unui program se poate face în două moduri:

manual: prin tastarea element cu element așa cum am văzut la punctul anterior

automat: prin introducerea comenzii

AUTO n, m

unde n este numărul primei linii de program, iar m este incrementul; ambele elemente sînt opționale, prin lipsă avînd valorile 10. După ce se termină introducerea programului se tastează **CTRL** și **C** sau **CTRL** și **BREAK** (tastate simultan). În acest mod de introducere, numerele de linie sînt furnizate de limbaj.

Exemplu: Comanda

AUTO 100,50

generează următoarele numere de linii: 100, 150, 200, etc.

■ **Editarea unui program se face astfel:**

• **selectarea liniei** ce urmează a fi corectată se face în două moduri:

- fie prin acționarea săgeților în "sus" sau în "jos" pînă se ajunge pe linia vizată

- fie prin tastarea comenzii

EDIT n

unde n este numărul liniei vizate.

• **determinarea poziției din linie** în care se face modificarea se obține prin tastarea săgeților "stînga" și "dreapta".

• **inserarea unor caractere** la poziția curentă se face astfel: se tastează **INS** și apoi se introduc caracterele vizate. De exemplu în linia

10 LET A=1

dorim să inserăm după A, caracterele B și C; ajunși în poziția =, se tastează:

INS

B C CR

• **ștergerea** unor caractere se face prin apăsarea tastei **DEL**, care la fiecare apăsare șterge caracterul de la poziția curentă. După terminarea ștergerii se tastează **CR**.



Important! Orice modificare (inserare, ștergere) se termină cu apăsarea tastei **CR**.

■ Principalele comenzi privind dinamica programelor:

F1 - listarea programului

F2 - execuția programului

F3 - încărcarea unui program în memorie. Se tastează:

F3 "nume"

unde "nume" este numele programului.

F4 - salvarea unui program. Se tastează:

F4 "nume"

SYSTEM - părăsirea dialogului cu limbajul GW-BASIC.

Exemplul 17.1.

■ Lansarea GW-BASIC se face cu

GW BASIC

■ Introducerea unui program, în modul automat, se realizează tastînd întîi, de exemplu:

```
AUTO 100,10
```

și apoi se scriu liniile de program

```
P17.1          100 INPUT A,B
                110 PRINT A,B
                120 PRINT A+B
                130 STOP
                140 CTRL C
```

Numerele de linii 100, 110, 120, 130, 140, sînt introduse automat, ca urmare a comenzii AUTO 100,10.

■ Listarea programului tastat anterior se obține prin comanda

```
F1
```

■ Executarea programului se face prin tastarea comenzii

```
F2
```

■ Salvarea programului sub numele "PROG1" se face cu

```
F4 "PROG1"
```

■ Încărcarea programul "PROG1" se face cu

```
F3 "PROG1"
```

■ Modificare programului prin adăugarea unei noi variabile, C, la liniile 100, 110, înlocuirea operației + din linia 120, cu * și inserarea unei linii noi care să afișeze (A-B)*C se face cu :

```
EDIT 100
```


urmează apoi deplasare pînă la sfîrșitul liniei și tastarea următoarelor caractere

C CR

obținîndu-se

100 INPUT A,B,C

În mod asemănător se obține și linia

110 PRINT A,B,C

Pentru a modifica în linia 120, +, cu * se procedează astfel:

EDIT 120

apoi ne deplasăm pînă sub + după care se tastează DEL și apoi INS; urmînd a se tasta:

*

Introducerea unei linii care să afișeze $(A-B)*C$ se obține tastînd, de exemplu:

125 PRINT (A-B)*C

Dacă dorim să ștergem, de exemplu, linia 120, atunci se tastează

120 CR

și linia va fi distrusă.

17.2. Etichete, constante, variabile, expresii

Etichetele, numerele ce preced instrucțiunile, au valori cuprinse între 0 și 65529.

Pe aceeași linie pot fi puse mai multe instrucțiuni despărțite cu : .

Constantele sînt de două tipuri: **numerice** și **șiruri de caractere**.

Exemple de constante numerice:

25, -16, 128

care sînt **întregi** (domeniul acestor fiind între -32768 și 32767) și

17.1, -18.02, 18.53E-7, -47.033E+4

care sînt **reale** (domeniul acestora fiind cuprins între $3 \cdot 10^{-39}$ și $1.7 \cdot 10^{38}$)

Menționăm că limbajul GW-BASIC conține și constante numerice **hexazecimale**, cu prefixul **&H** și **octale**, cu prefixul **&** sau **&O**. De exemplu:

&H10, &H0F, &12, &O023

care reprezintă constantele zecimale 16, 15, 10 și respectiv 19.

Constantele de tip **șir de caractere** sînt șiruri de maximum 255 caractere. De exemplu:

"ABC", "GW-BASIC".

Variabilele sînt identificate cu nume de orice lungime, dintre care maximum 40 sînt semnificative.

Variabilele sînt **numerice** sau de tip **șir de caractere**, ultimele terminîndu-se cu semnul \$.

Spre deosebire de BASIC, în GW-BASIC un nume de variabilă poate să conțină și simbolul ? și nu poate fi un nume ce identifică o instrucțiune sau comandă.

Exemple:

ION, I, ALFA

sînt variabile numerice, iar

A\$, POP\$, DAN\$

sînt variabile de tip șir de caractere.

Variabilele **structurate** și variabilele de **ciclare** sînt nume ca mai sus. De **exemplu:**

```
10 DIM AC(10)
20 DIM VARIABILA(10,2)
30 FOR ION=1 TO 10
40   LET AC(ION)=ION
50 NEXT ION
```

Într-o instrucțiune DIM pot să apară mai multe variabile.

Variabilele numerice în GW-BASIC pot fi clasificate în variabile **întregi** (cu sufixul %) și variabile **reale** (cu sau fără sufixul !). De exemplu:

A%, BAC%, Tot%

sînt variabile întregi (în care se află valori numerice de tip numere întregi), iar

B!, BEB, Bob

sînt variabile reale (conțin valori numerice reale).

Expresiile sînt alcătuite din constante, variabile sau constante/variabile legate prin diferite operații.

Operațiile se divid în următoarele grupe:

- aritmetice
- relaționale
- logice

Operațiile aritmetice sînt: ridicare la putere (^), înmulțirea (*), împărțirea (/), adunarea (+), scăderea (-), împărțirea întregă (\), restul împărțirii întregi (mod).

Operațiile relaționale sînt: egalitatea (=), mai mic (<), mai mare (>), mai mic sau egal (<=), mai mare sau egal (>=), neegal (<>).

Operațiile logice sînt: NOT, AND, OR, XOR.

Operațiile permise pentru șiruri de caractere sînt: **atribuirea**, **comparațiile** (=, <, >, <=, >=, <>) și **concatenarea** (+).

Menționăm în plus:

SQR (x)	rădăcina pătrată din numărul pozitiv x;
ABS (x)	valoarea absolută a numărului x;
INT (x)	partea întreagă a numărului x;
CHR\$ (x)	convertește valoarea întreagă x, în caracterul corespondent;
LEN (x\$)	lungimea șirului de caractere conținut în x\$;
STR\$ (x)	șirul de caractere corespunzător valorii numerice aflate în x;
VAL (x\$)	valoarea numerică a șirului de caractere format din cifre, aflat în variabila x\$;
SPACE\$ (x)	șirul de spații (blancuri) de lungime egală cu valoarea numerică din variabila x;
ASC (x\$)	valoarea numerică a primului caracter din șirul x\$.

Exemple:

$5^2=25$; $5*2=10$; $5/2=2.5$; $5+2=7$; $5-2=3$; $5\setminus 2=2$ (împărțire întreagă)

$5 \bmod 2=1$ (restul împărțirii lui 5 la 2)

$5=2$ - falsă; $5<>2$ - adevărată; $5<=2$ - falsă; $5<2$ - falsă;

$5>=2$ - adevărată; $5>2$ - adevărată;

$5=2$ OR $5<>2$ - adevărată; $5>=2$ AND $5<2$ - falsă;

$5=2$ XOR $5<>2$ - adevărată (o condiție falsă și cealaltă adevărată);

NOT $5=2$ - adevărată;

SQR(4)=2; ABS(2)=ABS(-2)=2;

INT(2.75)=2;

CHR\$(65)=A; CHR\$(97)=a; CHR\$(92)=\;

LEN("ABC")=3

STR\$(65)=A;

VAL("A")=65;

VAL(STR\$(65))=65; STR\$(VAL("A"))=A;

SPACE\$(4)=" " " "

Șirurile de caractere pot fi asociate cu variabile structurate. În cazul limbajului GW-BASIC precizăm că **orice șir de caractere are maximum 255 caractere**, chiar dacă sînt folosite de variabile structurate. Deci orice variabilă structurată de tip șir de caractere are componente de maximum 255 caractere.

Exemplul 17.2.**P17.2**

```
10 DIM AC$(2,2); REM tablou de siruri de caractere
20 LET AC$(1,1)="11"
30 LET AC$(1,2)="12"
40 LET AC$(2,1)="21"
50 LET AC$(2,2)="22"
60 PRINT AC$(1,1)+" "+AC$(2,2)
70 PRINT AC$(1,2)+" "+AC$(2,1)
80 DIM VEC$(2) : REM vector de siruri de caractere
90 LET VEC$(1)="VEC$ 1"
100 LET VEC$(2)="VEC$ 2"
110 PRINT VEC$(1)+VEC$(2)
```

17.3 Instrucțiuni

Instrucțiunea de scriere (afișare) este

```
PRINT e1, e2, ..., en
```

Expresiile e_i pot fi despărțite de , și atunci sînt scrise la începutul unei noi zone (o zonă are 14 poziții) sau de ; și atunci sînt scrise unele după altele.

Pentru a scrie la o anumită poziție pe ecran se utilizează instrucțiunea

```
LOCATE r, c
```

unde r desemnează valoarea unei linii (cu valori cuprinse între 1 și 25), iar c indică valoarea unei coloane (valorile sînt între 1 și 40 sau 1 și 80, în funcție de ecran).

Exemplu: Pentru a scrie șirul ABC în linia 5 începînd cu coloana 10 se utilizează

```
10 LOCATE 5, 10  
20 PRINT "ABC"
```

Instrucțiunea de citire este

```
INPUT v1, v2, ..., vn
```

avînd aceeași formă ca și cea din BASIC. Față de aceasta prezintă caracteristica citirii datelor de pe linia curentă. (La PC-uri ecranul nu conține linii rezervate pentru comenzi, mesaje și pentru citirea datelor).

Instrucțiunea de oprire a execuției unui program este

```
STOP .
```

Instrucțiunea de **atribuire** are forma

$$\text{LET } v = E$$

cu v și E ca în instrucțiunea corespunzătoare din BASIC, cu precizarea că în GW-BASIC LET este opțional.

De *exemplu*,

$$\text{LET } v = 2 + v$$

$$v = 4 + v$$

sînt, ambele, instrucțiuni de atribuire corecte în GW-BASIC.

Instrucțiunea **comentariu** este introdusă prin cuvîntul

$$\text{REM .}$$

Instrucțiunea GO TO are aceeași formă ca în BASIC.

Instrucțiunea IF are, fie forma cunoscută în BASIC

$$\text{IF exp_log THEN } S1:S2:\dots:Sn \quad (1)$$

cu semnificațiile cunoscute, fie forma

$$\text{IF exp_log THEN } S1:S2:\dots:Sn \text{ ELSE } T1:T2:\dots:Tm \quad (2)$$

Ultima formă a instrucțiunii se execută astfel: dacă exp_log are valoarea adevărat atunci se execută instrucțiunile $S1, S2, \dots, Sn$, iar în caz contrar se vor executa instrucțiunile $T1, T2, \dots, Tm$.

Exemplul 17.3.

P17.3 10 INPUT I,J
 20 IF I<J THEN PRINT I ELSE PRINT J

```
30 PRINT "STOP"  
40 STOP
```

Programul va tipări cea mai mică din valorile conținute în variabilele I, J.

Instrucțiunea FOR...NEXT are aceeași formă ca în BASIC.

Instrucțiunile READ, DATA, RESTORE, GO SUB și RETURN au în GW-BASIC aceleași forme și semnificații ca în BASIC.

17.4. Prelucrare sunet

Pentru a semnală sonor trecerea prin anumite puncte din program se utilizează

BEEP

ca în exemplul următor:

Exemplul 17.4.

```
P17.4 10 INPUT x  
20 IF x<0 THEN BEEP : GO TO 10  
30 PRINT x
```

Programul emite un "mesaj sonor" dacă valoarea citită este negativă.

Pentru a genera sunete echivalente unor note muzicale se folosește

SOUND f, d

unde f este frecvența, iar d indică durata în **unități**. De menționat că 18.2 unități echivalează cu o secundă.

Dăm mai jos frecvențele pentru notele specificate:

DO	269.630	523.250
RE	293.660	587.330
MI	329.630	659.260
FA	349.230	698.460
SOL	392.000	783.990
LA	440.000	880.000
SI	493.880	987.770

Prima coloană corespunde gamei do major.

Alegerea duratei este la latitudinea programatorului, dar trebuie să se țină cont de exprimarea parametrului d în secunde.

Realizarea pauzei între note se obține utilizând:

SOUND 32767, d

unde d indică durata pauzei între note.

17.5. Elemente de grafică

În cazul PC-urilor, ecranul este utilizat fie pentru a scrie și citi, fiind în starea **text**, fie pentru a desena, și atunci este în starea **grafică**. Există mai multe modele grafice (CGA, EGA, VGA etc.) care se deosebesc prin rezoluție, culoare. În mod implicit ecranul este în modul text.

Trecerea în modul grafic se poate face, de exemplu cu

SCREEN 2 .

În acest caz ecranul este împărțit în 640x200 puncte. Pentru **revenirea în modul text** se utilizează comanda

SCREEN 0 .

Ștergerea ecranului se face cu comanda CLS .

În modul grafic se pot efectua următoarele operații: desenări de puncte, trasări de drepte și de cercuri.

Pentru a "**pune**" un punct pe ecran folosim:

PSET (x, y) , C

unde x, y indică adresa pe ecran (x pe orizontală de la stînga la dreapta, iar y pe verticală de sus în jos), iar C culoarea. Valoarea C este opțională.

Trasarea unei linii se face cu:

LINE (x1, y1) - (x2, y2)

unde (x1, y1) și (x2, y2) reprezintă coordonatele extremităților liniei.

Trasarea unui cerc se face cu:

CIRCLE (x, y) , r

unde (x, y) reprezintă coordonatele centrului cercului, iar r indică raza acestuia.

Exemplul 17.5.

```
P17.5    10 REM 11 cercuri concentrice
        20 CLS : SCREEN 2
        30 FOR R=100 TO 0 STEP -10
        40   CIRCLE(160,100),R
        50 NEXT R
        60 REM Un. triunghi
        70 LINE(300,10) - (400,10)
        80 LINE (400,10) - (400,100)
        90 LINE (400,100) - (300,10)
```



ANEXA 1

Lista instrucțiunilor

- BEEP (cap.13,p.116;cap.17,p.180) = emisie sunet
- CIRCLE (cap.14,p.126;cap.17,p.182) (cerc) = trasare cerc
- CLS (cap.14,p.123;cap.17,p.182) (de la clear screen = curăță ecranul) = ștergere (inițializare) ecran
- DATA (cap.8,p.68;cap.17,p.180) (date) = definire date
- DIM (cap.11,p.96;cap.15,p.138;cap.17,p.175) (de la dimension = dimensiune) = definire variabile structurate
- DRAW (cap.14,p.123) (de la draw = a desena, a trasa) = trasare linie (BASIC)
- FOR...NEXT (cap.7,p.53;cap.17,p.180) (pentru ... următor) = instrucțiunea de ciclare
- GO SUB (cap.12,p.108;cap.17,p.180) (de la go subroutine = mergi în subprogram) = = instrucțiunea apel de procedură
- GO TO (cap.5,p.40;cap.17,p.179) (mergi la) = instrucțiunea de salt
- IF (cap.5,p.39;cap.17,p.179) (dacă) = instrucțiunea condițională
- INPUT (cap.3,p.20;cap.17,p.178) (intrare) = instrucțiunea de citire a datelor
- LET (cap.4,p.29;cap.17,p.179) (fie) = instrucțiunea de atribuire
- LINE (cap.17,p.182) (linie) = trasare linie (GW-BASIC)
- LOCATE (cap.17,p.178) (a localiza) = poziționarea cursorului pentru scriere pe ecran (GW-BASIC)
- PAUSE (cap.13,p.119) (pauză) = definește o pauză (BASIC)
- PLOT (cap.14,p.123) (pune) = "pune" un punct (BASIC)
- PRINT (cap.3,p.20;cap.17,p.178) (afișare) = instrucțiunea de scriere
- PSET (cap.17,p.182) (de la point set = pune punct) = "pune" un punct (GW-BASIC)
- READ (cap.8,p.68;cap.17,p.180) (citește) = instrucțiunea de inițializare
- REM (cap.4,p.31;cap.17,p.179) (de la remarks = comentarii) = instrucțiunea comentariu
- RESTORE (cap.8,p.71;cap.17,p.180) (reluare) = instrucțiunea de reluare a listei de date
- RETURN (cap.12,p.108;cap.17,p.180) (întoarce) = instrucțiunea de revenire din subrutină
- SCREEN (cap.17,p.182) (ecran) = definire ecran text/grafic (GW-BASIC)
- SOUND (cap.17,p.180) (sunet) = emisie sunet (GW-BASIC)
- STOP (cap.3,p.23;cap.17,p.178) (oprire) = instrucțiunea de oprire a programului

ANEXA 2

Lista comenzilor

BREAK (cap.2,p.17) (întrerupere) = întreruperea execuției unui program

EDIT (cap.2,p.15;cap.17,p.170) (editare) = editarea unei linii

LOAD (cap.2,p.17;cap.17,p.171) (încărcare) = încărcarea unui program

LIST (cap.2,p.17;cap.17,p.171) (listare) = listarea unui program

NEW (cap.2,p.16;cap.17) (nou) = distrugerea unui program

RUN (cap.2,p.15;cap.17,p.171) (alargă, aici cu sensul de execută) = executarea unui program

SAVE (cap.2,p.17;cap.17,p.171) (salvare) = salvarea unui program



ANEXA 3

Lista operațiilor

+, -, *, /, ^ (cap.4,p.28;cap.17,p.176) = operațiile de adunare, scădere, înmulțire, împărțire și ridicare la putere

\ (cap.17,p.176) = împărțire întreagă (GW-BASIC)

<, >, =, <=, >=, <> (cap.5,p.35;cap.17,p.176) = operații relaționale

NOT, AND, OR (cap.5,p.36;cap.17,p.176) (nu, și, sau) = operații logice

XOR (cap.5,p.38;cap.17,p.176) = operație logică "sau exclusiv" (GW-BASIC)

ABS (cap.7,p.59;cap.17,p.176) (de la absolute = valoare absolută) = valoare absolută

ASC (cap.17,p.176) (de la numele tabelii de conversie ASCII) = valoarea numerică a primului caracter dintr-un șir (GW-BASIC)

CHR\$ (cap.15,p.152;cap.17,p.176) (de la character = caracter) = determină caracterul corespunzător unei valori numerice

CODE (cap.15,p.152) (cod) = determină valoarea numerică a unui șir (BASIC)

INT (cap.7,p.60;cap.17,p.176) (de la integer = întreg) = determină partea întregă

LEN (cap.15,p.151; cap.17,p.176) (de la length = lungimă) = determină lungimea unui șir

MOD (cap.17,p.176) (de la modulus = modul) = determină restul împărțirii întregi (GW-BASIC)

RND (cap.3,p.23) (de la random = aleator) = furnizează un număr aleator între 0 și 1

SPACE\$ (cap.17,p.176) (spațiu) = determină un număr întreg de spații (GW-BASIC)

SQR (cap.7,p.61;cap.17,p.176) (de la square root = rădăcina pătrată) = rădăcina pătrată

STR\$ (cap.15,p.150;cap.17,p.176) (de la string = șir) = șirul de caractere corespunzător unui număr

VAL (cap.15,p.150;cap.17,p.176) (de la value = valoare) = valoarea numerică a unui șir de caractere

ANEXA 4

Lista exemplelor de programe

P1.1	(pag. 7)	-	Afişare text.
P1.2	(pag. 7)	-	Citire/scriere.
P1.3	(pag. 8)	-	Calculul sumei și produsului a două numere
P3.1	(pag. 21)	-	Transformare metri în centimetri.
P3.2	(pag. 21)	-	Idem; cu comentarii.
P3.3	(pag. 22)	-	Transformare grade Fahrenheit în grade Celsius.
P3.4	(pag. 22)	-	Calculul unor expresii (suma, diferența, produsul, cîmul a două fracții).
P3.5	(pag. 23)	-	Utilizarea instrucțiunii STOP.
P3.6	(pag. 23)	-	Scrierea unor valori pe aceeași linie.
P3.7	(pag. 24)	-	Scrierea instrucțiunilor pe aceeași linie.
P4.1	(pag. 31)	-	Calculul valorii unei expresii. *
P4.2	(pag. 31)	-	Linie comentariu.
P4.3	(pag. 31)	-	Calculul valorii unei expresii, cu comentarii.
P4.4	(pag. 32)	-	Schimbarea locurilor inițiale ale unor elemente.
P5.1	(pag. 39)	-	Suma a două numere citite la execuție.
P5.2	(pag. 40)	-	Utilizarea instrucțiunii GO TO.
P5.3	(pag. 41)	-	Suma a două valori pozitive cu test de semn.
P5.4	(pag. 41)	-	Suma unei succesiuni de numere; varianta 1.
P5.5	(pag. 41)	-	Idem; varianta 2.
P5.6	(pag. 41)	-	Idem; varianta 3.
P5.7	(pag. 41)	-	Idem; varianta 4.
P6.1	(pag. 50)	-	Produsul mai multor valori.
P6.2	(pag. 51)	-	Prelucrarea numerelor în funcție de semn.
P7.1	(pag. 54)	-	Utilizare FOR...NEXT.
P7.2	(pag. 54)	-	Idem pentru împărțirea a două expresii.
P7.3	(pag. 56)	-	Utilizare FOR...NEXT cu STEP pozitiv.

ANEXA 4 Lista exemplurilor de programe

P7.4	(pag. 57)	-	Utilizare FOR...NEXT cu STEP negativ.
P7.5	(pag. 58)	-	Utilizare FOR...NEXT interior.
P7.6	(pag. 58)	-	Exemplu de utilizare greșită FOR...NEXT.
P7.7	(pag. 58)	-	Decrementare (micșorarea valorii pasului).
P7.8	(pag. 59)	-	Program care ciclează.
P7.9	(pag. 59)	-	Valoare absolută.
P7.10	(pag. 60)	-	Calculul părții întregi.
P7.11	(pag. 63)	-	Divizibilitate; varianta 1.
P7.12	(pag. 64)	-	Divizibilitate; varianta 2.
P7.13	(pag. 64)	-	Determinarea a trei numere care satisfac o relație.
P7.14	(pag. 65)	-	Cîțul și restul unei împărțiri.
P8.1	(pag. 68)	-	Utilizare READ și DATA.
P8.2	(pag. 69)	-	Idem.
P8.3	(pag. 69)	-	Stabilirea lunii și a zilei.
P8.4	(pag. 70)	-	Determinarea zilei din săptămîină în care cade o anumită dată.
P8.5	(pag. 72)	-	Utilizare RESTORE.
P8.6	(pag. 72)	-	Determinarea elementului maxim dintr-un șir.
P9.1	(pag. 75)	-	Utilizare șiruri de caractere.
P9.2	(pag. 79)	-	Idem.
P9.3	(pag. 79)	-	Idem.
P9.4	(pag. 80)	-	Concatenare șiruri.
P9.5	(pag. 80)	-	Afișarea lunilor cu numărul de zile.
P9.6	(pag. 80)	-	Idem; modificat.
P9.7	(pag. 81)	-	Determinarea numărului lunii.
P9.8	(pag. 81)	-	Schimbarea locurilor inițiale (P4.4) utilizînd șiruri de caractere.
P9.9	(pag. 82)	-	Determinarea șirului maxim de caractere dintr-o mulțime de șiruri.
P9.10	(pag. 82)	-	Utilizarea aceluiași nume pentru variabile de tipuri diferite.
P10.1	(pag. 85)	-	Rezolvarea ecuației de gradul I.

P10.2	(pag. 86)	-	Determinarea numerelor direct proporționale cu altele date.
P10.3	(pag. 87)	-	Reprezentarea la scară.
P10.4	(pag. 88)	-	Calculul unei mărimi în funcție de procent.
P10.5	(pag. 89)	-	Algoritmul lui Euclid.
P10.6	(pag. 89)	-	Joc cu întrebări și răspunsuri.
P11.1	(pag. 97)	-	Calculul sumei a doi vectori cu lungime dată.
P11.2	(pag. 97)	-	Idem pentru vectori de lungime variabilă.
P11.3	(pag. 98)	-	Schimbarea locurilor inițiale (P4.4) cu număr variabil de scaune.
P11.4	(pag. 99)	-	Construirea unui șir în funcție de elementele altui șir.
P11.5	(pag.100)	-	Operații elementare cu mulțimi.
P11.6	(pag.102)	-	Idem; varianta 2.
P12.1	(pag.108)	-	Operații cu numere în funcție de semn.
P12.2	(pag.109)	-	Idem; subrutină.
P12.3	(pag.110)	-	Identificarea poziției unor valori într-un șir.
P12.4	(pag.111)	-	Identificarea apartenenței unui element la un șir.
P12.5	(pag.112)	-	Determinarea valorii maxime dintr-un șir.
P12.6	(pag.112)	-	Program cu mai multe subrutine.
P13.1	(pag.118)	-	Utilizare BEEP.
P13.2	(pag.118)	-	Cîntecel.
P13.3	(pag.119)	-	Semnalarea sonoră a nîndeplinirii unei condiții.
P13.4	(pag.120)	-	Cîntecul „Banul Măracine“.
P14.1	(pag.123)	-	Trasarea unei linii.
P14.2	(pag.124)	-	Trasare linii între puncte date.
P14.3	(pag.125)	-	Trasarea unui triunghi.
P14.4	(pag.125)	-	Trasarea unui triunghi cu linii în interior.
P14.5	(pag.126)	-	Trasarea unui pătrat cu cerc în interior.
P14.6	(pag.131)	-	Trasarea unei spirale din linii drepte.
P14.7	(pag.133)	-	Trasarea unor linii colorate.
P14.8	(pag.134)	-	Colorarea interiorului unui dreptunghi folosind puncte.
P14.9	(pag.134)	-	Colorarea interiorului unui dreptunghi folosind linii.

ANEXA 4 Lista exemplelor de programe

P15.1	(pag.139)	-	Citirea și scrierea unui tablou de dimensiune fixă.
P15.2	(pag.139)	-	Idem pentru dimensiune variabilă.
P15.3	(pag.140)	-	Transferul unui tablou în alt tablou.
P15.4	(pag.142)	-	Trasarea unci figuri geometrice.
P15.5	(pag.145)	-	Afișarea unui catalog cu elevi și note.
P15.6	(pag.146)	-	Calculul mediei notelor.
P15.7	(pag.150)	-	Utilizare variabile numerice și de tip șir de caractere.
P15.8	(pag.151)	-	Calculul lungimii unui șir.
P15.9	(pag.153)	-	Afișarea unui șir de litere.
P16.1	(pag.159)	-	Subrutină pentru cel mai mare divizor comun.
P16.2	(pag.159)	-	Program de apel al subrutinei P16.1.
P16.3	(pag.160)	-	Operații cu numere raționale.
P16.4	(pag.163)	-	Calculul valorii unui polinom.
P16.5	(pag.165)	-	Transformarea din baza 10 în baza 2.
P17.1	(pag.172)	-	Introducerea automată a unui program în GW-BASIC.
P17.2	(pag.177)	-	Tablou de șiruri de caractere.
P17.3	(pag.179)	-	Tipărirea valorii minime.
P17.4	(pag.180)	-	Prelucrare sunet.
P17.5	(pag.183)	-	Cercuri concentrice.

BIBLIOGRAFIE

1. I. Diamandi: *Partenerul meu de joc calculatorul*, Recoop, București, 1989.
2. I. Diamandi: *Cum să realizăm jocuri pe calculator*, Editura Agni, București 1992.
3. I. Diamandi, Gh. Păun: *40 de jocuri logice în BASIC*, Editura Agni, București 1992.
4. H. Georgescu, T. Bălănescu, V. Ștefănescu: *Matematică, manual de algebră pentru clasa a IX-a*, cap.VII, Editura Didactică și Pedagogică București, 1989.
5. H. Georgescu, T. Bălănescu, V. Ștefănescu: *Matematică, manual de algebră pentru clasa a X-a*, cap.V, Editura Didactică și Pedagogică București, 1989.
6. D.M. Monro: *Interactive computing with Basic. A first course*, Edward Arnold, 1974 (republicată în 1977, 1978, 1980, 1981).
7. L. State: *Limbajul BASIC pentru microcalculatoare*, Editura Agni, București 1992.
8. S. Vickers: *ZX-SPECTRUM BASIC programming*, Sinclair Research, 1982.
9. *** Colecția *Gazetei de informatică* pe anii 1991, 1992.
10. *** *GW-BASIC Interpreter, User's Reference*.
11. *** *GW-BASIC Interpreter, User's Guide*.



Cine ești tu, BASIC?

este o introducere
pentru începători în limbajul de
programare BASIC. Poate fi folosit atât
pentru calculatoarele compatibile
Sinclair Spectrum, cât și pentru cele
compatibile IBM-PC (GW-BASIC).
Este un material didactic ideal pentru
predarea informaticii la clasele V-VIII

Domnul Marian GHEORGHE este
lector la Facultatea de Matematică,
Universitatea București. Predă cursuri
de limbaje de programare: BASIC,
Pascal și C. Se ocupă de pregătirea
lotului pentru olimpiadele
internaționale de informatică.

