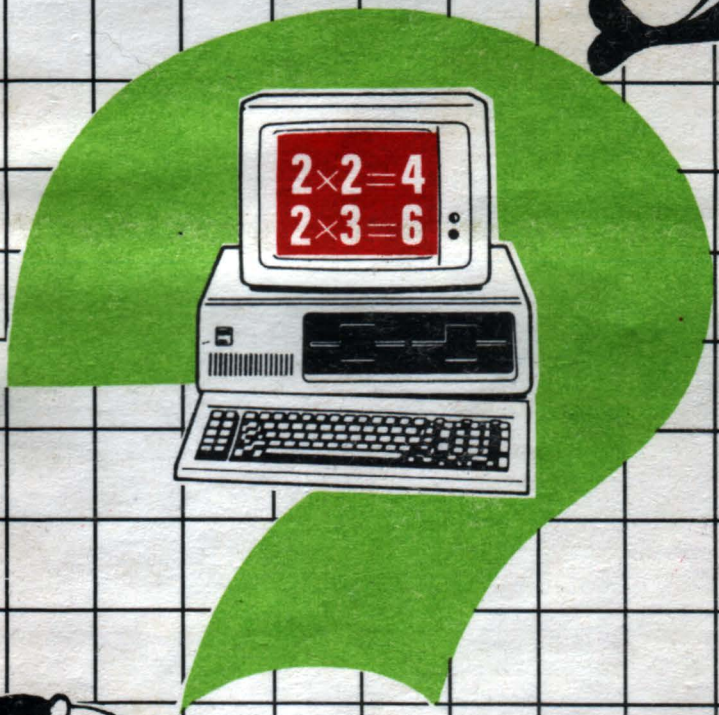
MARIAN GHEORGHE

Cine ești tu BASIC?



Editura Agni

Biblioteca de Informatică

Marian Gheorghe

Cine eşti tu, BASIC ?

Editura Agni

Bucureşti

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, BUCUREŞTI

CUPRINS

Au apărut:

Ion Diamandi

♦Cum să realizăm jocuri pe calculator

Luminiţa 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 ştie LOGO ?

Răzvan Andonie, Ilie Gârbacea

♦Algoritmi fundamentali în C++

Puteţi să primiţi aceste cărţi scriindu-ne pe adresa:

EDITURA AGNI C.P. 30-107 Bucureşti

Model de cerere:

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

Vă rog să-mi expediaţi prin colet poştal cu ramburs:

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

INTRODUCERE

Cartea cuprinde noţiunile cele mai des întîlnite 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.) şi un scurt rezumat al caracteristicilor variantei GW-BASIC (disponibilă pe calculatoare compatibile IBM-PC).

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

Dacă în anii 70 limbajul era destinat iniţierii în informatică a studenţilor din primii ani ai facultăţilor tehnice, economice şi de ştiinţă, în anii 80, odată cu apariţia calculatoarelor personale şi a creşterii 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 iniţiere în informatică pentru elevii de gimnaziu. Cartea se adresează elevilor claselor a VI-a (primele 10 capitole) şi celor din clasa a VII-a, pentru trimestru I (capitolele 11 la 16).

Cartea nu urmăreşte prezentarea sistematică a noţiunilor limbajului BASIC ci o introducere graduală (cu exemple şi exerciţii rezolvate, dar şi propuse) a materialului. Astfel, manualul conţine atît noţiuni specifice limbajului BASIC (instrucţiunile de intrare/ieşire, de atribuire, IF, GO TO, FOR NEXT, READ, DATA, RESTORE, la clasa a VI-a şi subrutine, variabile structurate, elemente de grafică la clasa a VII-a), cît şi unele concepte ale limbajului ce modelează definiţii din matematică sau fizică (valoarea absolută, rădăcina pătrată, partea întreagă) şi probleme, în general de matematică, care sînt rezolvate cu calculatorul (probleme de divizibilitate, exprimări procentuale, operaţii cu mulţimi, calcularea valorii unor polinoame, construcţia unor figuri geometrice).

Materialul de faţă presupune citirea cărţilor destinate elevilor de clasa a V-a şi înţelegerea unor noţiuni şi fapte descrise acolo: **memoria calculatorului** şi **locaţie de memorie, tastatura,** cu evidenţierea unor **taste funcţionale** cu rol special, punerea în funcţiune a calculatorului (legarea la reţeaua electrică, conectarea la ecranul pe care afişăm, eventual racordarea la un casetofon).

Avem în vedere în expunerea de faţă că se lucrează pe calculatoare din familia Sinclair SPECTRUM într-o configuraţie minimală (calculator + monitor + casetofon). Cei care beneficiază de dischetă vor face apel la un manual care prezintă cele cîteva elemente necesare lucrului cu aceasta. Posesorii de calculatoare compatibile IBM-PC după ce parcurg capitolele de la 3 la 16 vor citi şi capitolul 17 care prezintă principalele diferenţe dintre limbajul BASIC disponibil pe calculatoare din familia Sinclair SPECTRUM şi 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 noţiunea 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 instrucţiune, `PRINT`, numită **instrucţiune de scriere.** Efectul acestui program este acela de a afişa pe ecran următorul text:

```
Primul program in limbajul BASIC
```

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

Următorul program "citeşte" două valori numerice (aşteaptă să fie introduse de la tastatură) pe care le depune în variabilele `A` şi `B`. Aceste valori sînt apoi afişate pe ecran.

P1.2

```
10 INPUT A, B
20 PRINT A, B
```

Instrucţiunea `INPUT` care citeşte valorile numerice în `A` şi `B` se numeşte **intrucţiune de citire.** Valorile sînt afişate de instrucţiunea `PRINT`.

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

a) Se citesc cele două numere în două variabile A şi B;
b) Se calculează A+B şi A· B, iar rezultatele sînt depuse în S şi P;
c) Se afişează valorile conţinute în variabilele S şi P.

Textul programului care realizează ce am menţionat 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 ***observaţii***:

- liniile care compun programul sînt numerotate crescător cu 10, 20, 30, 40
- instrucţiunea de citire, INPUT, care apare pe linia 10 citeşte în A şi B două valori numerice
- în liniile 20 şi 30 se calculează suma şi produsul valorilor aflate în A şi B care se depun în S, respectiv P; observăm că operaţia de înmulţire are simbolul *; instrucţiunea LET de pe aceste linii se numeşte **instrucţiunea de atribuire**
- linia 40 conţine instrucţiunea de afişare a conţinutului variabilelor A şi B.

Pentru a putea scrie şi executa aceste programe trebuie să mai aveţi puţină răbdare pînă la parcurgerea paragrafului următor.

1.2. Introducerea unui program. Taste funcţionale

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

Pornirea calculatorului constă din următoarele operaţii:

- conectarea acestuia la un monitor sau televizor (care în prealabil este pornit şi reglat);

- legarea la reţeaua 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 informaţiei 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 instrucţiunii `INPUT`.

Înaintea instrucţiunii `INPUT` se poate introduce un număr, de exemplu `10`. Tastînd în ordine `1, 0, INPUT`, atunci din starea iniţială prezentată în figura 1.2 se ajunge la situaţia prezentată în figura 1.3:

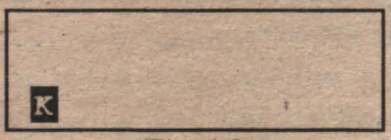


Fig. 1.2

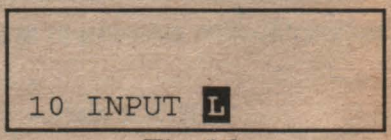


Fig. 1.3

Privind cu atenţie figura 1.3, observăm că apare şi o literă, **L**, care semnifică o altă stare decît 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 obţine A atunci trebuie să se ţină apăsată tasta **CS** (sau **CAPS SHIFT**) şi concomitent să se apese pe **A**. Pentru a tasta un simbol special, de exemplu +, se apasă pe **SS** (sau **SYMBOL SHIFT**) şi în acelaşi timp pe tasta pe care scrie +.

Dacă, de exemplu, sîntem în situaţia prezentată în figura 1.3 şi tastăm:

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

şi se obţine imaginea din figura 1.4:

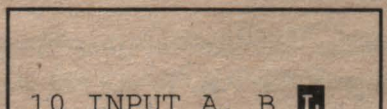


Fig. 1.4

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** şi **L** menţionate anterior, calculatorul poate să se mai afle în stările **E** sau **G**.

Trecerea din **K** sau **L** în **E** şi invers se obţine tastînd simultan **SS** şi **CS**. Trecerea în starea **G** şi revenirea apoi în starea iniţială se obţine tastînd simultan **CS** şi **9**.

În starea **K** puteţi introduce (urmărind tastatura de la stînga la dreapta şi 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 obţin literele mari: `Q, W, E,` etc.

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

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

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

În starea **E** şi ţinînd **SS** apăsată se obţin informaţiile trecute sub taste (de exemplu: `BEEP, INK, PAPER` etc.).

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

Pentru a executa programul din figura 1.1 se apasă, în starea **K**, tasta ce desemnează comanda **RUN** şi apoi **CR**. Pe ecran apare **L**, semn că trebuie tastat un simbol; în acest caz se aşteaptă 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 şi apoi **CR**. Pe ecran sînt afişate două valori numerice, 14 şi 40, reprezentînd 10+4 şi respectiv 10*4. Revenirea la program se obţine tastînd **CR**.

În figura 1.5 este prezentată situaţia după introducerea numerelor 10 şi 4, iar în figura 1.6 apar rezultatele.

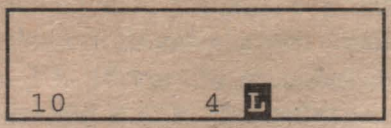


Fig. 1.5

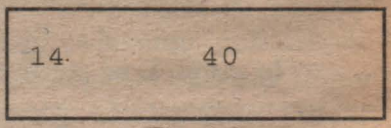


Fig. 1.6

☞ Reţineţi !

■ Liniile unui program sînt numerotate crescător cu numere cuprinse între 1 şi 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.).

■ Operaţia de înmulţire se reprezintă prin simbolul *, iar operaţia de împărţire prin simbolul /.

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

■ Obţinerea literelor mari se realizează ţinînd apăsată tasta **CS**, iar obţinerea simbolurilor speciale (;, :, +, -, * etc.) se realizează acţionînd tasta **SS** şi cea pe care se află simbolul ales.

1. Tastaţi următorul program:

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

Există diferenţe între programul pe care l-aţi tastat şi cel care a apărut pe ecranul televizorului ?

2. Executaţi programul de la exerciţiul 1, introducînd datele 20 şi 10.

3. Scrieţi un program similar cu cel din exerciţiul 1, dar schimbaţi - cu * şi + cu /.

2

Comenzi de bază. Ecranul de afişare

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

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

Comanda de listare - LIST

Pentru a obţine pe ecran textul unui program deja scris se acţionează tasta **LIST** şi apoi **CR**. Pentru a lista un program de la un anumit număr de linie, după comanda **LIST** se introduce numărul de linie al primei instrucţiuni ce se doreşte a fi afişată. Dacă textul care se afişează 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 afişează 22 de linii apare mesajul

```
scroll?
```

la care se poate răspunde cu **Y** pentru a continua afişarea 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
```

acţionarea comenzii **LIST 10** are ca efect afişarea textului începînd cu linia 10. Cînd se ajunge la linia 32 listarea se opreşte şi este afişat mesajul

```
scroll?
```

Dacă se tastează **Y** atunci sînt afişate şi următoarele linii din program.

Comanda de rulare - RUN

Executarea unui program se obţine în urma acţionării tastei **RUN** şi apoi a tastei **CR**. Încercaţi această comandă pentru programul de mai sus.

Comanda de modificare a unei linii - EDIT

Urmărind textul unui program observaţi că una din linii este precedată de semnul >. Această linie va fi numită **linia curentă.** Comanda de modificare a unei linii, care se obţine tastînd **EDIT** (**CS** şi **1** tastate simultan) se referă la linia curentă. Dacă dorim să modificăm altă linie decît cea curentă, se acţionează săgeţile ↓ (**CS** şi **6** tastate simultan) şi ↑ (**CS** şi **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 ↑ şi apoi **EDIT**. Este coborîtă pentru a fi modificată linia

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

Modificarea unei linii se obţine acţionînd săgeţile → (**CS** şi **8** tastate simultan), ← (**CS** şi **5** tastate simultan) şi **DELETE** (**CS** şi **0** tastate simultan). Săgeţile stînga şi dreapta ne mută la fiecare apăsare cîte un caracter la stînga, respectiv la dreapta. Tastînd **DELETE** este şters caracterul aflat la stînga poziţiei curente. Pentru a introduce caractere noi într-o anumită poziţie, după ce se ajunge în poziţia 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` şi a lui `4` cu `31`, procedăm astfel:

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

Linia `3` a programului anterior devine:

```
3 LET BC=31
```

Comanda de ştergere 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** şi **SPACE** tastate simultan) are scopul de a întrerupe executarea unui program.

Comenzi pentru salvarea şi încărcarea programelor - SAVE şi LOAD

Programele pe care dorim să le păstrăm trebuie salvate, cel mai adesea şi mai comod, pe casete. În prealabil casetofonul trebuie să fie conectat la calculator şi 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"
```

şi pornim casetofonul pentru înregistrare.

Operaţia inversă, de încărcare a unui program aflat pe casetă, se face tastînd:

```
LOAD "nume"
```

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

```
LOAD ""
```

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

Ecranul de afişare

Ecranele pe care sînt afişate programele, rezultatele acestora, precum şi o serie de comenzi, au o structură precizată, deşi acestea pot avea mărimi diferite. Astfel, indiferent de mărimea sa ecranul este împărţit 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).

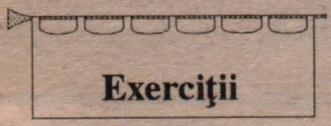
În partea inferioară a ecranului mai sînt două linii (22 şi 23) rezervate pentru a se afişa comenzi, date care se tastează la instrucţiunea `INPUT`, mesaje diverse.

Reţineţi că aceste linii (22 şi 23) ne sînt inaccesibile, în sensul că nu putem afişa rezultate şi nici nu putem lista programe pe aceste linii.

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

☞ Reţineţi !

- **LIST** - listarea programului.
- **RUN** - executarea programului.
- **EDIT** - modificarea liniei curente.
- **NEW** - ştergerea programului din memorie.
- **BREAK** - întreruperea execuţiei programului.
- **SAVE** - salvarea programului.
- **LOAD** - încărcarea programului.
- Ecranul este împărţit în 22 de linii, numerotate de la 0 la 21 şi 32 de coloane, numerotate de la 0 la 31.
- Comenzile şi instrucţiunile sînt introduse în aceeaşi stare, **K**.



Exerciţii

1. Scrieţi un program cu 5 linii.

2. Listaţi programul scris la exerciţiul 1.

3. Executaţi programul introdus la exerciţiul 1.

4. Modificaţi programul înlocuind variabila care apare prima într-o instrucţiune de la exerciţiul 1 (dacă există), cu alt nume.

5. Rulaţi programul modificat.

6. Salvaţi programul iniţial pe casetă.

7. Ştergeţi programul din memorie şi încărcaţi programul de pe casetă (cel care a fost salvat la exerciţiul 6).

8. Fie programul:

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

Faceti următoarele operaţii cu acest program: a) executaţi-l; b) modificaţi în linia `20`, `+` cu `-`; c) salvaţi programul modificat sub numele `PROGNOU`; d) executaţi programul modificat.

3

Instrucţiunile de citire şi scriere. Instrucţiunea STOP

Am văzut în capitolul 1 că instrucţiunile de citire şi scriere sînt INPUT şi respectiv PRINT.

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

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

unde v1, v2,..., vn sînt zone de memorie ale calculatorului, în care se copiază diferite valori. Numim aceste zone de memorie, **variabile**. Rezultă astfel variabilele v1, v2,..., vn. În urma executării acestei instrucţiuni valorile tastate sînt depuse în variabilele din instrucţiunea 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 aşteptare (pînă la tastarea valorilor numerice ce urmează a fi depuse în variabilele A, B şi C). În acest moment se tastează succesiv:

10 **CR** 20 **CR** 30 **CR**

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

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

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

unde e1, e2,..., en sînt variabile, constante sau expresii.

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

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

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

Instrucţiunea INPUT citeşte datele care apar pe ecran pe liniile 22 şi 23 (în partea de jos a ecranului). Instrucţiunea PRINT afişează datele pe ecran pe una din liniile de la 0 la 21. Afişarea începe de la linia 0.

Exemplul 3.1. Acest program citeşte valorile unor măsurători în metri şi calculează aceste valori în centimetri:

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

Instrucţiunea INPUT (ca şi instrucţiunea PRINT) prevede posibilitatea ca valoarea care se citeşte (care se scrie) să fie precedată de un text care să clarifice modul de introducere al datelor (sau semnificaţia rezultatelor afişate). Textul care urmează a fi trimis pe ecran prin aceste instrucţiuni este cuprins între simbolurile " şi poartă numele de **şir 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 însoţesc operaţiile de citire şi de scriere clarifică execuţia programului.

Observaţii:

♦ Folosind , ca delimitator de variabile, datele tastate sau afişate 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 modificaţi exemplul 3.2, substituind , cu ; şi apoi să executaţi programul.

♦ Folosind ; după ultima expresie dintr-o instrucţiune de scriere, nu se mai trece pe linia următoare la viitoare operaţie 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; a:b-c:d; a:b•c:d; (a:b):(c:d),

conform cu:

(a•d+b•c):b•d; (a•d-b•c):b•d; a•c:b•d; a•d:b•c,

presupunînd că sînt îndeplinite condiţiile ca aceste fracţii 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 afişa numere obţinute la întîmplare (numite aleatoare), prin utilizarea operaţiei **RND**:

```
10 PRINT RND
```

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

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

P3.5

```
10 INPUT A
20 PRINT A
30 STOP
40 INPUT A
```

se execută astfel: se citeşte în variabila A o valoare numerică, apoi se scrie valoarea citită, după care programul se opreşte (instrucţiunea STOP). Instrucţiunea de pe linia 40 nu se mai execută.

Exemplul 3.5. Se citesc trei valori numerice şi se scriu pe aceeaşi 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 aceeaşi linie se obţine folosind ; după ultima expresie din instrucţiunea PRINT de pe liniile 20 şi 40, în cazul de faţă " ".

Limbajul BASIC prevede posibilitatea de a scrie mai multe instrucţiuni pe aceeaşi linie, despărţite 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
```

☞ Reţineţi !

■ Exemple de instrucţiuni INPUT:

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

■ Exemple de instrucţiuni PRINT:

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

■ Instrucţiunea STOP opreşte execuţia programului.

■ Instrucţiunile de pe aceeaşi linie se despart cu :.

■ Generarea numerelor întîmplătoare (aleatoare) cuprinse între 0 şi 1 se face utilizînd RND.

Exerciţii

1. Să se citească în două variabile două mărimi exprimate în kilometri şi să se afişeze valorile în metri ale sumei şi diferenţei celor două mărimi.

2. Să se citească în `A` şi `B` şi în `C` şi `D` cîte două valori exprimate în metri şi să se calculeze `A/B+C/D` în centimetri.

3. Să se citească două valori numerice şi să se afişeze produsul acestora.

4. Să se citească patru valori numerice în variabilele `A, B, C` şi `D` şi să se afişeze `A+B, A+B+C, A+B-C*D`.

5. Să se citească patru valori numerice în variabilele `A, B, C` şi `D` şi să se afişeze `A/C+B/C` şi `A/D-B/D`, presupunînd că `C` şi `D` au valori diferite de zero.

6. Să se citească două valori numerice în `A` şi `B` şi să se afişeze `A+B, A-B, A*B` şi `A+(B-A)`.

7. Să se citească două valori numerice în `A` şi `B` şi să se afişeze `A+B`, apoi să se citească alte două valori numerice tot în `A` şi `B` şi să se afişeze pe aceeaşi linie `A-B`, despărţind rezultatele cu simbolul "/".

8. Să se scrie un program care conţine 3 instrucţiuni `PRINT` ce afişează toate rezultatele pe aceeaşi linie.

9. Să se scrie un program care să scrie trei numere generate întîmplător.

10. Să se scrie un program care citeşte o valoare numerică în variabila A şi afişează suma dintre valoarea conţinută în A şi un număr generat întîmplător.

11. Să se scrie un program care afişează suma a trei numere generate întîmplător.

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 afişează şi să se explice.

14. Modificaţi programul de la exerciţiul 13 astfel încît să afişeze şi valoarea lui A.

4

Constante, variabile numerice, expresii aritmetice. Instrucţiunea 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 menţionat că numerele reale se scriu cu "." în loc de"," şi că alături de notaţiile de mai sus, constantele numerice pot fi specificate şi prin menţionarea unei puteri a lui 10 ce înmulţeşte numărul. De exemplu, notaţiile

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

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

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 iniţial, 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 şi 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 conţin alte simboluri decît cele menţionate (+, ?, _), fie nu încep cu o literă.

Observaţie:

♦ Numele de identificatori pot conţine atît litere mari cît şi litere mici. Nu se face distincţie între acestea. De exemplu Ana, ANa, ANA, aNa, reprezintă aceeaşi variabilă.

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

- Ordinea după care se efectuează operaţiile este cea cunoscută din matematică: întîi se efectuează ridicările la putere, apoi înmulţirile şi împărţirile, iar la sfîrşit adunările şi scăderile. În exemplul de mai sus, ordinea efectuării operaţiilor este:

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

Exemple de expresii aritmetice:

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

Exerciţiu

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 operaţiilor este identică cu cea cunoscută din matematică. Astfel, ştim că pentru a indica anumite calcule ce trebuie să se efectueze înaintea altora, se folosesc parantezele rotunde, drepte, acolade. Acest fapt este valabil şi în limbajul BASIC cu remarca 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↑2)-3/4).
```

Cu ajutorul noţiunilor de **variabilă** şi **expresie aritmetică** se construieşte o instrucţiune de bază în limbajul BASIC, **instructiunea de atribuire**, avînd forma:

```
· LET V=E
```

unde `LET` desemnează instrucţiunea de atribuire, `V` este numele unei variabile, iar `E` este o expresie. Efectul acestei instrucţiuni este următorul: se calculează expresia `E` şi rezultatul este depus în variabila `V`.

De *exemplu*:

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

desemnează instrucţiuni de atribuire.

Observaţii:

♦ Instrucţiunea de atribuire exprimă puterea de calcul (gamă de operaţii, precizie) a unui limbaj. Din acest punct de vedere se observă că limbajul BASIC conţine operaţiile de bază (adunare, scădere, înmulţire, împărţire, ridicare la putere; ulterior vom introduce şi radicalul).

♦ Dacă expresiile care apar în instrucţiunea de atribuire, se compun cu ajutorul unor variabile, atunci acestea trebuie să aibă o valoare în momentul execuţiei.

Exemplu: In momentul executării instrucţiunii

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

trebuiau să se fi executat alte instrucţiuni care să dea valori variabilelor a şi b. De exemplu instrucţiunile:

```
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 şi 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
```

Instrucţiunea de atribuire din linia 20 conţine expresia `5*X*Y+X↑2+Y↑2-2*X+1.5` care se calculează pentru valorile lui `X` şi `Y` citite de instrucţiunea din linia 10. Rezultatul este depus în variabila `E`, care este apoi scrisă cu instrucţiunea `PRINT` de la linia 30.

Instrucţiunea comentariu, REM, are doar efectul unor clarificări în program şi nu produce nici un efect la execuţie.

Exemplul 4.2.

Programul de mai jos nu are nici un efect la execuţie, 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 uşor generalizat înlocuind instrucţiunile de atribuire din linia 40 cu o instrucţiune de citire. Vă propunem această variantă ca exerciţiu.

Exemplul 4.4.

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

Ana	Maria	Lucia
S1	S2	S3

Cei trei copii se mută ajungînd în următoarea situaţie:

Maria	Lucia	Ana
S1	S2	S3

Care sînt mutările necesare pentru a obţine situaţia de mai sus ?

O variantă ar fi următoarea:

1) Ana eliberează scaunul S1 trecînd 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 şi 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
```

☞ Reţineţi !

■ Operaţiile de adunare, scădere, înmulţire, împărţire, ridicare la putere se reprezintă prin `+`, `-`, `*`, `/` şi respectiv `↑`.

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

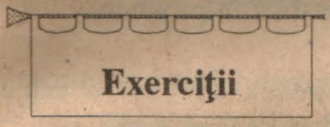
■ Ordinea efectuării operaţiilor este cea din matematică: ridicarea la putere, înmulţirea şi împărţirea, apoi adunarea şi scăderea.

■ Instrucţiunea de atribuire are forma :

```
LET V=E .
```

■ Instrucţiunea comentariu are forma :

```
REM Secvenţă de caractere .
```



Exerciţii

1. Scrieţi un program care să calculeze $1+2\cdot3-4\cdot5+6\cdot7-8\cdot9$ şi să afişeze 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 exerciţiul 2 să se conceapă un program care citind pe x să calculeze pe E(x).

4. Introduceţi un comentariu în programul ce rezolvă exerciţiul 3 pentru a explica scopul programului.

5. Se dau expresiile

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

şi

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

Să se scrie expresiile în limbajul BASIC.

6. Se dau expresiile

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

a) Să se scrie aceste expresii în limbajul BASIC.
b) Ce valori iau E(X) şi F(X) pentru X=1?
c) Ce se poate spune despre aceste două expresii atunci cînd X ia diferite valori?

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

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

9. Pentru expresiile date în exerciţiul 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 şi Y citite în program.

10. In două sertare A şi B se află 3 caiete şi respectiv 3 cărţi. Să se scrie un program care să schimbe între ele conţinuturile celor două sertare folosind o etajeră.

11. S ă se scrie un program care să afişeze valorile variabilelor `In, NOI, IOn şi vAR` care conţin valorile `12E+1, 1.1E-1, -3.14` şi respectiv `128.123E-2`.

5

Expresii logice. Instrucţiunile IF şi GO TO

5.1. Expresii condiţionale (condiţii)

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

În limbajul BASIC pot fi exprimate pe lîngă calcule numerice şi anumite **condiţii**, de exemplu `x<y`, care înseamnă `x` mai mic decît `y`, sau `z∈[1,2]` care înseamnă că `z` aparţine intervalului [1,2], adică `z` este mai mare sau egal cu `1` şi mai mic sau egal cu `2`.

Valoare unei astfel de condiţii poate fi **adevărată**, atunci cînd valorile variabilelor satisfac condiţia (de exemplu pentru `x=3, y=4, z=1.5` în cazul de mai sus) sau **falsă**, atunci cînd nu sînt satisfăcute condiţiile (inversaţi valorile lui `x` şi `y` specificate mai sus şi condiţia devine `4<3`, adică falsă).

Exprimarea condiţiilor se face cu ajutorul următoarelor operaţii, numite **relaţionale**:

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

Exemple:

Condiţii adevărate:

```
1<2, 1<=2, 1<>2, 1=1, 1<=1, 1>=1.
```

Condiţii false:

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

Condiţia `x<=2` este adevărată pentru toate valorile lui `x` mai mici sau egale cu `2` (de exemplu `2, 1, -1` etc.) şi falsă pentru toate valorile lui `x` mai mari decît `2` (de exemplu `3, 4, 4.5` etc.)

Un caz mai general de condiţie este cel în care variabilele sau constantele numerice despărţite de simbolurile operaţiilor relaţionale sînt înlocuite de expresii aritmetice.

De ***exemplu***:

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

exprimă condiţii. Astfel prima condiţie este adevărată atunci cînd valoarea expresiei `x+1` este mai mare decît `3`.

5.2. Expresii logice

Exprimarea matematică "$z \in [1,2]$" utilizată anterior nu a putut fi transpusă cu ajutorul condiţiilor introduse. In continuare vom cunoaşte alte operaţii care ne vor permite să transpunem in limbajul BASIC această expresie matematică. Expresiile condiţionale prezentate în paragraful anterior pot fi combinate cu ajutorul operaţiilor logice **AND, OR** şi **NOT** obţinînd **expresiile logice**. Expresiile logice la fel ca şi cele condiţionale pot fi **adevărate** sau **false**.

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

De *exemplu*:

```
x<2 AND x>0
```

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

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

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

O condiţie precedată de NOT este o expresie logică **adevărată** atunci cînd condiţia este **falsă**.

De *exemplu*:

```
NOT x<3
```

este adevărată atunci cînd x>=3. Evident acest exemplu poate fi scris cu ajutorul condiţiei

```
x>=3.
```

Operaţiile AND, OR, NOT pot lega nu numai condiţii, dar chiar şi expresii logice.

Exemplul 5.1. Expresia logică

```
x<3 AND y>0 OR z+1<>-1
```

se evaluează astfel:

► întîi se determină condiţiile x<3, y>0, z+1<>-1;

► apoi se determină expresia logică x<3 AND y>0, care este adevărată dacă x<3 şi y>0 sînt adevărate simultan;

► ultima expresie determinată este cea indicată prin operaţia OR şi este adevărată dacă expresia logică determinată anterior este adevărată sau condiţia z+1<>-1 este adevărată.

Observaţii:

♦ Dacă într-o expresie logică apar mai multe operaţii logice atunci se efectuează mai întîi operaţiile NOT, apoi AND şi la sfîrşit OR.

♦ În orice expresie logică înaintea operaţiilor NOT, AND, OR se efectuează operaţiile relaţionale ce definesc expresiile condiţionale.

♦ Dacă în condiţii apar expresii aritmetice, acestea din urmă se efectuează primele.

Exemplul 5.2. Fie expresia logică

```
x+3<a+b-1 AND 2*x>=y+1 .
```

Întîi se calculează expresiile aritmetice x+3, a+b-1, 2*x, y+1, apoi se efectuează operaţiile relaţionale ce definesc condiţiile x+3<a+b-1, 2*x>=y+1. Ultima operaţie efectuată este operaţia logică AND.

Observaţie:

♦ În anumite variante ale limbajului BASIC este prezentă o operaţie logică asemănătoare operaţiei OR, notată de obicei XOR şi numită **sau exclusiv**. Două condiţii legate cu XOR furnizează o expresie logică adevărată atunci cînd o condiţie este adevărată, iar cealaltă este falsă.

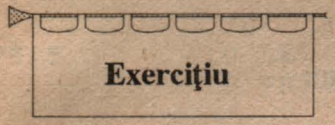
5.3. Instrucţiunea IF

Instrucţiunea **IF** defineşte o secvenţă de instrucţiuni care sînt executate numai dacă sînt îndeplinite (adevărate) anumite condiţii.

Exemplul 5.3. Următorul program calculează suma a două numere pozitive care sînt citite la execuţia 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"
```

Instrucţiunile `PRINT a+b` şi `STOP` sînt executate atunci cînd expresia logică `a>0 AND b>0` este adevărată.



Rescrieţi acest exemplu utilizînd în linia 30 o expresie logică formată cu ajutorul operaţiei `OR`.

Forma instrucţiunii `IF` este următoarea:

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

unde `exp_log` indică o expresie logică (în particular o condiţie), iar `S1, S2, ..., Sn`, instrucţiunile care se execută atunci cînd `exp_log` are valoarea adevărată. Dacă `exp_log` are valoarea falsă atunci instrucţiunile `S1, S2, ..., Sn` nu sînt executate şi prima instrucţiune care se execută este cea care urmează după instrucţiunea `IF`.

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

5.4. Instrucţiunea GO TO

Instrucţiunea GO TO are ca efect întreruperea execuţării secvenţiale a programului şi transferarea execuţiei la o altă linie. De exemplu instrucţiunea:

```
10 GO TO 40
```

trece controlul la instrucţiunea de la lina 40 care urmează să fie executată.

Prin urmare instrucţiunea 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 şi apoi se merge la linia 50 (prin instrucţiunea GO TO) unde se afişează valoarea lui a (egală cu 1). Instrucţiunea de pe linia 40 nu este executată datorită instrucţiunii anterioare, GO TO 50.

Exemplul 5.5. Acest program citeşte două valori pozitive şi 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 cîteva 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
```

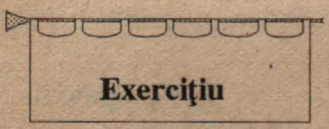
Primele două variante sînt simple, ultimele două sînt puţin mai complicate, deci necesită cîteva explicaţii. Vom comenta varianta 3, lăsînd ca pe ultima să o înţelegeţi singuri.

- Suma dată se va calcula în `s`, care iniţial are valoarea `0`.
- Instrucţiunile de pe liniile 30 şi 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 înţelege mai exect cum evoluează valorile variabilelor `a` şi `s` vă propunem să introduceţi în exemplul anterior instrucţiunea:

```
45 PRINT a, s
```

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

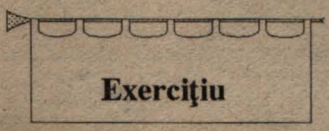


Vă propunem să modificaţi în varianta 3 liniile 10 şi 20 astfel:

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

Care este rezultatul afişat? Explicaţi rezultatul.

Observaţie: Variantele 3 şi 4, deşi mai complicate, pot fi uşor adaptate pentru a calcula sume cu mai mulţi termeni. Acest fapt nu este valabil şi pentru primele două variante.



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

☞ Reţineţi !

■ Operatorii relaţionali care definesc expresiile condiţionale sînt: `<, >, =, <=, >=, <>.`

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

■ Ordinea de efectuare a operaţiilor este următoarea:

- ▸ se efectuează operaţiile aritmetice (după ordinea cunoscută din matematică)
- ▸ se efectuează apoi operaţiile relaţionale
- ▸ ultimele se efectuează cele logice în ordinea `NOT, AND, OR.`

■ Instrucţiunea `IF` are forma

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

■ Instrucţiunea `GO TO` are forma

```
GO TO nr
```

unde `nr` indică numărul unei linii de program.

Exerciţii

1. Menţionaţi ce valori de adevăr au condiţiile

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

2. Calculaţi valoarea condiţiei `x+1<3*a`, pentru următoarele valori ale lui `x` şi `a` :

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

3. Determinaţi trei valori ale lui `x` pentru care condiţia `x+3>10` este adevărată. Aceeaşi problemă pentru `3*x<10, x+x<=x, x+2*x<>0`.

4. Fie expresiile logice:

(1) `a+b>3 AND a-b<1`
(2) `a<=3-b 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 OR a>=0`;
2. `a<0 AND a>=0`;
3. `NOT (a<0 OR a>=0)`;
4. `a<=1 AND a>-1`;
5. `a<1 AND a>0 OR a<3 AND a>2`;
6. `a<1 AND a<2 AND a<3`.

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

6. Fie expresia

```
a>-1 AND a<3 OR a>4 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 afişată pentru s? Explicaţi rezultatul.

8. Modificaţi programul de la exerciţiul 7 înlocuind linia 10 cu

```
10 LET i=1
```

Ce valoare este afişată pentru s? Explicaţi rezultatul.

9. Să se scrie un program care să calculeze suma

```
1·2+2·3+3·4+4·5+5·6+6·7+7·8.
```

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 şi să calculeze suma `1-1+1-1+1-1...`, cu n apariţii 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 execuţia programului se introduc pentru A, B, C, valorile 1000, 0.1 şi respectiv 50, atunci precizaţi ce rezultate sînt afişate. Explicaţi rezultatele obţinute.

13. Să se modifice programul de la exerciţiul precedent înlocuind, în linia 20, AND cu OR. Precizaţi cîteva valori pentru variabilele ce intervin în condiţia din linia 20 pentru care instrucţiunea 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
```

Precizaţi secvenţa de instrucţiuni care se execută şi rezultatul care se afişează. Scrieţi un program mai simplu ca cel de sus care să aibă acelaşi efect.

6

Scheme logice

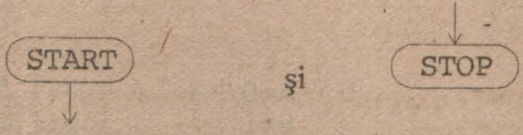
O problemă mai complexă necesită pentru a fi transpusă în program anumite explicaţii, o prezentare care să se detaşeze de limbajul de programare. O astfel de prezentare se poate face în limbajul natural, aşa cum am procedat pînă acum, dar acest mod de prezentare deşi uşor de urmărit are dezavantajul lipsei de precizie.

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

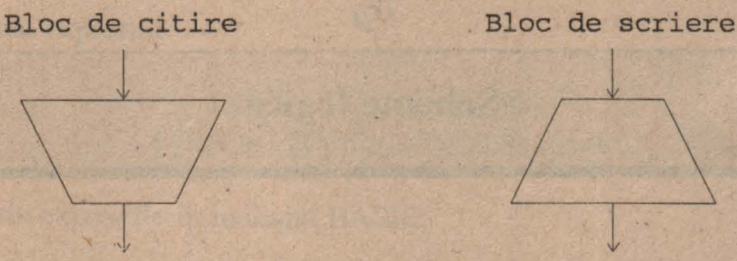
Elementele care compun o schemă logică sînt:

a) blocurile de început şi sfîrşit
b) blocurile de citire şi scriere
c) blocul de calcul
d) blocul de decizie.

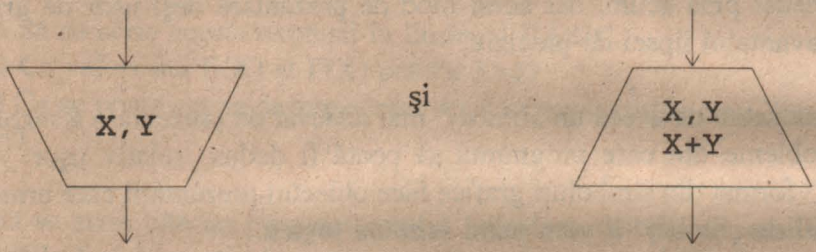
a) **Blocurile de început şi sfîrşit** se pun la începutul şi respectiv la sfîrşitul unei scheme logice şi au formele



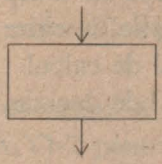
b) **Blocurile de citire şi scriere** se folosesc pentru a specifica operaţii de citire şi respectiv de scriere avînd formele:



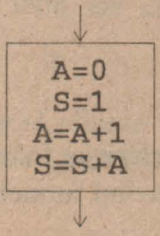
De exemplu pentru a indica faptul că se citesc variabilele X şi Y şi se scriu aceleaşi variabile şi suma acestora X+Y, folosim următoarele blocuri



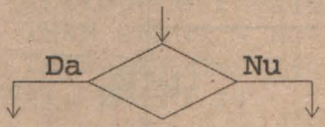
c) **Blocul de calcul** desemnează operaţii de calcul de tipul atribuirilor şi are forma



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

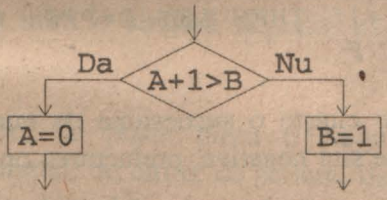


d) **Blocul de decizie** desemnează o condiţie sau o expresie logică (cu valoare adevărat sau fals) şi alternativele pentru cele două valori ale expresiei. Forma acestui bloc este:



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

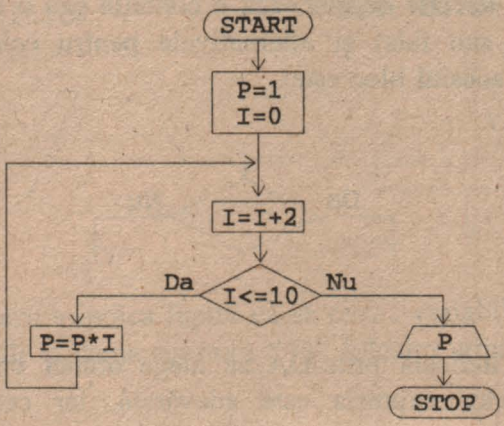
De exemplu, pentru a exprima faptul că în cazul în care expresia `A+1>B` este adevărată se execută `A=0`, iar în caz contrar se execută `B=1`, folosim:



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

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

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

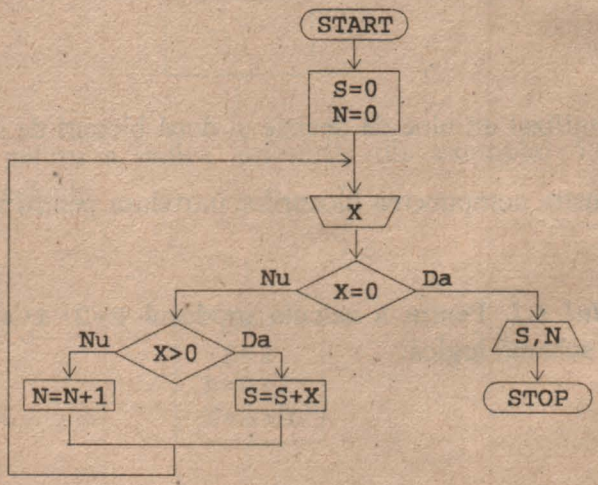


Din schema logică de mai sus putem deduce următorul program în limbaj BASIC:

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 citeşte o succesiune de numere şi se adună cele pozitive, se numără cîte sînt negative, prelucrarea oprindu-se atunci cînd se introduce 0.



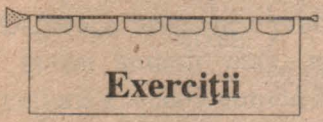
Programul BASIC care se deduce din schema logică anterioară:

P6.2
```
10 S=0 : N=0
20 INPUT "Introduceti 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
```

☞ Reţineţi !

■ Blocurile ce alcătuiesc o schemă logică sînt:
- ▸ blocul de început/sfîrşit
- ▸ blocul de citire/scriere
- ▸ blocul de calcul
- ▸ blocul de decizie

■ Pentru blocurile descrise facem următoarele precizări:
- ▸ în fiecare bloc intră o singură săgeată, excepţie 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 săgeţi (din blocul de decizie), fie nu pleacă nici o săgeată (din blocul de sfîrşit).



Exerciţii

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 exerciţiile 9 şi 10 de la capitolul 5.

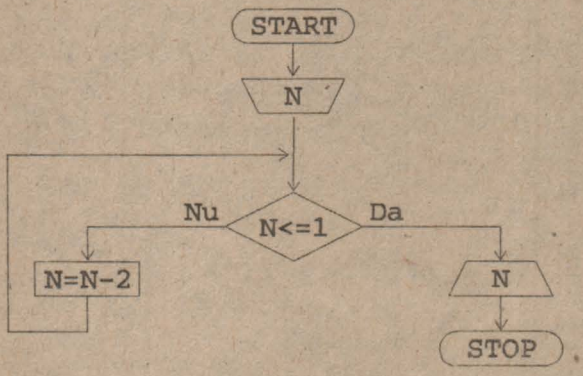
4. Să se conceapă schema logică pentru a rezolva problema determinării numărului de elemente mai mari decît o valoare specificată. Elementele se citesc succesiv.

5. Să se conceapă schema logică şi programul pentru calcularea sumei

$$1^2+2^2+...+8^2 .$$

6. Să se facă o schemă logică pentru a decide dacă un număr întreg se află între două valori A şi B (condiţia iniţială este A<=B).

7. Se dă schema logică



a) Ce problemă rezolvă această schemă logică?

b) Să se scrie programul în limbaj BASIC.

c) Găsiţi o generalizare a acestei probleme.

7

Instrucţiunea FOR...NEXT. Valoarea absolută, partea întreagă şi radicalul (ABS, INT, SQR)

7.1. Instrucţiunea FOR...NEXT

Reamintim programul prezentat în exemplul 5.6, varianta 3, care calcula suma 1+2+3+...+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 instrucţiunile de pe liniile 30, 40, 50 se execută de 10 ori, iar variabila a ia succesiv valori întregi cuprinse între 1 şi 10. Dintre aceste trei instrucţiuni, ultima, adică instrucţiunea de pe linia 50, are doar rolul de a controla numărul de execuţii al grupului de instrucţiuni de pe liniile 30 şi 40.

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

Rescriem exemplul de mai sus utilizînd această instrucţiune.

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 instrucţiunea `FOR...NEXT` conţine o singură instrucţiune, 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 "evoluţia" instrucţiunii 30 în timpul execuţiei programului este următoarea:

```
s=s+1      (iniţial 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 paşi rezultatul cuprins în `s` va fi `s=0+1+2+3+...+10`.

Exemplul 7.2. Ne propunem să calculăm valoarea

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

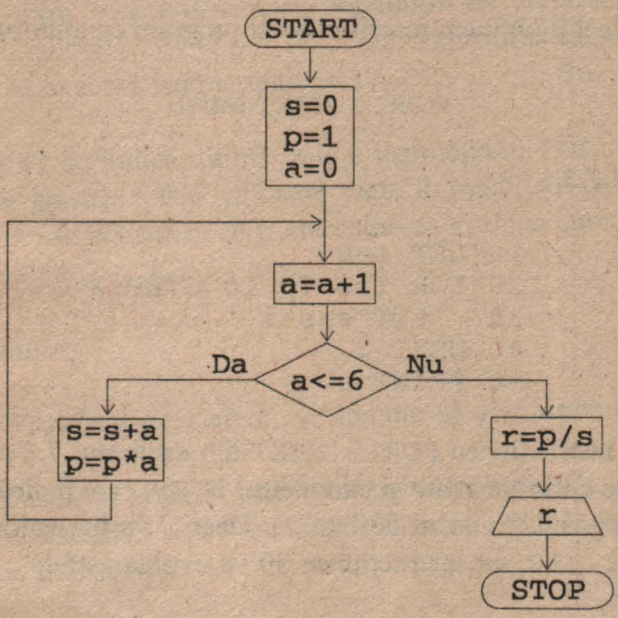
Programul în limbajul BASIC este următorul:

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 **exerciţiu** rescrierea acestui program fără a utiliza intrucţiunea FOR...NEXT.

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

Deducem astfel că acest program calculează suma

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

Acelaşi efect îl obţinem rescriind acest program cu ajutorul instrucţiunii 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 instrucţiunea FOR...NEXT din exemplul 7.3 are un element nou; în linia de definiţie apare şi parametrul STEP, care indică valoarea care măreşte pe i, pasul (în cazul de faţă 2). Deci i va lua valorile succesive: 2, 4, 6, 8, 10, iar instrucţiunea 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 instrucţiuni este:

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

unde vi şi vf sînt expresii aritmetice ce desemnează **valoarea iniţială** şi respectiv **valoarea finală** între care se desfăşoară execuţia acestei instrucţiuni. Prin v este desemnată **variabila de ciclare** care ia succesiv valorile vi, vi+p, vi+2p,..., vf. Pentru fiecare din valorile lui v de mai sus se execută instrucţiunile S1, S2,..., Sn.

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

Observaţii:

♦ Variabila de ciclare, v, trebuie să fie formată dintr-o singură literă.

♦ Pasul, p, poate fi orice valoare număr real.

♦ Condiţia vi≤vf nu este obligatorie după cum rezultă şi 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+...+1.

♦ Instrucţiunile S1, S2,...,Sn din interiorul instrucţiunii FOR...NEXT pot fi instrucţiuni FOR...NEXT, în acest caz acestea numindu-se instrucţiuni 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
```

Instrucţiunile 40-60 definesc o instrucţiune `FOR...NEXT` interioară celei definite de liniile 30-80.

♦ Deşi rezultă din definiţia instruţiunii `FOR...NEXT`, subliniem totuşi faptul că două instrucţiuni `FOR...NEXT` nu pot fi decît separate sau una interioară celeilalte. Cazul unei "acoperiri parţiale", ca în exemplul care urmează, este **greşit**:

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 afişează pe ecran valorile `10, 9, 8, 7, 6, 5, 4`, fiecare pe cîte 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 execuţii al instrucţiunilor ce compun o instrucţiune 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 şi radicalul - ABS, INT, SQR

Cunoaştem din matematică faptul că **valoarea absolută** sau **modulul** unui număr a este notat cu |a| şi definit prin: |a|=a pentru a ≥0, iar |a|=-a, pentru a<0.

În limbajul BASIC, dacă a este o variabilă ce conţine 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 execuţia acestui program pentru instrucţiunea INPUT se tastează 10 atunci pe ecran se afişează valoarea 10, iar dacă se tastează -10, se afişează tot valoarea 10.

O altă noţiune matematică utilizată în limbajul BASIC este **partea întreagă** a unui număr x, notată cu [x].

Matematic, [x] se defineşte ca fiind cel mai mare număr întreg mai mic sau egal decît x.

Exemple: [3,14]=3, [3]=3, [5,99]=5, [0,1]=0, [0,99]=0, [-0,1]=-1, [-0,99]=-1, [-5,99]=-6, [-3]=-3.

În limbajul BASIC, dacă o variabilă a conţine o valoare, atunci prin

```
INT a
```

se desemnează partea întreagă a valorii din variabila a.

Exemplul 7.7.

P7.10

```
10 INPUT a,b
20 PRINT a/b, INT(a/b)
```

Dacă la executarea acestui program se introduc valorile 3 şi 2 pentru variabilele a, respectiv b, atunci instrucţiunea 20 va afişa următoarele rezultate:

```
 1.5                1
```

Remarcăm faptul ca parantezele din expresia INT(a/b) sînt obligatorii pentru a executa întîi împărţirea lui a la b şi pentru a calcula apoi partea întreagă a rezultatului.

Rădăcina pătrată dintr-un număr pozitiv a, reprezentată în matematică prin simbolul $\sqrt{}$, este acel număr pozitiv b care are proprietatea $b^2=a$. În limbajul BASIC rădăcina pătrată dintr-un număr a este reprezentată prin simbolul SQR. Astfel dacă o variabilă conţine o valoare

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** tastînd 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

```
INT (x + y)
```

se efectuează astfel: se calculează întîi x + y şi apoi se calculează partea întreagă a rezultatului.

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

```
INT (RND*6) .
```

7.3. Probleme de divizibilitate

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

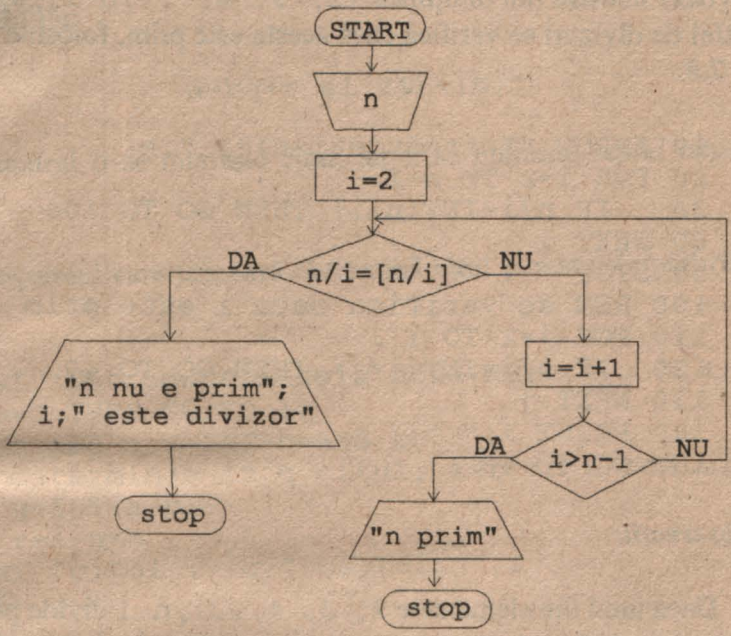
Reamintim că un număr natural n este prim dacă are ca singuri divizori pozitivi pe 1 şi pe n. De exemplu

```
2, 3, 5, 7, 11
```

sînt numere prime.

Vom determina că n este prim, verificînd că nici unul din numerele 2, 3, 4, . . . , 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 găsiţi un algoritm mai bun decît cel de sus, care să testeze divizorii pînă la [n/2] şi numai pentru numere impare. Încercaţi să demonstraţi că aceste condiţii sînt suficiente pentru determinarea calităţii de număr prim.

Exemplul 7.9. Să se afişeze toţi divizorii primi ai unui număr natural n. Metoda de rezolvare constă în următoarele: se verifică dacă n admite divizori numere din mulţimea {2, 3, 4, . . . ,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
```

Observaţii:

♦ Dacă unul din elementele 2, 3, 4, . . . , n-1 divide pe n atunci se merge la linia 100 unde începe secvenţa de instrucţiuni ce verifică dacă divizorul lui n este prim.

♦ După verificarea calităţii de număr prim a divizorului i se revine la instrucţiunea FOR. . .NEXT cu GO TO 40.

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

```
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 ştiut din matematică următorul rezultat, cunoscut sub numele de **teorema de împărţire cu rest**: "pentru orice două numere întregi a, b cu b<>0, există două numere întregi q şi r care satisfac relaţiile:

$$a=bq+r \text{ şi } 0 \le r < |b|"$$

Numărul q se numeşte *cîtul*, iar r se numeşte *restul împărţirii* lui a la b.

Programul care urmează determină cîtul şi restul împărţirii 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
```

☞ Reţineţi !

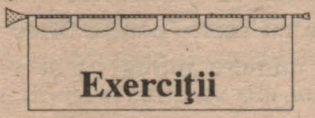
■ Instrucţiunea FOR...NEXT are forma

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

■ Instrucţiunile FOR...NEXT nu se pot suprapune parţial.

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

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



Exerciţii

1. Să se rezolve exerciţiile 9 şi 10 de la capitolul 5 utilizînd instrucţiunea `FOR...NEXT`.

2. Să se rezolve exerciţiile 4 şi 5 de la capitolul 6 utilizînd instrucţiunea `FOR...NEXT`.

3. Pentru calcularea sumei

```
1+1/(1*2)+1/(1*2*3)+...+1/(1*2*...*10)
```

să se întocmească o schemă logică şi un program. Aceeaşi problemă pentru suma

```
1+1/(1+2)+1/(1+2+3)+...+1/(1+2+...+10)
```

Generalizare. Modificaţi programul pentru a calcula

```
1+1/(1*2)+...+1/(1*2*...*n),
```

respectiv

```
1+1/(1+2)+...+1/(1+2+...+n),
```

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

4. Să se completeze programul prezentat în exemplul 7.5 astfel încît instrucţiunile de pe liniile 40 şi 50 să se execute de 5 ori.

5. Să se calculeze `INT x` şi `ABS x` pentru `x` luînd valorile `2.4, 2.9, 10.10, -0.1, -0.5, -1.1, -10`. Scrieţi un program care să verifice răspunsurile date.

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

Indicaţie. Se iau pentru a toate valorile cuprinse între 1 şi 999 şi se verifică următoarea condiţie: 17 divide a şi 19 divide (1000-a).

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

Indicaţie. Se consideră toate valorile lui a şi b cuprinse între 0 şi 9 şi se verifică următoarea condiţie

9 divide (3+a+2+b).

8. Să se determine numărul numerelor naturale prime mai mici decît n pentru n>=3 dat.

Indicaţie. Se consideră numerele 3, 5, 7, 9, . . . , 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 decît n.

9. Să se calculeze cîte din cele n numere care se citesc în variabila x satisfac condiţia

1 <= SQR x <= 2.

8

Instrucţiunile READ, DATA, RESTORE

Am văzut că introducerea datelor într-un program se face cu instrucţiunea INPUT. O altă modalitate de introducere a datelor este oferită de instrucţiunile **READ** şi **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
```

instrucţiunea de pe linia 10 atribuie lui a şi b valorile 10 şi respectiv 20 specificate în instrucţiunea DATA. Cele două instrucţiuni READ şi DATA funcţionează în mod similar instrucţiunii INPUT cu menţiunea că datele care se introduc sînt specificate în program prin instrucţiunea DATA.

Forma instrucţiunii DATA este următoarea :

```
DATA lista
```

unde lista este o listă de valori despărţite cu virgulă.

Exemplu:

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

Forma instrucţiunii READ este următoarea:

```
READ lista
```

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

Exemplu:

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

iniţializează variabilele a,b,c cu valori din instrucţiunea DATA.

Observaţii:

- Într-un program pot exista mai multe instrucţiuni READ şi DATA.
- Instrucţiunea 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 citeşte în variabila a valorile 1, 2, 3, 4, 5, 6 specificate în cele două instrucţiuni DATA şi le afişează pe o linie despărţite de cîte un spaţiu.

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

P8.3

```
10 REM Program pentru calcularea lunii şi zilei
20 REM corespunzatoare unei valori intre 1 si 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 obţinem că 7 martie este într-o luni.

Pentru a rezolva problema procedăm astfel:

a) codificăm zilele săptămînii astfel: luni=1, marţi=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 instrucţiune 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 împărţirii rezultatului de la punctul c) la 7. Restul obţinut 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 marţi.

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 **exerciţiu** efectuarea schemei logice pentru acest exemplu.

Dacă dorim ca un set de date să fie reluat atunci se foloseşte instrucţiunea **RESTORE**. Prima instrucţiune READ ce urmează să se execute după RESTORE va lua lista de valori de la prima valoare a primei instrucţiuni DATA.

Exemplul 8.4. Presupunem că avem două secvenţe de valori pe care întîi le adunăm, iar apoi le înmulţim:

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 afişa rezultatele: `36` pentru `s` şi `40320` pentru `p`, adică `s=1+2+3+4+5+6+7+8` şi `p=1*2*3*...*8`. Calcularea valorii lui `p` (liniile de program 80-140) a fost posibilă ca urmare a utilizării instrucţiunii `RESTORE` în linia 90, care a condus la reluarea secvenţei de valori specificate în instrucţiunile `DATA` de la prima valoare a primei instrucţiuni `DATA`.

Exemplul 8.5. Aflarea celui mai mare element dintr-un şir de valori specificate cu instrucţiunea `DATA`. Se consideră un şir 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
```

☞ Reţineţi !

■ Instrucţiunea DATA are forma:

```
DATA v1, v2,...,vk
```

unde v1, v2,...,vk sînt constante numerice.

■ Instrucţiunea READ are forma:

```
READ a1, a2,...,an
```

şi are ca efect trecerea în variabilele a1,a2,...,an a n valori consecutive specificate în instrucţiuni DATA.

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

Exerciţii

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 exerciţiile 1 şi 2 de la capitolul 3 utilizînd în locul instrucţiunilor INPUT, instrucţiuni READ şi DATA.

3. Să se scrie un program ce calculează suma a 6 valori specificate într-o instrucţiune DATA.

4. Să se scrie un program care să calculeze cîte elemente mai mari ca 0 se află într-o succesiune de valori specificate într-o instrucţiune DATA.

5. Să se calculeze suma şi produsul a 8 valori numerice specificate într-o instrucţiune DATA, parcurgînd o singură dată această secvenţă.

6. Să se modifice programul din exemplul 8.5 pentru a determina cel mai mic element din şir.

7. Să se scrie un program care să determine cel mai mare şi cel mai mic element dintr-un şir de valori specificate într-o instrucţiune DATA, folosind eventual instrucţiunea RESTORE.

9

Şiruri de caractere. Variabile şi expresii de tip şir de caractere

Şirul de caractere a fost prezentat o dată cu instrucţiunile de citire şi de scriere. Reamintim că un şir de caractere se specifică printr-o succesiune de caractere cuprinse între " (ghilimele).

Exemplu:

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

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

```
a$, b$, x$
```

reprezintă variabile de tip şir 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 şir de caractere şi cu şiruri de caractere se pot efectua o serie de operaţii ca:

- **atribuiri**
- **comparaţii**
- **concatenări** (alipiri de şiruri de caractere).

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

● **Comparaţiile** între şiruri de caractere (variabile de tip şir de caractere) au rezultatul **adevărat** sau **fals** ca şi comparaţiile între expresii numerice.

În acest capitol vom considera doar şiruri de caractere care conţin **litere mari sau mici** din alfabetul latin. Dacă şirurile conţin **numai litere mari** sau **numai litere mici**, compararea se poate efectua ţinînd cont că **două litere pot fi comparate pe baza poziţiilor lor în alfabet.** Astfel, comparaţiile `"a" < "b"`, `"u" >"d"`, `"M" < "Q"`, `"R" < "S"` sînt adevărate deoarece `"a"` se află în faţa lui `"b"`, `"u"` după `"d"`, `"M"` înaintea lui `"Q"` şi, respectiv `"R"` în faţa lui `"S"`.

? Ce puteţi spune despre comparaţiile:

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

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

`"ANA" < "ANICA"` - este **adevărată** deoarce prima diferenţă dintre literele ce compun cele două cuvinte apare pe poziţia a 3-a unde primul şir are `"A"`, iar al doilea `"I"`, iar `"A" < "I"` este adevărată (primele două poziţii sînt identice).

`"BARBU" > "BARBESCU"` - este **adevărată**, deoarece pe poziţia a 5-a în primul şir apare `"U"`, iar pe aceeaşi poziţie în al doilea apare `"E"` (poziţiile de la 1 la 4 coincid în ambele cazuri).

`"ANA" > "BARBU"` - este **falsă** deoarce poziţia întîi din primul şir este `"A"`, iar poziţia corespunzătoare din al doilea şir este `"B"`.

`"casa" <= "masa"` - este **adevărată** deoarece primele poziţii din cele două şiruri conţin `"c"` respectiv `"m"`, iar `"c" < "m"`.

"casa" <> "masa" - este **adevărată** deoarece cele două şiruri nu conţin aceleaşi componente.

? Ce puteţi spune despre următoarele comparaţii:

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

Dacă se compară caractere reprezentînd litere mici şi caractere reprezentînd litere mari atunci trebuie să se ţină cont că:

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

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

iar comparaţiile: "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

? Puteţi spune care din următoarele condiţii sînt adevărate:

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

Avînd aşezarea 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 şiruri sînt astfel încît primul se identifică cu primele litere din al doilea şir, atunci primul este mai mic decît al doilea. De exemplu:

"POP" < "POPA", "POPA" < "POPAZU", "DAN" <= "DANIEL"

sînt adevărate.

? Ce puteţi spune despre:

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

Şirurile 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 şi acestea se pun unul înaintea celuilalt dacă la compararea acestora cel care se pune primul are pe prima poziţie pe

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

Ordonarea şirurilor de caractere ţine însă seama şi de diferenţa între litere mari şi litere mici aşa cum am văzut mai devreme.

● Fiind date două şiruri "ab" şi "cd" numim şir de caractere obţinut prin **concatenarea** acestora, şirul "abcd". Operaţia 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$="abc"
30 PRINT a$
```

Şirul `"abc"` este atribuit variabilei `a$` (instrucţiunea 20), iar apoi acesta este afişat pe ecran (instrucţiunea 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 instrucţiunea `INPUT` un şir de caractere care dacă este `"BASIC"` atunci este executată instrucţiunea `PRINT` de pe linia 30, altfel se execută instrucţiunea 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 afişează numele fiecărei luni precum şi 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 încît să se renunţe la codificarea zilelor săptămînii.

```
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=l+1 TO 12
140  READ x
150 NEXT i
180 IF z<1 OR z>nl THEN GO TO 50
190 LET nrz=nrz-nl+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. Dîndu-se numele unei luni a anului, să se determine a cîta lună din an este şi cîte 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 şi Lucia de pe trei scaune S1, S2 şi respectiv S3, pe scaunele S3, S1 şi 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 şirul de caractere care este cel mai mare într-o mulţime de şiruri 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$
```

Observaţie: Într-un program pot fi utilizate variabile numerice şi variabile de tip şir de caractere cu acelaşi nume. Acestea sînt totuşi 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 afişa în urma execuţiei:

```
100 a
```

ceea ce arată că variabila `a` conţine `100` iar variabila `a$` conţine `"a"`.

☞ Reţineţi !

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

■ Numele unei variabile de tip şir de caractere este formată dintr-o literă şi din simbolul `$`.

■ Operaţii permise cu şiruri de caractere:

- **atribuirea**
- **comparaţiile** (`<`, `>`, `<=`, `>=`, `=`, `<>`)
- **concatenarea** (`+`).

Exerciţii

1. Să se scrie un program care citeşte două şiruri de caractere în două variabile şi afişează aceste şiruri despărţite de simbolul `"+"`.

2. Să se scrie un program care citeşte două şiruri de caractere şi afişează întîi pe al doilea şi apoi pe primul.

3. Să se scrie un program care citeşte două şiruri de caractere şi menţionează care din relaţiile =, <, >, <=, >=, <> sînt satisfăcute de cele două şiruri.

4. Arătaţi care din condiţiile 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 instrucţiuni:

```
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 şi numărul de zile. Folosind aceste instrucţiuni să se conceapă un program care citind o lună (ca un şir de caractere) să afişeze numărul de zile ale lunii.

6. Dîndu-se o mulţime de şiruri de caractere:

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

să se conceapă un program care să citească un şir de caractere şi să precizeze dacă acesta se află în mulţimea de mai sus.

7. Se dă o mulţime de şiruri de caractere. Să se scrie un program care să determine cel mai mic şir de caractere şi cel mai mare şir de caractere.

Indicaţie. Mulţimea de şiruri de caractere se poate specifica printr-o instrucţiune DATA. Încercaţi programul pentru următoarele şiruri: "A", "AN", "ANA", "M", "MA", "MAMA", "EU".

10

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 ecuaţiei de gradul I, determinarea unor elemente direct proporţionale 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 cunoştinţele dobîndite pînă acum referitoare la limbajul BASIC.

10.1. Rezolvarea ecuaţiei de gradul I

Ecuaţia de gradul I are forma:

```
ax+b=0
```

Se ştie că aceasta are soluţie unică, `x=-b/a` dacă `a<>0`. Pentru `a=0` sînt două cazuri: dacă `b=0` atunci ecuaţia are mai multe soluţii, iar dacă `b<>0` atunci ecuaţia nu are soluţie.

```
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 proporţionale

Să se determine trei numere x, y, z direct proporţionale cu a, b, c şi 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 relaţiile de mai sus obţinem:

$$y=(b/a)\cdot x \qquad (1)$$
$$z=(c/a)\cdot x \qquad (2)$$
$$x(1+b/a+c/a)=s \qquad (3)$$

Din aceste relaţii rezultă că numerele x, y, z se pot determina în mod unic, dacă ecuaţia (3) are soluţie 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 ţări sau localităţi este nevoie de o reprezentare a acestui loc la o anumită scară. Reprezentarea la scară, s, înseamnă că unei unităţi u de pe teren să-i corespundă pe hartă o altă unitate u' (s=u'/u). Cele două unităţi 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 şi B de pe teren este o distanţă de 100 de kilometri, atunci pentru a afla distanţa dintre punctele corespunzătoare de pe hartă, calculăm:

s=1/500000=x/100; x=100·s. Deci x=0,0002 km=20 cm.

În general, dacă scara este s, distanţa între puncte pe teren este d (în kilometri), atunci distanţa d' pe hartă (în centrimetri) 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 "Distanta pe harta: ";dp;" cm"
80 STOP
```

10.4. Determinarea unor mărimi cînd se cunoaşte procentul

Dîndu-se o mărime x şi 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 şi a restului împărţirii a două numere întregi (exemplul 7.11), folosind teorema de împărţire cu rest. Acelaşi rezultat se poate folosi şi pentru a determina cel mai mare divizor comun astfel:

Dîndu-se a, b numere întregi putem scrie că

$$a=b\cdot q+r_1 \text{ şi } r_1 < b \qquad (1)$$

acelaşi rezultat se poate apoi aplica pentru b şi r_1

$$b=r_1\cdot q_1+r_2 \text{ şi } r_2 < r_1 \qquad (2)$$

şi relaţia poate continua dacă $r_2 > 0$ prin

$$r_1=r_2\cdot q_2+r_3 \text{ şi } r_3 < r_2 \qquad (3)$$

pînă cînd

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

iar $r_k > 0$, deci

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

Vă propunem să demonstraţi că şirul de relaţii (1), (2), (3),...,(k) este finit şi că r_k este cel mai mare divizor comun al lui a şi b.

Trebuie remarcat că r_k poate fi şi 1, caz în care a şi 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 şi răspunsuri

Încercaţi programul următor, remarcînd utilizarea şirurilor 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 citeşte întrebarea în `i$` şi răspunsul corect în `r$`; răspunsul dat de la tastatură se află în `x$`. Dacă răspunsul este corect (adică `x$=r$`) atunci se afişează pe ecran `"Corect"`, în caz contrar se afişează `"Incorect"`.

Instrucţiunile `DATA` de pe liniile 140-180 conţin întrebările şi răspunsurile.

Exerciţii

1. Arătaţi care din următoarele secvenţe de instrucţiuni sînt corecte:

a) `10 LET A=10^2`
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 afişează instrucţiunea `PRINT s`? Explicaţi rezultatul.

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

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

`ax=by=cz şi x+y+z=s` pentru `a, b, c, s` 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` şi `b` (notat `cmmmc(a,b)`) ştiind că

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

6. Dîndu-se valorile `y` şi `p`, să se determine `x` astfel încît `y` să reprezinte `p%` din `x`. Să se scrie un program care să calculeze `x` avînd ca date de intrare `y` şi `p`.

7. Dacă prin **max(a,b)** înţelegem cel mai mare număr dintre `a` şi `b`, iar prin **min(a,b)** înţelegem cel mai mic număr dintre `a` şi `b`, să se scrie un program care să calculeze

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

pentru `a` şi `b` citite la execuţie.

8. Dîndu-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 cîte un program care să determine:

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

Indicaţie:
a) a poate lua valorile 0, 2, 4, 6, 8;
b) a se determină din relaţia 1+2+0+a divizibil cu 3;
c) x şi y satisfac relaţiile

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

d) relaţia 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. Ştiind că cmmdc(a,b,c)=cmmdc(a,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 şi al celui mai mare divizor comun al lui b şi c) să se scrie un program care să determine cmmdc(a,b,c) pentru a, b, c, trei numere naturale citite la execuţie.

12. Să se scrie un program care să determine:

a) ce distanţă parcurge un vehicul ce merge cu viteza v un timp t;

b) în cît timp parcurge un vehicul distanţa d dacă merge cu o viteză v.

Să se verifice programul pentru:

a) v=40 km/h şi t=1/2 h;

b) v=40 km/h şi d=100 km.

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

14. Să se calculeze suma:

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

ştiind că numerele $x_6, x_5, \ldots, x_0$ sînt prezente într-o instrucţiune DATA astfel:

$$\text{DATA } x_0, x_1, \ldots, x_6 \cdot$$

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 şir de 15 numere întregi. Să se calculeze următoarele sume de numere:

a) a celor care se află înaintea primului element egal cu 0;

b) 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 şirul 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 şi 2-2+1;

c) 1+2+1+2.

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

16. Completaţi enunţul problemei anterioare cu cerinţa ca sumele care se vor afişa să fie divizibile cu 2.

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

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

11

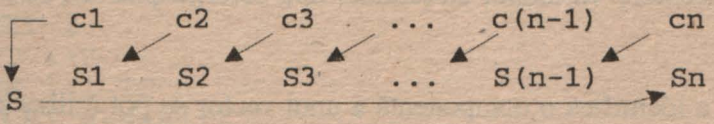
Variabile structurate

Pînă acum am folosit variabile ce conţineau valori numerice sau şiruri de caractere. Aceste variabile le numim **simple** deoarece conţin la un moment dat o singură valoare numerică sau un singur şir. Pentru a reţine mai multe valori avem nevoie de mai multe variabile.

Reamintim exemplul 4.4 care schimba locurile ocupate de `Ana`, `Maria` şi `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 `S1, S2, S3,...,Sn` scaune ocupate de `c1, c2,c3,...,cn` copii şi dorim să facem următoarele schimbări `c2` să ocupe scaunul `S1`, `c3` să ocupe scaunul `S2,...,cn` să ocupe scaunul `S(n-1)`, iar `c1` să ocupe scaunul `Sn`. Mutările sînt sugerate de următoarea schemă:



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

- `c1` trece pe `S`
- `c2` trece pe `S1`, `c3` trece pe `S2,...,cn` trece pe `S(n-1)`
- `c1` (aflat pe `S`) trece pe `Sn`.

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 S1, S2, S3, S4, S5, iar pentru n=7 de S1, S2, S3, S4, S5, S6, S7 etc. Numărul instrucţiunilor de atribuire se modifică şi 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 conţinînd cîte o valoare.

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

```
DIM a(n)
```

unde a este o variabila structurată cu n componente; n are în mod necesar o valoare în momentul apariţiei în această instrucţiune.

Observaţii:

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

♦ O variabilă structurată ca cea de mai sus va mai fi numită şi **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 şi 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 citeşte 10 valori numerice într-un vector a şi 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 citeste in compenenta 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
```

Instrucţiunea de pe linia 30 citeşte valori numerice în variabilele `a(1)`, `a(2)`,...,`a(10)` (componentele variabilei structurate `a`).

Instrucţiunea de pe linia 70 calculează suma

$$a(1)+a(2)+...+a(10)$$

Acest program poate fi uşor modificat pentru a funcţiona pentru vectori avînd 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 instrucţiunii din linia 10 se precizează valoarea lui `n`, deci instrucţiunea 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 şi de copii sînt arbitrare. Codificăm copiii c1, c2,..., cn cu 1, 2, 3,..., 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
```

Observaţi tehnica de declarare a vectorului S.

Pentru a înţelege mecanismul referirii la componentele unei variabile structurate, dezvăluim modul în care se fac atribuirile (linia 100) din instrucţiunea 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 şir. Dîndu-se un şir x1, x2,..., xn şi un element a, se cere să se construiască un nou şir x1', x2',..., x(n+1)', astfel:

- dacă elementul a nu este egal cu nici unul din elementele şirului dat atunci se pune la sfîrşitul acestuia, deci şirul de n+1 elemente este următorul:

x1, x2,..., xn, a

- dacă elementul a este egal cu un anumit element xi din şir, atunci se pune după acesta, deci şirul de n+1 elemente arată astfel:

x1, x2,..., xi, a, x(i+1),..., xn.

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 citeste elementul care se cauta in sir
 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 in acest caz a nu se afla in sir
140 LET x(n+1)=a: GO TO 220
150 REM in acest caz a se afla in sir pe pozitia i
160 REM elementele de la i+1 se muta catre 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 obtinut
230 FOR i=1 TO n+1
240  PRINT x(i)
250 NEXT i
260 STOP
```

Observaţie: Programul conţine o secvenţă pe care este indicat să o reţineţi (liniile 180-200), deoarece execută deplasarea către dreapta cu o poziţie a elementelor x(i+1),..., x(n).

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

Vom construi un program care pentru două mulţimi A şi B va calcula diferenţa, reuniunea şi intersecţia acestor mulţimi notate cu A-B, A+B, respectiv A*B.

Mulţimile vor fi reprezentate ca vectori conţinînd elementele mulţimilor. De exemplu, dacă A={1, 2, 5, 7}, atunci vom reprezenta această mulţime ca un vector A, definit cu DIM A(4), astfel încît A(1)=1, A(2)=2, A(3)=5, A(4)=7.

Înainte de a trece la prezentarea programului reamintim definiţiile matematice ale operaţiilor mai sus menţionate.

$$A - B = \{x \mid x \in A \text{ şi } x \notin B\}$$
$$A + B = \{x \mid x \in A \text{ sau } x \in B\}$$
$$A * B = \{x \mid x \in A \text{ şi } x \in B\}$$

Programul care urmează va citi întîi mulţimile A şi B, apoi le va afişa şi 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 mulţimi (varianta 2).

Vom relua problema din exemplul anterior, dar de această dată vom alege un alt mod de a reprezenta mulţimile. Vom presupune că lucrăm cu mulţimi avînd elemente cuprinse între 1 şi n. În acest caz mulţimea A va fi reprezentată de vectorul A definit cu DIM A(n), în care A(i)=1, dacă i∈A şi A(i)=0, în caz contrar.

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

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

şi

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

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
```

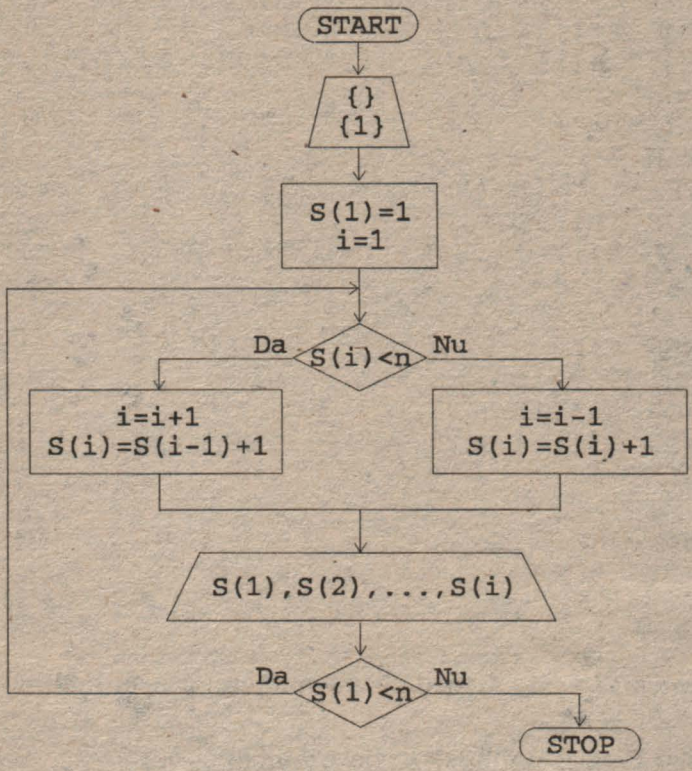
Exemplul 11.6. Generarea submulţimilor unei mulţimi.

Fiind dată o mulţime A, de exemplu A={1, 2, 4}, submulţimile acesteia sînt:

{} - mulţimea 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.

Mulţimile vor fi reprezentate ca în varianta 1 a operaţiilor cu mulţimi (exemplul 11.4).

Pentru a genera submulţimile unei mulţimi A care conţine elementele 1, 2, 3,...,n, considerăm următoarea schemă logică:



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

{}, {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}.

☞ Reţineţi !

■ Declararea 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.

Exerciţii

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 S1, S2, . . . , Sn ocupate de n copii c1, c2, . . . , cn. Să se scrie un program care să realizeze următoarele mutări:

- c1 trece pe scaunul S2
- c2 trece pe scaunul S3
...
- c(n-1) trece pe scaunul Sn
- cn trece pe scaunul S1.

3. Se citeşte un şir de n numere într-un vector. Să se determine cel mai mare şi cel mai mic element din şirul dat.

4. Se citeşte un şir de n numere într-un vector. Să se determine cel mai mare element şi cel mai mic element din şirul dat; să se indice în plus poziţiile din şir în care apar aceste elemente. Să se execute programul pentru n=5 şi pentru şirul 1, 2, 1, 2, 1. Trebuie să se obţină maximul egal cu 2 şi poziţia a 2-a, iar minimul egal cu 1 şi poziţia întîi.

5. Se citeşte un şir de n numere într-un vector. Să se determine cel mai mare şi cel mai mic element din şirul dat. Să se schimbe locurile acestor două elemente între ele. Să se afişeze şirul după modificările efectuate.

6. Se citeşte un şir de n numere într-un vector şi un element a. Dacă a este egal cu un element din şir atunci acest element se elimină din şir, altfel şirul rămîne nemodificat.

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

8. Să se scrie un program care pentru două mulţimi A şi B să determine dacă A este o submulţime a lui B.

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

```
Y(1)=X(1)+X(n), Y(2)=X(2)+X(n-1),

Y(3)=X(3)+X(n-3),...
```

Să se precizeze cîte elemente are Y şi 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 submulţimile de cîte m elemente ale unei mulţimi de n elemente. Să se verifice programul pentru m=2, n=4 (soluţia: {1,2}, {1,3}, {1,4}, {2,3}, {2,4}, {3,4}).

12. Să se scrie un program care să genereze submulţimile unei mulţimi X care conţin o mulţime Y. Se va verifica programul pentru X={1,2,3,4,5}, Y={1,3} (soluţia: {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 conţine n-1 elemente şi un loc liber pe poziţia i.

a) Să se transforme vectorul astfel încît poziţia liberă să fie j.

b) Să se transforme vectorul astfel încît poziţia liberă să fie prima poziţie (ultima poziţie).

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

```
1, 2,  ,-1, 2, 4, 3, 4
```

se transformă în

a)
```
1, 2,-1, 2, 4,  , 3, 4
```

b)
```
 , 1, 2,-1, 2, 4, 3, 4
```

```
1, 2,-1, 2, 4, 3, 4,  .
```

14. Se dă un vector cu n componente avînd ocupate primele r componente şi ultimele n-r-1 componente, componenta r+1 fiind liberă. Se cere să se obţină următoarea aranjare a elementelor în vector: ultimele n-r-1 componente iniţiale se pun pe primele n-r-1 poziţii, iar primele r componente iniţiale se pun pe ultimele r poziţii, iar poziţia 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 şi vectorul

```
-1, -1, -2,  , 1, 2, 0, 1
```

se ajunge la vectorul:

```
1, 2, 0, 1,  , -1, -1, -2 .
```

12

Subrutine

Există situaţii în care anumite părţi de program se repetă de două sau mai multe ori. De exemplu dorim să scriem un program care citeşte două numere, verifică dacă sînt pozitive şi le adună, apoi citeşte alte două numere verifică dacă sînt pozitive şi le înmulţeşte. 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 secvenţele de program 10-20 şi 40-50 sînt identice.

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

```
RETURN
```

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

```
GO SUB n
```

unde `n` este numărul de linie al primei instrucţiuni 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) citeşte `a` şi `b` şi verifică dacă acestea sînt pozitive. Subrutina este apelată în liniile 30 şi 60. Programul se execută astfel:

- se apelează subrutina (linia 30) care citeşte `a, b` şi 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 conţine `STOP` (linia 80) care este necesar în acest caz deoarece în lipsa acestuia ar fi fost executate, după calcularea produsului, instrucţiunile subrutinei.

Exemplul 12.2. Dîndu-se un şir de numere $x_1, x_2, \ldots, x_n$, să se determine poziţiile pe care se află primele două elemente în ordine descrescătoare din şir. De exemplu pentru şirul:

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

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

subrutină, vom pune apoi pe poziţia celui mai mare element o valoare foarte mică şi vom apela din nou subrutina, obţinînd astfel poziţia celui mai mare element dintre cele ale noului şir.

```
P12.3
  10 REM se citeste sirul
  20 INPUT "Cite elemente are sirul?:";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 in pozitia maximului, k, s-a pus -10000000
 100 GO SUB 1000
 110 LET MAX2=k
 120 PRINT MAX1;" ";MAX2
 130 STOP
1000 REM subrutina care determina pozitia
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 pozitia maximului este k
1070 RETURN
```

Exemplul 12.3. Căutarea unui element într-un şir.

Dîndu-se un şir $x_1, x_2, \ldots, x_n$ să se determine dacă un element a se află în acest şir. Programul pentru rezolvarea acestei probleme îl vom concepe astfel:

- se citeşte şirul $x_1, x_2, \ldots, x_n$ într-un vector x
- se citeşte elementul a
- se caută elementul a în şir, indicînd şi poziţia (pentru aceasta vom construi o subrutină).

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

- elementul se pune pe poziţia n+1 în şir
- se compară apoi elementele şirului cu a.

În urma acestei căutări a va trebui să se afle printre cele n+1 elemente, dar el se află în şirul iniţial dacă 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
```

Observaţie: Instrucţiunea RETURN este ultima instrucţiune care se execută într-o subrutină, dar nu trebuie să fie totdeauna pe ultima poziţie în subrutină. A se vedea pentru exemplificare instrucţiunea RETURN de pe linia 1030 din exemplul anterior.

Exemplul 12.4. Ordonarea crescătoare a elementelor unui şir.

Programul care rezolvă această problemă are următoarele etape:

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

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

Remarcăm faptul că determinarea maximului şi a poziţiei 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

utilizînd subrutine pentru a calcula suma, diferenţa, cîtul şi produsul elementelor a şi b (se presupune b<>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
```

☞ Reţineţi !

■ Apelarea unei subrutine se face cu

```
GO SUB n
```

■ Ultima instrucţiune 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.

Exerciţii

1. Dîndu-se două şiruri $a_1, a_2, \ldots, a_n$ şi $b_1, b_2, \ldots, b_n$, să se conceapă un program care să calculeze un şir $c_1, c_2, \ldots, c_n$, astfel încît $c_1=a_1+b_1, c_2=a_2+b_2, \ldots, c_n=a_n+b_n$.

Programul va utiliza o subrutină pentru a calcula şirul $c_1, c_2, \ldots, c_n$.

2. Aceeaşi problemă de la 1 cu menţiunea că şirul $c_1, c_2, \ldots, c_n$ este dat de $c_1=a_1+b_n$, $c_2=a_2-b_{n-1}$, $c_3=a_3+b_{n-2}$, $c_4=a_4-b_{n-3}$, ...

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 afişa acest program valoarea 2?

5. Scrieţi un program pentru realizarea operaţiilor elementare cu mulţimi (reuniunea, intersecţia, diferenţa) concepînd pentru fiecare din acestea cîte o subrutină.

6. Rescrieţi exemplele 12.3 şi 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` conţine `1, 2, 0, 3, 0, 1` şi după prelucrare devine `1, 2, 3, 1`.

8. Se dă un şir de elemente $x_1, \ldots x_n$ şi o valoare a. Să se scrie un program care să transforme acest şir astfel: toate elementele mai mici decît a rămîn neschimbate, iar cele mai mari decît a se schimbă cu a. De exemplu pentru şirul 1, 2, -1, 10, 20, 1 şi a=5 se obţine 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)=...=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:

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

folosind cîte o subrutină pentru a calcula suma şi produsul.

13

Instrucţiunea BEEP

Calculatorul efectuează aşa cum am văzut pînă acum o serie de operaţii (adunări, scăderi, înmulţiri, comparaţii etc.) cu ajutorul cărora am construit programe pentru rezolvarea anumitor probleme. Calculatorul însă poate să emită şi...sunete!. În continuare vom vedea cum putem folosi (programa) această facilitate.

Dacă nu aţi 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 aveţi puţină ... ureche muzicală şi dacă difuzorul calculatorului nu este dereglat, aţi recunoscut primele note ale gamei do major (`do, re, mi, fa`) şi faptul că toate au aceeaşi durată.

Din exemplul discutat deducem că forma generală a acestei instrucţiuni (este totuşi o instrucţiune!) este:

```
BEEP d,i
```

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

Pentru a putea transcrie o melodie (de pe un portativ) într-o secvenţă de instrucţiuni **BEEP** este necesar să cunoaştem ceva mai mult despre legătura dintre notele muzicale şi valorile înălţimii (parametrul al doilea din `BEEP`).

Iată cîteva valori ale înălţimii:

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

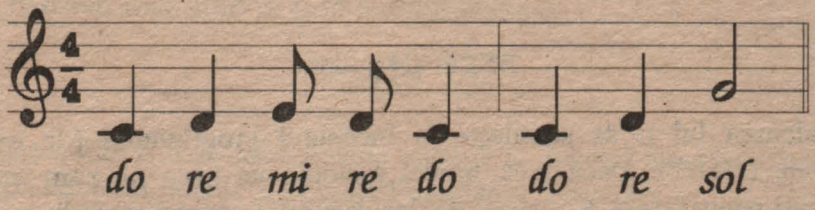
Dacă dorim note muzicale sub nota do din gama cu acelaşi nume, atunci luăm valori negative:

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

Aţi observat că diferenţa a două înălţimi consecutive este de 1 sau 2 după cum notele corespunzătoare determină un semiton, respectiv un ton.

Celălalt parametru al instrucţiunii BEEP, durata, se exprimă în secunde (1 - indică o secundă). Asocierea unei unităţi 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 şi pentru şaisprezecime un sfert de secundă.

Fie următorul portativ:



Programul care-l traduce în instrucţiuni 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 şi mai generală de a transpune un portativ într-un program este aceea dată de utilizarea instrucţiunilor READ, DATA, eventual RESTORE.

Transcriem cu această tehnică portativul anterior şi în acelaşi 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 "cinta" 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 "cinta" toate notele
  80   READ d,in
  90   BEEP d,in
 100 NEXT i
 110 STOP
```

Ştim 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 şi calculatorului nostru! Pentru aceasta înlocuim liniile 40 şi 90 cu

```
BEEP d*p,in.
```

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

Instrucţiunea BEEP poate fi folosită în programe şi pentru a semnala sonor faptul că se execută sau nu anumite secvenţe de instrucţiuni. De exemplu, dorim să introducem două numere pozitive pe care să le comparăm pentru a-l afişa pe cel mai mare; dacă la citire numerele nu satisfac condiţia de a fi pozitive, atunci se emite un semnal sonor şi 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 instrucţiuni, remarcăm necesitatea existenţei unei instrucţiuni care să modeleze pauzele ce apar între anumite note muzicale. Această instrucţiune este

```
PAUSE n
```

unde n indică numărul de unităţi. Reţinem că o secundă are circa 50 de unităţi. 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 Mărăcine" şi utilizează un parametru p care, dacă este modificat la diversele rulări schimbă ritmul melodiei (încercaţi 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
```

☞ Reţineţi !

■ Instrucţiunea BEEP are forma

```
BEEP d,in
```

unde d este durata, iar in este înălţimea.

■ Pauza între note se realizează cu

```
PAUSE n
```

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

Exerciţii

1. Scrieţi gama do major (`do, re, ..., do`) luînd ca durată unitatea (`BEEP 1,n`).

2. Scrieţi gama utilizînd un parametru pentru durată, pe care să-l introduceţi cu `INPUT`.

3. Rescrieţi exemplul 13.1 parametrizînd durata.

4. Scrieţi un program în care înainte de fiecare afişare pe ecran să emiteţi cîte un semnal sonor.

5. Alegeţi o melodie şi transcrieţi-o cu ajutorul instrucţiunilor `BEEP`, `READ`, `DATA`, eventual `PAUSE` şi `RESTORE`.

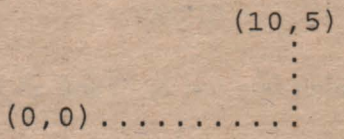
14

Elemente de grafică în limbajul BASIC

Ecranul pe care pînă acum am scris cu ajutorul instrucţiunii PRINT poate fi utilizat şi pentru a desena. Ecranul este împărţit în puncte care sînt identificate prin poziţia lor pe ecran. Poziţia unui punct pe ecran este indicată prin două coordonate: coordonata x (pe orizontală), cuprinsă între 0 şi 255 şi coordonata y (pe verticală) cuprinsă între 0 şi 175. Rezultă astfel că pe orizontală sînt cîte 256 de puncte, iar pe verticală cîte 176 de puncte. Orice punct de pe ecran are corespondent un cuplu în care prima valore arată coordonata x, iar a doua coordonata y. Ecranul apare marcat de următoarele coordonate extreme:

Pentru a determina un punct de coordonate (x,y), se parcurg pe orizontală x puncte (de la punctul de coordonate (0,0)) şi apoi pe verticală y puncte.

De exemplu punctul de coordonate (10,5) se determină plecînd din punctul de coordonate (0,0) şi deplasîndu-ne apoi spre dreapta 10 puncte şi în sus 5 puncte.



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

Pregătirea ecranului pentru a desena pe acesta, ştergînd toate informaţiile anterior scrise sau desenate, se face cu instrucţiunea `CLS`. După executarea acestei instrucţiuni, ecranul este şters, iar poziţia curentă este în punctul de coordonate `(0,0)`, adică în stînga jos.

Instrucţiunile de grafică ale limbajului BASIC ne permit să desenăm **puncte, linii şi cercuri.**

Desenarea punctelor se face cu instrucţiunea:

```
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)` şi `(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 instrucţiunea

```
DRAW x,y
```

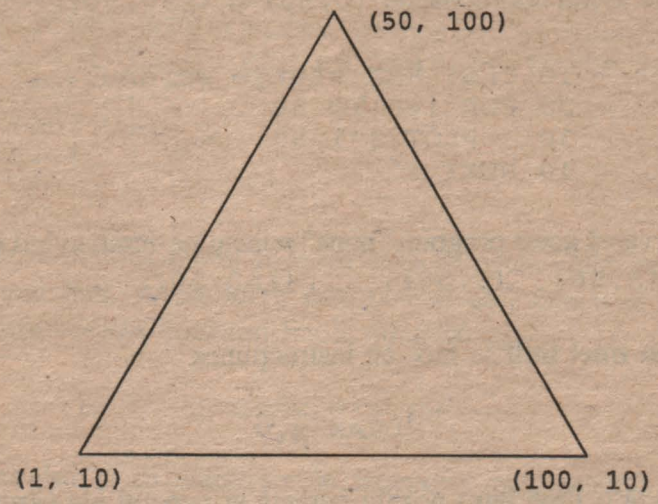
care desenează o linie între poziţia curentă şi punctul ale cărui coordonate se obţin adunînd la cele ale poziţiei curente `x`, respectiv `y`.

Exemplul 14.2. Programul trasează linii între punctele de coordonate (10,10) - (50,10) (linie orizontală), (100,100) - (100,50) (linie verticală) şi (20,20) - (50,50) (linie oblică). Pentru a trasa aceste linii trebuie ca înainte de utilizarea instrucţiunilor DRAW să folosim instrucţiunea PLOT pentru a obţine poziţia 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 instrucţiunii DRAW se pot trasa diferite figuri geometrice ca: triunghiuri, dreptunghiuri, pătrate etc.

Exemplul 14.3. Pentru a desena triunghiul de mai jos

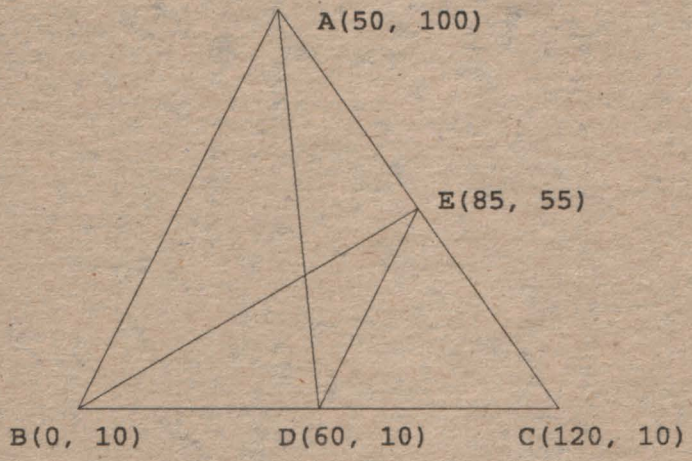


utilizăm următorul program:

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 instrucţiunii DRAW. Parametri necesari trasării vor fi calculaţi şi depuşi într-o instrucţiune DATA (linia 20). Aceşti parametri indică deplasările ce trebuie calculate pentru a trasa segmentele în ordinea indicată mai sus.

P14.4

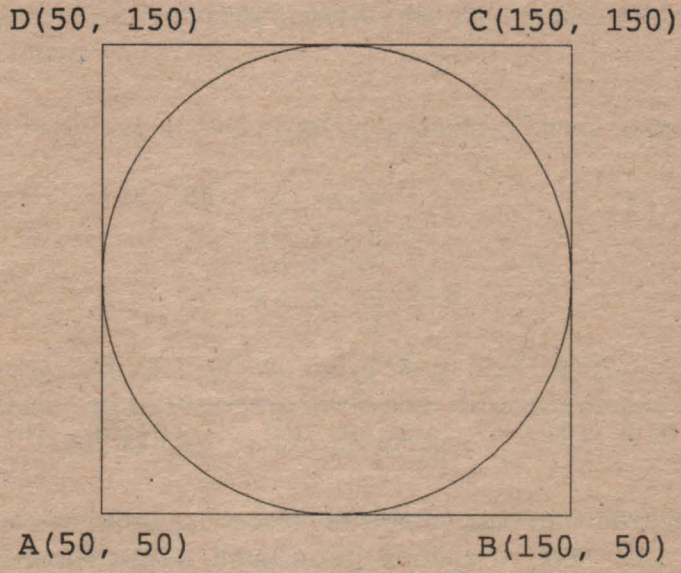
```
10 DATA -50,-90,120,0,-70,90,10,-90,25,45,-85,-45
20 CLS
30 PLOT 50,100
40 FOR i=1 TO 6
50   READ x,y
60   DRAW x,y
70 NEXT i
```

Trasarea cercurilor se execută cu ajutorul instrucţiunii:

```
CIRCLE x,y,r
```

unde x şi y indică poziţia centrului cercului, iar r raza acestuia.

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



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

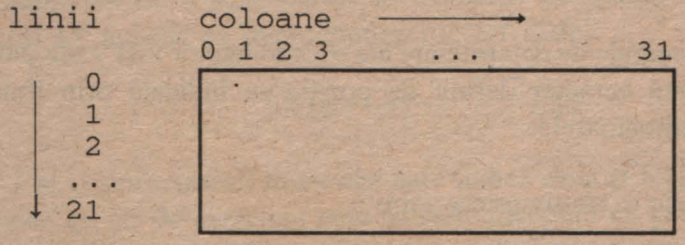
P14.5

```
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 cîteva precizări în legătură cu ecranul şi cu modul de scriere şi de desenare pe acesta.

Am văzut în acest capitol că instrucţiunile PLOT, DRAW, CIRCLE se referă la ecranul împărţit în 256x176 de **puncte**. În cazul instrucţiunilor de scriere, PRINT, ecranul este divizat în **caractere** în număr de 32x22 (32 coloane şi 22 linii). Corespondenţa dintre **caractere** şi **puncte** se face ţinînd cont de faptul că pentru a reprezenta un caracter sînt necesare 8x8 puncte. Pentru a depista corespondenţa exactă dintre caractere şi puncte menţionăm că identificarea unui caracter se face printr-o valoare indicînd **linia** şi printr-o valoare indicînd **coloana**. Valorile liniilor şi coloanelor sînt prezentate mai jos:

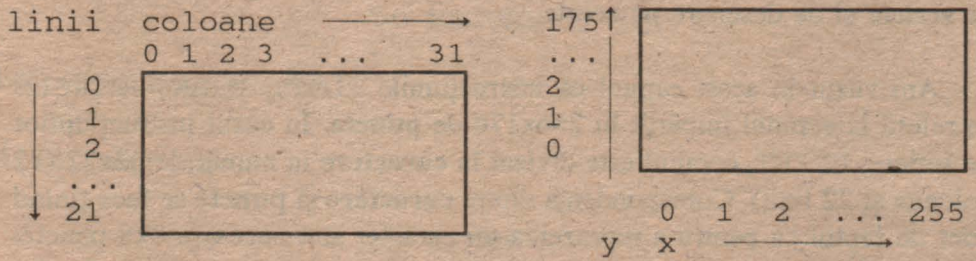


Rezultă astfel că pentru a determina poziţia unui caracter, specificată printr-un număr de linie, l, şi unul de coloană, c, se procedează astfel:

- din colţul din stînga sus se coboară l poziţii;
- din poziţia curentă se merge spre dreapta c poziţii.

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

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



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

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

- coordonatele y sînt cuprinse între

$$8\cdot(21-l) \quad \text{şi} \quad 8\cdot(21-l)+7$$

- coordonatele x sînt cuprinse între

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

Vă propunem ca **exerciţiu** să determinaţi formulele ce indică linia şi coloana unui caracter care acoperă un punct de coordonate x şi y date.

Referindu-ne la exemplul 14.4 putem deduce că vîrfurile A, B, C ale triunghiului au proprietatea că aparţin caracterelor ale căror poziţii indicate prin valorile liniilor şi 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 pute fi notată cu literele A, B, C în vîrfuri, dacă aceste caractere ar putea fi scrise pe poziţiile avînd următoarele coordonate:

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

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

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

unde l şi c sînt expresii numerice care indică linia şi respectiv coloana, iar sir este un şir de caractere care se scrie începînd cu poziţia 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"
```

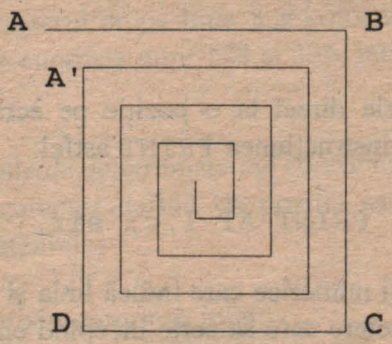
Exerciţiu

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

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

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

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



plecînd din punctul A şi urmînd sensul indicat de săgeată facem următoarele observaţii:

a) punctul A are coordonatele (x, y); lungimea iniţială a liniei este d, iar valoarea care o micşorează este p;

b) liniile: orizontală sus (AB) şi verticală dreapta (BC) au aceeaşi lungime d;

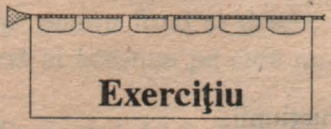
c) liniile: orizontală jos (CD) şi verticală stînga (DA') au aceeaşi lungime d-p (d fiind lungimea anterioară);

d) după ce s-a parcurs un ciclu de tipul descris la punctele b) şi c), valoarea d se modifică astfel: d=d-p. Dacă d>2·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 micsorare ";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
```

Instrucţiunile 10-30 citesc datele de intrare x, y, d şi p. Instrucţiunea 50 fixează coordonatele punctului de plecare. Instrucţiunea 70 este un apel de subrutină. Subrutina trasează 4 linii conform paşilor b) şi c) din algoritm. Trasarea se face cît timp condiţia d<=2*p este adevărată.



Exerciţiu

Completaţi acest program astfel încît să apară pe desen şi notaţiile pentru vîrfurile A, B, C şi D.

Realizarea desenelor devine şi mai atractivă dacă acestea pot fi făcute în culori. Limbajul BASIC oferă posibilitatea de a desena colorat, iar dacă dispuneţi şi 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)
- roşu (2)
- violet sau magenta (3)
- verde (4)
- albastru deschis sau cyan (5)
- galben (6)
- alb (7).

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

Trei sînt elementele care pot fi colorate:

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

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

```
PAPER nr
```

unde `nr` este o valoare între 0 şi 7.

Bordura se colorează prin invocarea instrucţiunii:

```
BORDER nr
```

unde `nr` este o valoare între 0 şi 7.

Desenele sînt executate cu instrucţiuni precum `PLOT`, `DRAW`, `CIRCLE`. Pentru a le trasa colorat se utilizează înaintea instrucţiunilor de mai sus, instrucţiunea:

```
INK nr
```

care prin `nr` indică o valoare între 0 şi 7.

Exemplul 14.7. Desenăm pe o foaie galbenă, cu bordură roşie o linie albastră şi 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
```

ATENŢIE !

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

Este recomandabil să nu daţi culori identice sau foarte apropiate ca nuanţe pentru fond şi desen, deoarece după execuţia programului obţineţi un text al acestuia pe care nu-l mai distingeţi 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 colţul din stînga sus şi `(55,10)` în colţul din dreapta jos.

Varianta 1.

P14.8

```
10 CLS
20 BORDER 2: PAPER 6: INK 1
30 PLOT 10,100: REM coltul din stinga 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
```

Observaţii:

♦ Remarcaţi diferenţa între vitezele de execuţie ale celor două programe.

♦ Programul din varianta 1 oferă totuşi mai multe posibilităţi de a umple (haşura) interiorul dreptunghiului. Varianta prezentată umple compact dreptunghiul. Vă propunem ca **exerciţiu** să umpleţi interiorul dreptunghiului cu alte haşuri. De exemplu, pe liniile pare puneţi puncte numai pe poziţiile pare, iar pe liniile impare numai pe poziţiile impare.

♦ Modificaţi ambele variante astfel încît să umpleţi interiorul dreptunghiului după cum urmează: vor fi trasate numai punctele de pe liniile pare.

♦ Modificaţi atît variantele 1 şi 2 de mai sus cît şi cele pe care le-aţi conceput conform observaţiilor anterioare, alegînd alte culori pentru bordură, fond şi desene.

Reţineţi !

- Ecranul grafic are 256x176 puncte (pe verticală şi pe orizontală).
- Iniţializarea 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 poziţie, a unui şir se face cu

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

- Stabilirea culorilor se face prin instrucţiunile

```
BORDER nr   bordura;
PAPER nr    fondul;
INK nr      desenul.
```

Exerciţii

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 cîte un cerc cu centrul în punctele specificate şi de rază 10.

3. Să se traseze un triunghi dreptunghic, un triunghi obtuzunghic şi unul ascuţitunghic.

4. Să se traseze un pătrat de latură 50 şi să se ducă diagonalele acestuia.

5. Să se traseze un triunghi şi să se unească apoi vîrfurile triunghiului cu mijloacele laturilor opuse.

6. Să se traseze un triunghi şi să se ducă dintr-un vîrf înălţimea şi să se unească acest vîrf cu mijlocul laturii opuse.

7. Să se traseze un pătrat, să se împartă apoi în patru pătrate egale şi să se umple interiorul pătratelor din stînga sus şi 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 execuţie.

9. Să se traseze un pătrat şi apoi să se unească mijlocul unei laturi cu vîrfurile opuse.

10. Trasaţi şapte cercuri de culori diferite.

11. S ă se traseze un triunghi şi să se construiască cu o linie punctată una din înălţimi.

15

Variabile structurate de tip tablou. Variabile structurate şiruri de caractere

15.1. Tablouri

În capitolul 11 am discutat noţiunea de variabilă structurată sau vector. În acest capitol vom cunoaşte o structură de date mai generală pe care o vom numi **tablou** şi care poate fi considerată ca fiind formată din mai multe date de tip vector aşezate unele lîngă altele. Dacă de exemplu, avem doi vectori de aceeaşi dimensiune şi îi aşezăm unul sub celălalt obţinem un tablou cu două **linii**. Fie vectorii avînd componentele `1, 2, 3, 4, 5` şi `-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 evidenţă **liniile** formate din componentele vectorilor. Putem însă să remarcăm şi elementele aşezate pe verticală: `1` şi `-1`, `2` şi `-2`, `3` şi `-3` etc. Aceste elemente formează cîte o **coloană** în tablou.

Un alt tablou de elemente este şi următorul:

```
1   1   2   3   4
0   2   3   4   5
1   0   2   0   1
```

Din exemplele şi explicaţiile de mai sus deducem următoarele caracteristici ale unui tablou:

- elementele aşezate pe orizontală formează **liniile** tabloului

(0 2 3 4 5 formează linia a doua în tablou);

- elementele aşezate pe verticală formează **coloanele** tabloului

(2 3 2 formează coloana a treia în tablou);

- toate **liniile** au acelaşi număr de elemente (în cazul ultimului exemplu, numărul de elemente este 5);
- toate **coloanele** au acelaşi 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 şi al unui vector este format dintr-o literă şi 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),...,t(1,5)
t(2,1), t(2,2),...,t(2,5)
t(3,1), t(3,2),...,t(3,5)
```

Exemplul 15.1. Programul citeşte 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 citeşte elementele unui tablou (ale cărui dimensiuni se specifică la execuţie) şi însumează elementele de pe linii afişînd la sfîrşit 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
```

Observaţie: Instrucţiunea de pe linia 60 va afişa `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 instrucţiunea `PRINT x(i,j)`.

Vă propunem să reţineţi sumele calculate pe linii într-un vector avînd numărul de componente egal cu numărul de linii ale lui x, iar după ce sînt toate sumele calculate să fie afişate.

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),...,X(1,m)`) în primele `m` elemente din `V`;
- a doua linie din `X` (elementele `X(2,1),...,X(2,m)`) în următoarele `m` elemente din `V` (`V(m+1),...,V(m+m)`) etc.

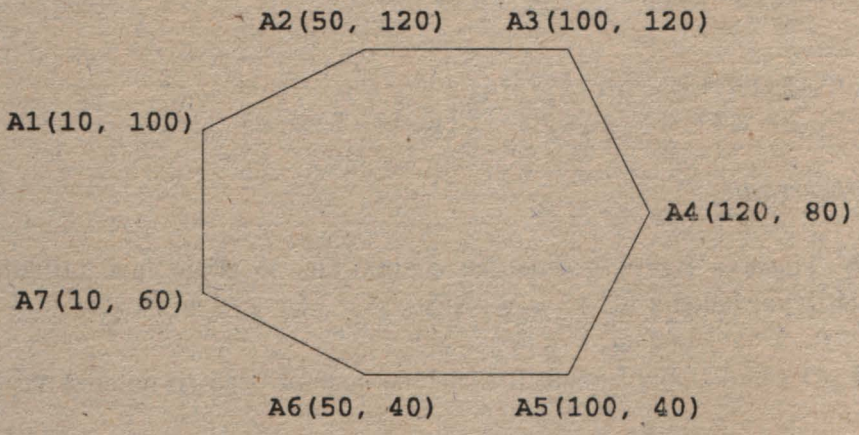
Deducem din cele de mai sus că la un moment dat vom transfera elementele liniei `i`, $1 \leq i \leq n$, din `X` (elementele `X(i,1), ..., X(i,m)`) în elementele `V((i-1)*m+1), ..., V((i-1)*m+m)`).

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 `A1, A2,...,A7`, care au coordonatele trecute în paranteză.



Tabloul care conţine 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)
```

Observaţii:

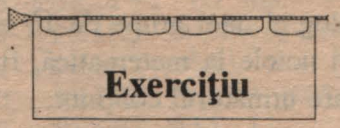
♦ Trasarea figurii geometrice de mai sus se poate face utilizînd şi facilităţile de culoare (INK, PAPER).

♦ Desenul obţinut 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 şir de date specificate într-o instrucţiune `DATA`.



Exerciţiu

Transpuneţi în programe observaţiile de mai sus.

15.2. Variabile structurate şiruri de caractere

În capitolul 9 am introdus noţiunile de şir de caractere şi variabilă de tip şir 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 şir de caractere.

La fel ca în cazul numeric şi în cazul şirurilor de caractere se pot considera variabile structurate (compuse) formate din mai multe componente, fiecare de tip şir de caractere. În acest caz trebuie însă precizată **lungimea maximă** a şirurilor ce alcătuiesc componentele.

De exemplu pentru a specifica o variabilă structurată de tip şir de caractere avînd trei componente de maximum 10 caractere fiecare, trebuie folosită o instrucţiune:

```
DIM a$(3,10)
```

Dacă dorim ca în aceste trei componente să punem şirurile "Ana", "Andrei", "Maria" atunci scriem

```
LET a$(1)="Ana"
LET a$(2)="Andrei"
LET a$(3)="Maria"
```

Observaţi că fiecare şir are cel mult 10 caractere.

Exemplul 15.4. Vom întocmi un catalog al unei grupe de 10 copii care conţine numele şi prenumele acestora şi notele la matematică, fizică şi informatică (în această ordine). Catalogul are următorul conţinut:

```
 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 şirului de caractere format cu nume şi prenume (s-a considerat şi poziţia reprezentînd spaţiul dintre nume şi prenume).

Acest catalog poate fi memorat în două variabile:

- o variabilă structurată şir de caractere, n$, care să conţină numele şi prenumele în ordinea de mai sus;

- o variabilă de tip tablou, s, cu 10 linii şi 3 coloane care să conţină notele la matematică (în coloana 1), la fizică (în coloana 2), la informatică (în coloana 3).

Următorul program citeşte informaţiile din catalog (nume, prenume, note la matematică, fizică şi informatică) şi apoi scrie acest catalog astfel încît 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:

- afişarea elevilor care au nota 10 la o materie (matematică sau informatică);
- afişarea elevilor care au nota 10 la matematică şi la informatică;
- afişarea elevilor care au numai note de 9 şi 10;
- calcularea mediei pe baza notelor la cele 3 obiecte;
- afişarea 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 rămîn ca **exerciţii.**

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
```

şi, evident, cu liniile programului **P15.6.**

Afişarea elevilor care au medii peste 8 se face în corelaţie 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
```

şi cu liniile programului **P15.6**

15.3. Alte operaţii cu şiruri de caractere

Operaţiile de comparare a şirurilor de caractere apar foarte frecvent şi de aceea este necesar să înţelegem exact cum se efectuează. Să ne reamintim că am învăţat deja cum se compară şiruri de caractere formate numai din litere mari sau numai din litere mici. Mulţimea caracterelor din care se pot selecţiona elemente pentru a forma şirurile conţine pe lîngă litere şi cifre, semne speciale (+, -, *, / etc.).

Pentru a efectua comparaţii între şiruri de caractere trebuie să ştim să facem comparaţii între caractere (şiruri de caractere formate dintr-un element).

Ştim că se pot efectua comparaţii între litere deoarece acestea apar într-o anumită ordine. De exemplu:

```
"A" <= "C"
```

deoarece A se află în faţa 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 **convenţie impusă de calculator**.

Celelalte simboluri care desemnează caractere (cifre, simboluri speciale) sînt şi acestea ordonate. Următorul tabel conţine în ordine principalele caractere utilizate în şirurile 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 comparaţii sînt adevărate:

```
"!" < "$", "%" < "(", "*" < "/", "<" < "=", "=" <
">", "A" < "[", "I" < "c", "m" < "{", "o" > "+", "p"
> ";".
```

Ce puteţi spune despre următoarele comparaţii ?

```
"." < "0", "7" > "(", "(" > "(", "A" < "8", "a" >
"9", "{" < "}" .
```

Putem să considerăm şi comparaţii între şiruri de caractere pe baza relaţiilor 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 şir coincide cu primele trei componenete ale celui de al 2-lea şir;

`"{{1}" >= "{{2}"` - este **falsă** deoarece `"1" < "2"` este adevărată.

Rezultă astfel că efectuarea comparaţiilor între şiruri de caractere se face după aceleaşi principii stabilite pentru şirurile de caractere ce conţineau numai litere, adică două şiruri se pun unul înaintea celuilalt dacă prima poziţie ce conţine elemente diferite în cele două şiruri, are în şirul de caractere care se pune primul, un caracter care se află în tabelul anterior înaintea caracterului corespunzător din al doilea şir de caractere.

Exemplele de mai sus evidenţiază aceste observaţii. Pe baza acestor ordonări ale şirurilor de caractere se pot stabili valorile de adevăr ale comparaţiilor.

Observaţii:

♦ Caracterul **spaţiu** poate face parte din şiruri de caractere. De exemplu: `"A B", " 12", "aA B".`

Trebuie remarcat că acest caracter se află **înaintea tuturor** elementelor din tabelul prezentat. Astfel următoarele comparaţii sînt adevărate:

```
" " < "!", " " < "A", " " <= "1", "A " <= "AA", "A b"
< "Ab".
```

♦ Analizînd tabelul caracterelor observăm că acesta conţine şi caracterul " care este folosit pentru a delimita şirurile de caractere. Atunci cînd se doreşte utilizarea caracterului " într-un şir de caractere este obligatorie **dublarea sa**. De exemplu pentru a construi şirul `A"B`, luăm `"A""B"`, iar pentru `A""B`, luăm `"A""""B"`.

În exemplele de mai sus am utilizat în şirurile de caractere cifrele de la 0 la 9. Distingem astfel

numerele: `123, 12, 3`

şi

şirurile de caractere: `"123", "12", "3".`

Să ne reamintim că numerele pot fi adunate, înmulţite, comparate, atribuite, citite, scrise, pe cînd şirurile de caractere pot fi doar atribuite, comparate, concatenate, citite, scrise.

Exemplul 15.5. Programul ilustrează utilizarea variabilelor numerice şi a celor de tip şir 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 şir de caractere şi invers.

Trecerea de la un număr la un şir de caractere se face cu

```
STR$ nr
```

unde `nr` este un număr, iar rezultatul este şirul de caractere corespunzător. De exemplu:

```
STR$ 125 = "125"
STR$ 3 = "3"
STR$ 2+"1" = "21"
"1"+STR$ 45+"+" = "145+"
```

Operaţia inversă, de obţinere a valorii numerice a unui şir de caractere (avînd drept componente caracterele `"0"` la `"9"`) se realizează cu:

```
VAL sir
```

unde `sir` este un şir de caractere. De exemplu:

```
VAL "15" = 15
VAL "15"+1 = 16
1+VAL "125"-2 = 124.
```

O altă informaţie utilă referitoare la un şir de caractere este lungimea acestuia care poate fi aflată prin:

```
LEN sir
```

unde `sir` este un şir de caractere, iar rezultatul este lungimea acestuia. De exemplu:

```
LEN "ABC" = 3
LEN "125AB" = 5
```

Observaţii:

♦ `LEN`, `VAL` şi `STR$` se tastează treçînd în modul **E** (**CS** şi **SS** tastate simultan).

♦ `LEN`, `VAL` şi `STR$` se aplică şi variabilelor, după cum rezultă şi din exemplul următor:

Exemplul 15.6. Programul calculează lungimea unui şir care se citeşte; dacă şirul este format dintr-o cifră atunci, dacă este `"0"` sau `"9"`, se scrie, iar dacă nu, se scrie şirul format din cifra anterioară şi cea care urmează (de exemplu pentru `"3"` se scrie `"24"`). Pentru celelalte cazuri se scrie şirul 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 şir de caractere şi invers se poate face într-un mod mai general decît cel de sus.

Să presupunem că pentru fiecare caracter există o valoare numerică ce-i corespunde şi care are legătură cu poziţia 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 decît cea asociată lui y.

Fiind dat un caracter x, valoarea sa numerică v, este dată prin următoarea operaţie:

```
v = CODE x .
```

De exemplu:

```
CODE "a" = 97
CODE "b" = 98
CODE "A" = 65
CODE " " = 32
```

Se poate face şi o operaţia inversă, de determinare a caracterului x, ce corespunde unei valori numerice v, prin:

```
x = CHR$ v .
```

De exemplu:

```
CHR$ 97 = "a"
CHR$ 98 = "b"
CHR$ 32 = " "
```

Observaţii:

♦ CODE şi CHR$ se tastează trecînd în modul **E** (**CS+SS** tastate simultan).

♦ CODE şi CHR$ se aplică şi variabilelor de tip şir de caractere şi respectiv numerice, după cum rezultă şi din exemplul anterior:

Exemplul 15.7. Următorul program afişează 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
```

☞ Reţineţi !

■ Tablourile sînt variabile structurate avînd un număr specificat de linii şi coloane. Un tablou t cu n linii şi m coloane se introduce prin

```
DIM t(n,m)
```

■ Variabilele structurate cu elemente şiruri de caractere sînt caracterizate de numărul de elemente şi de lungimea maximă a şirului. O variabilă structurată a$ avînd n şiruri de caractere, fiecare de lungime maximă m, se introduce prin

```
DIM a$(n,m)
```

■ Lungimea unui şir de caractere s, se obţine prin

```
LEN s
```

■ Valoarea numerică a unui şir de caractere conţinînd caractere de tip cifre se calculează cu

```
VAL s
```

■ Şirul de caractere corespunzător unui număr n se obţine cu

```
STR$ n
```

STR$ şi VAL sînt funcţii inverse una alteia.

■ Asocierea dintre un caracter x şi o valoare numerică v se face cu

```
v=CODE x  şi  x=CHR$ v.
```

CODE şi CHR$ sînt funcţii inverse una alteia.

Exerciţii

1. Să se scrie un program care să citească un tablou t(3,5) pe linii şi 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) şi să scrie tabloul o(2,2) obţinut din cele două astfel: o(i,j)=t(i,j)-p(i,j), dacă t(i,j)>=p(i,j) şi o(i,j)=0, în caz contrar.

3. Să se scrie un program care citeşte un tablou t(2,3) şi 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: 0 1
   1 2
   2 2
```

4. Să se scrie un program care citeşte un tablou `x(4,5)` şi mută elementele sale într-un vector `v(20)` astfel: se mută întîi prima coloană, apoi a doua etc. Să se afişeze `x` şi `v`.

5. Să se scrie un program care citeşte un tablou `p(5,5)` şi determină cel mai mare element al său şi poziţia sa în tablou. Aceeaşi problemă pentru cel mai mic element din tablou.

6. Să se scrie un program care citeşte un tablou `t(n,m)` cu `n` şi `m` introduse într-o instrucţiune `DATA`, şi determină cîte elemente mai mici ca 0 se află în tabloul `t`.

7. Să se scrie un program care să citească un tablou `t(4,4)` şi să scrie liniile tabloului `t` astfel: prima linie şi a treia de la stînga la dreapta, iar a doua şi a patra în sens invers. Generalizare pentru un tablou `t(2n,k)`, cu `n` şi `k` citite cu `INPUT`.

8. Să se scrie un program care citeşte un tablou `t(m,n)` cu elemente pozitive şi 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 funcţie de `m` şi `n`. De exemplu pentru tabloul `t(3,5)`

```
1 2 5 7 1
2 1 7 2 2
3 1 4 5 3
```

vectorul `v` va conţine 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<>0` să se calculeze `sa/sb`.

10. Să se scrie un program care să afişeze poziţiile elementelor dintr-un tablou (linia şi coloana) care sînt nule. Parcurgerea se va face pe linii.

11. Fiind dat un tablou `t` cu `n` linii şi `n` coloane, elementele `t(1,1)`, `t(2,2)`, . . . , `t(n,n)` definesc **diagonala principală** a tabloului. Să se scrie un program care să afişeze elementele care se află într-un tablou deasupra diagonalei principale. Parcurgerea se va face pe linii. Aceeaşi 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 conţine 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<>0`, atunci să se calculeze şi `(lp+ln)/(cp+cn)`.

13. Să se scrie un program pentru întocmirea unui catalog care conţine numele şi prenumele a 5 elevi precum şi notele la 4 materii.

a) Să se afişeze elevii şi notele acestora.
b) Să se afişeze elevii care au note maxime la cel puţin 2 materii.

14. Să se afle cel mai mare element dintr-o mulţime de şiruri de caractere de lungime maximă 20; mulţimea are 10 elemente.

15. Să se scrie un program care să citească o succesiune de şiruri de caractere şi să afişeze lungimea acestora.

16. Să se scrie un program care citeşte două caractere şi afişează apoi toate caracterele care au valorile numerice asociate cuprinse între valorile numerice ale caracterelor citite. De exemplu dacă se citesc `"a"` şi `"c"`, atunci se afişează `"a" "b" "c"`.

17. Să se scrie un program care citeşte două caractere pentru care diferenţa valorilor numerice asociate este cel mult 5 şi afişează şirul de caractere format din caracterele care au valorile numerice cuprinse între valorile numerice ale celor două caractere citite. De exemplu, dacă se citesc `"1"` şi `"4"` atunci se afişează `"1234"`.

18. Rezolvaţi:

```
LEN "LEN"=?; LEN "1" + 1=?; VAL "125" + 1=?; STR$ 100
+ 1=?; STR$ (100+1)=?; "1" + STR$ 1=?; CHR$ CODE
"a"=?; CODE CHR$ 97=? .
```

16

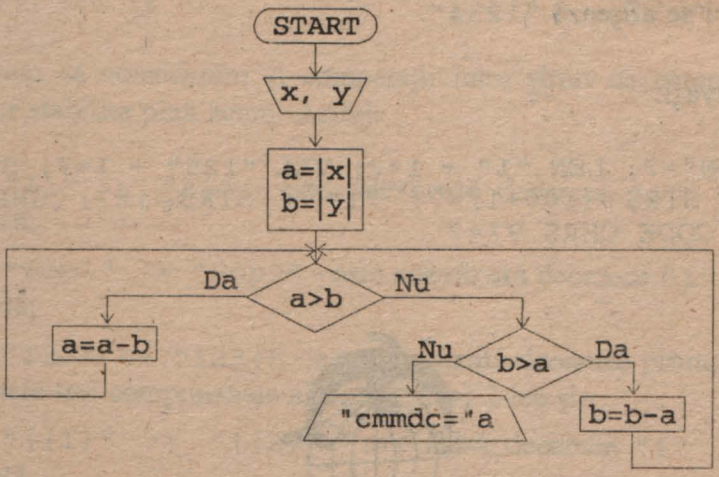
Probleme rezolvate şi propuse

În acest capitol vom rezolva cîteva probleme la care vom recapitula cunoştinţele de BASIC învăţate pînă acum şi 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 si y (pe scurt cmmdc(x,y)) a mai fost discutată şi 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 cmmdc(|x|,|y|), pentru două numere întregi x şi y:



Acest algoritm de o mare simplitate şi eleganţă poate fi "tradus" într-un program (renunţăm la prezentarea argumentelor matematice care justifică faptul că algoritmul conduce la cmmdc(|x|,|y|)). Deoarece vom mai utiliza acest algoritm şi 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. Operaţii cu numere raţionale

Sînt cunoscute din matematică regulile după care se adună, scad, înmulţesc şi împart numerele raţionale. În acest paragraf vom concepe programe pentru operaţiile amintite cu numere raţionale şi de asemenea pentru simplificarea fracţiilor (punerea numerelor raţionale sub forma unei fracţii ireductibile). Obţinerea formei ireductibile a unui număr raţional x/y se realizează prin simplificare cu cmmdc(|x|,|y|).

Următorul program realizează operaţiile de adunare, scădere, înmulţire şi împărţire a două numere raţionale x/y şi xp/yp. Rezultatele vor fi puse sub formă ireductibilă. Programul solicită una din următoarele opţiuni:

a - adunare
s - scădere
m - înmulţire
d - împărţire
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 **exerciţiu** să modificaţi acest program astfel încît să testeze dacă numitorii sînt nenuli (variabilele `y, yp`) şi afişarea să fie efectuată de o subrutină (afişarea de la linia 250).

16.3. Valoarea unui polinom. Schimbarea bazei de numeraţie

Pentru un polinom

$$5 \cdot X^3+2 \cdot X^2-X+1 \qquad (1)$$

a-i calcula valoarea pentru `X=2` înseamnă a înlocui în expresia care defineşte polinomul pe `X` cu `2`, deci:

$$5 \cdot 2^3+2 \cdot 2^2-2+1=47$$

Acelaşi polinom se poate pune sub forma

$$((5 \cdot X+2) \cdot X-1) \cdot X+1 \qquad (2)$$

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).

Calculele care se fac pentru această formă şi pentru X=2 pot fi descrise prin următoarele relaţii:

$$y=5\cdot 2+2 \qquad (2.1)$$
$$z=y\cdot 2-1 \qquad (2.2)$$
$$u=z\cdot 2+1 \qquad (2.3)$$

În general, un polinom se scrie sub forma

$$a_nX^n+a_{n-1}X^{n-1}+\ldots+a_1X+a_0 \qquad (1)$$

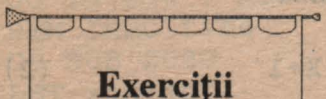
iar forma echivalentă este

$$(\ldots((a_nX+a_{n-1})\cdot X+\ldots+a_1)\cdot X+a_0 \qquad (2)$$

Valoarea acestuia pentru X=v se obţine înlocuind pe X cu v.

Relaţiile după care se calculează valoarea acestui polinom pentru X=v sînt următoarele:

$$y_1=a_n\cdot v+a_{n-1} \qquad (2.1)$$
$$y_2=y_1\cdot v+a_{n-2} \qquad (2.2)$$
$$y_3=y_2\cdot v+a_{n-3} \qquad (2.3)$$
$$\ldots$$
$$y_n=y_{n-1}\cdot v+a_0 \qquad (2.n)$$



Exerciţii

1. Consideraţi polinomul:

$$5X^4-2X^3+X^2-X+1 \qquad (1')$$

şi puneţi-l în forma (2') aşa cum polinomul anterior a fost adus de la forma (1) la forma (2).

2. Scrieţi relaţii similare cu (2.1), (2.2), (2.3), pentru forma (2') a polinomului de mai sus şi pentru X=-1.

Cea mai mare putere a lui X într-un polinom, numită **gradul polinomului**, ne indică numărul de relaţii ce definesc valoarea polinomului adus la forma (2), pentru o anume valoare v a lui X.

Analizînd forma (2) a polinomului şi relaţiile (2.1) - (2.3) ca şi forma şi relaţiile echivalente ale polinomului din exerciţiul de mai sus, putem uşor înţelege 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
```

Coeficienţii polinomului (numerele care înmulţesc pe X^3, X^2, X, X^0) apar în instrucţiunea DATA. Evident schimbarea polinomului implică şi modificarea acestei instrucţiuni.

Programul anterior se poate utiliza şi pentru a trece un număr k scris într-o bază b, în baza 10.

Într-adevăr, se ştie că un număr k avînd cifrele k_n, k_{n-1}, ..., k_1, k_0, în această ordine, se poate scrie astfel:

$$k=k_n \cdot b^n+k_{n-1} \cdot b^{n-1}+\ldots+k_1 \cdot b+k_0 .$$

Este evidentă asemănarea scrierii lui k în baza b, cu determinarea valorii unui polinom, pentru valoarea b. Vă lăsăm ca **exerciţiu** să scrieţi polinomul din care se obţine k pentru X=b.

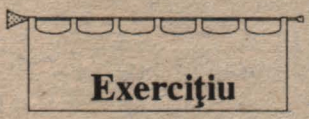
Executînd programul anterior pentru un anume k (număr în baza b), valoarea a, care se afişează reprezintă scrierea lui k în baza 10.

În informatică se foloseşte destul de frecvent scrierea unui număr în baza 2. Din acest motiv, dar şi 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 şi rezultă un cît, c şi un rest, r_i;
c) Dacă c>=2, atunci se face x=c şi i=i+1, apoi se merge la b);
d) În caz contrar (adică c<2) obţinem reprezentarea în baza 2 ca fiind formată din următoarele cifre: $c, r_i, r_{i-1}, \ldots, r_1$.



Aplicaţi acest algoritm pentru x=10.

Pentru a reţine cifrele scrierii lui x în baza 2, avem nevoie de un vector notat r. Pentru a determina dimensiunea acestui vector folosim următoarea

Observaţie: Numărul n de componente ale vectorului r, satisface relaţia:

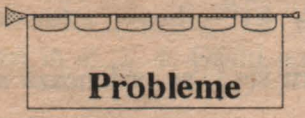
$$2^{n-1} \le x < 2^n.$$

De *exemplu:*

$2^2 \leq 5 < 2^3$, $5_{10}=101_2$; $2^3 \leq 10 < 2^4$, $10_{10}=1010_2$.

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
```



Probleme

1. Să se rezolve exerciţiul din paragraful 16.2.

2. Să se modifice programul de calculare a valorii unui polinom (paragraful 16.3) astfel încît coeficienţii polinomului să fie introduşi cu instrucţiunea 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. Modificaţi programul astfel încît baza în care se trece să fie introdusă prin INPUT.

5. Se dă un şir de numere întregi. Să se aranjeze acest şir astfel încît pe primele poziţii 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 şir de numere întregi. Să se ordoneze şirul astfel încît primele poziţii să conţină numere pare ordonate crescător, iar apoi să conţină 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 şir de numere întregi. Să se aranjeze acest şir astfel încît primele poziţii să conţină 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 şir $a_1, a_2, \ldots, a_n$ şi o valoare k, $1<k\leq n$. Să se ordoneze şirul astfel:

$$a_k\ a_{k+1}\ \ldots\ a_n\ a_1\ a_2\ \ldots\ a_{k-1}.$$

De exemplu, şirul 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 şi 5 coloane. Să se afişeze elementele acestui tablou astfel: întîi linia de sus, apoi coloana din dreapta, urmează linia de jos (de la dreapta la stînga) şi la sfîrşit coloana din stînga (de jos în sus);

tabloul cu 4 linii şi 4 coloane ce nu a fost listat încă este afişat în acelaşi mod; apoi cel de 3 linii şi 3 coloane, pînă se afişează toate elementele tabloului. De exemplu tabloul de 4 linii şi 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 şi `n` coloane.

13. Se dă şirul `1 2 3...n -1 -2...-n`. Să se ordoneze şirul astfel încît să devină `1 -1 2 -2 3 -3...n -n`

14. Se dă şirul `1 2 3...n -1 -2... -n`. Să se ordoneze şirul astfel încît să devină `1 -n 2 -(n-1)...n -1`

15. Se dă un şir de numere întregi aflat într-un vector. Se cere să se obţină şirul numerelor pozitive. De exemplu şirul `-1 2 3 -2 -3 4` se transformă în `2 3 4`. Găsiţi o soluţie care nu foloseşte un alt vector.

17

Limbajul GW-BASIC
Diferenţe faţă 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 faţă **este înţeleasă numai dacă au fost parcurse capitolele precedente.** Limbajul pe care-l prezentăm se numeşte 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 numeşte **prompt.**

■ Pentru a **tasta** un program facem următoarele precizări:

Tastatura este alcătuită din următoarele elemente:

- litere mari şi mici
- cifrele 0 la 9
- simboluri speciale (<, >, =, :, ! etc.)
- tasta **SPACE**, pentru introducerea caracterului spaţiu

Introducerea numelor de variabile, a instrucţiunilor, 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 funcţională **CR** sau **ENTER** sau **RETURN**).

- taste funcţionale:

CR (sau **ENTER** sau **RETURN**)

F1, F2, ... , F10 (eventual **F11, F12**)

SHIFT (cînd este ţinută apăsată transmite litere mici şi caracterele scrise în partea de sus a tastelor)

INS, DEL

CTRL (sau CONTROL)

BREAK

ALT

săgeţile (→,←,↑,↓)

Introducerea unui program se poate face în două moduri:

manual: prin tastarea element cu element aşa 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 opţionale, prin lipsă avînd valorile 10. După ce se termină introducerea programului se tastează **CTRL** şi **C** sau **CTRL** şi **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 acţionarea săgeţilor în "sus" sau în "jos" pînă se ajunge pe linia vizată

- fie prin tastarea comenzii

```
EDIT n
```

unde n este numărul liniei vizate.

• **determinarea poziţiei din linie** în care se face modificarea se obţine prin tastarea săgeţilor "stînga" şi "dreapta".

• **inserarea** unor caractere la poziţia curentă se face astfel: se tastează **INS** şi apoi se introduc caracterele vizate. De exemplu în linia

```
10 LET A=1
```

dorim să inserăm după A, caracterele B şi C; ajunşi în poziţia =, se tastează:

```
INS
B C CR
```

• **ştergerea** unor caractere se face prin apăsarea tastei **DEL**, care la fiecare apăsare şterge caracterul de la poziţia curentă. După terminarea ştergerii se tastează **CR**.

Important! Orice modificare (inserare, ştergere) se termină cu apăsarea tastei **CR**.

■ Principalele **comenzi privind dinamica programelor**:

F1 - **listarea** programului

F2 - **execuţia** 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

```
GWBASIC
```

- Introducerea unui program, în modul automat, se realizează tastînd întîi, de exemplu:

```
AUTO 100,10
```

şi 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 obţine 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 operaţiei `+` din linia `120`, cu `*` şi inserarea unei linii noi care să afişeze `(A-B)*C` se face cu :

```
EDIT 100
```

urmează apoi deplasare până la sfîrşitul liniei şi tastarea următoarelor caractere

```
C CR
```

obţinîndu-se

```
100 INPUT A,B,C
```

În mod asemănător se obţine şi 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** şi apoi **INS**; urmînd a se tasta:

```
*
```

Introducerea unei linii care să afişeze `(A-B)*C` se obţine tastînd, de exemplu:

```
125 PRINT (A-B)*C
```

Dacă dorim să ştergem, de exemplu, linia 120, atunci se tastează

```
120 CR
```

şi linia va fi distrusă.

17.2. Etichete, constante, variabile, expresii

Etichetele, numerele ce preced instrucţiunile, au valori cuprinse între 0 şi 65529.

Pe aceeaşi linie pot fi puse mai multe instrucţiuni despărţite cu : .

Constantele sînt de două tipuri: **numerice** şi **şiruri de caractere.**

Exemple de constante numerice:

```
25, -16, 128
```

care sînt **întregi** (domeniul acestor fiind între -32768 şi 32767) şi

```
17.1, -18.02, 18.53E-7, -47.033E+4
```

care sînt **reale** (domeniul acestora fiind cuprins între $3*10^{-39}$ şi $1.7*10^{38}$)

Menţionăm că limbajul GW-BASIC conţine şi constante numerice **hexazecimale,** cu prefixul **&H** şi **octale,** cu prefixul **&** sau **&O.** De exemplu:

```
&H10, &H0F, &12, &O023
```

care reprezintă constantele zecimale 16, 15, 10 şi respectiv 19.

Constantele de tip **şir de caractere** sînt şiruri 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 **şir de caractere**, ultimele terminîndu-se cu semnul $.

Spre deosebire de BASIC, în GW-BASIC un nume de variabilă poate să conţină şi simbolul ? şi nu poate fi un nume ce identifică o instrucţiune sau comandă.

Exemple:

```
ION, I, ALFA
```

sînt variabile numerice, iar

```
A$, POP$, DAN$
```

sînt variabile de tip şir de caractere.

Variabilele **structurate** şi 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 instrucţiune `DIM` pot să apară mai multe variabile.

Variabilele numerice în GW-BASIC pot fi clasificate în variabile **întregi** (cu sufixul %) şi 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 (conţin valori numerice reale).

Expresiile sînt alcătuite din constante, variabile sau constante/variabile legate prin diferite operaţii.

Operaţiile se divid în următoarele grupe:

- aritmetice
- relaţionale
- logice

Operaţiile aritmetice sînt: ridicare la putere (^), înmulţirea (*), împărţirea (/), adunarea (+), scăderea (-), împărţirea întreagă (\), restul împărţirii întregi (mod).

Operaţiile relaţionale sînt: egalitatea (=), mai mic (<), mai mare (>), mai mic sau egal (<=), mai mare sau egal (>=), neegal (<>).

Operaţiile logice sînt: NOT, AND, OR, XOR.

Operaţiile permise pentru şiruri de caractere sînt: **atribuirea, comparaţiile** (=, <, >, <=, >=, <>) şi **concatenarea** (+).

Menţionăm 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$(x)`	converteşte valoarea întreagă x, în caracterul corespondent;
`LEN(x$)`	lungimea şirului de caractere conţinut în x$;
`STR$(x)`	şirul de caractere corespunzător valorii numerice aflate în x;
`VAL(x$)`	valoarea numerică a şirului de caractere format din cifre, aflat în variabila x$;
`SPACE$(x)`	şirul de spaţii (blancuri) de lungime egală cu valoarea numerică din variabila x;
`ASC(x$)`	valoarea numerică a primului caracter din şirul x$.

Exemple:

5^2=25; 5*2=10; 5/2=2.5; 5+2=7; 5-2=3; 5\2=2 (împărţire întreagă)
5 mod 2=1 (restul împărţirii 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 condiţie falsă şi 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)="   "

Şirurile de caractere pot fi asociate cu variabile structurate. În cazul limbajului GW-BASIC precizăm că **orice şir de caractere are maximum 255 caractere**, chiar dacă sînt folosite de variabile structurate. Deci orice variabilă structurată de tip şir 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 Instrucţiuni

Instrucţiunea de **scriere** (afişare) este

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

Expresiile `ei` pot fi despărţite de , şi atunci sînt scrise la începutul unei noi zone (o zonă are 14 poziţii) sau de ; şi atunci sînt scrise unele după altele.

Pentru a scrie la o anumită poziţie pe ecran se utilizează instrucţiunea

```
LOCATE r,c
```

unde `r` desemnează valoarea unei linii (cu valori cuprinse între 1 şi 25), iar `c` indică valoarea unei coloane (valorile sînt între 1 şi 40 sau 1 şi 80, în funcţie de ecran.

Exemplu: Pentru a scrie şirul `ABC` în linia 5 începînd cu coloana 10 se utilizează

```
10 LOCATE 5,10
20 PRINT "ABC"
```

Instrucţiunea de **citire** este

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

avînd aceeaşi formă ca şi cea din BASIC. Faţă de aceasta prezintă caracteristica citirii datelor de pe linia curentă. (La PC-uri ecranul nu conţine linii rezervate pentru comenzi, mesaje şi pentru citirea datelor).

Instrucţiunea de **oprire** a execuţiei unui program este

```
STOP .
```

Instrucţiunea de **atribuire** are forma

```
LET v=E
```

cu v şi E ca în instrucţiunea corespunzătoare din BASIC, cu precizarea că în GW-BASIC LET este opţional.

De *exemplu*,

```
LET v=2+v

v=4+v
```

sînt, ambele, instrucţiuni de atribuire corecte în GW-BASIC.

Instrucţiunea **comentariu** este introdusă prin cuvîntul

```
REM .
```

Instrucţiunea GO TO are aceeaşi formă ca în BASIC.

Instrucţiunea IF are, fie forma cunoscută în BASIC

```
IF exp_log THEN S1:S2:...:Sn          (1)
```

cu semnificaţiile cunoscute, fie forma

```
IF exp_log THEN S1:S2:...:Sn ELSE T1:T2:...:Tm  (2)
```

Ultima formă a instrucţiunii se execută astfel: dacă exp_log are valoarea adevărat atunci se execută instrucţiunile S1, S2, ..., Sn, iar în caz contrar se vor executa instrucţiunile T1, T2, ..., 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 conţinute în variabilele I, J.

Instrucţiunea FOR...NEXT are aceeaşi formă ca în BASIC.

Instrucţiunile READ, DATA, RESTORE, GO SUB şi RETURN au în GW-BASIC aceleaşi forme şi semnificaţii ca în BASIC.

17.4. Prelucrare sunet

Pentru a semnala 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 foloseşte

```
SOUND f,d
```

unde f este frecvenţa, iar d indică durata în **unităţi**. De menţionat că 18.2 unităţi echivalează cu o secundă.

Dăm mai jos frecvenţele 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 ţină cont de exprimarea parametrului d în secunde.

Realizarea pauzei între note se obţine utilizînd:

```
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 şi citi, fiind în starea **text**, fie pentru a desena, şi atunci este în starea **grafică**. Există mai multe modele grafice (CGA, EGA, VGA etc.) care se deosebesc prin rezoluţie, 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 împărţit în 640x200 puncte. Pentru **revenirea în modul text** se utilizează comanda

```
SCREEN 0 .
```

Ştergerea ecranului se face cu comanda `CLS` .

În modul grafic se pot efectua următoarele operaţii: desenări de puncte, trasări de drepte şi 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 opţională.

Trasarea unei linii se face cu:

```
LINE (x1,y1) - (x2,y2)
```

unde `(x1,y1)` şi `(x2,y2)` reprezintă coordonatele extremităţilor 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 instrucţiunilor

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 = curăţă ecranul) = ştergere (iniţializare) 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) = instrucţiunea de ciclare

GO SUB (cap.12,p.108;cap.17,p.180) (de la go subroutine = mergi în subprogram) = = instrucţiunea apel de procedură

GO TO (cap.5,p.40;cap.17,p.179) (mergi la) = instrucţiunea de salt

IF (cap.5,p.39;cap.17,p.179) (dacă) = instrucţiunea condiţională

INPUT (cap.3,p.20;cap.17,p.178) (intrare) = instrucţiunea de citire a datelor

LET (cap.4,p.29;cap.17,p.179) (fie) = instrucţiunea de atribuire

LINE (cap.17,p.182) (linie) = trasare linie (GW-BASIC)

LOCATE (cap.17,p.178) (a localiza) = poziţionarea cursorului pentru scriere pe ecran (GWBASIC)

PAUSE (cap.13,p.119) (pauză) = defineşte o pauză (BASIC)

PLOT (cap.14,p.123) (pune) = "pune" un punct (BASIC)

PRINT (cap.3,p.20;cap.17,p.178) (afişare) = instrucţiunea 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) (citeşte) = instrucţiunea de iniţializare

REM (cap.4,p.31;cap.17,p.179) (de la remarks = comentarii) = instrucţiunea comentariu

RESTORE (cap.8,p.71;cap.17,p.180) (reluare) = instrucţiunea de reluare a listei de date

RETURN (cap.12,p.108;cap.17,p.180) (întoarce) = instrucţiunea 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) = instrucţiunea de oprire a programului

ANEXA 2

Lista comenzilor

BREAK (cap.2,p.17) (întrerupere) = întreruperea execuţiei 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) (alergă, aici cu sensul de execută) = executarea unui program

SAVE (cap.2,p.17;cap.17,p.171) (salvare) = salvarea unui program

ANEXA 3

Lista operaţiilor

+, -, *, /, ^ (cap.4,p.28;cap.17,p.176) = operaţiile de adunare, scădere, înmulţire, împărţire şi ridicare la putere

\ (cap.17,p.176) = împărţire întreagă (GW-BASIC)

<, >, =, <=, >=, <> (cap.5,p.35;cap.17,p.176) = operaţii relaţionale

NOT, AND, OR (cap.5,p.36;cap.17,p.176) (nu, şi, sau) = operaţii logice

XOR (cap.5,p.38;cap.17,p.176) = operaţie 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 tabelei de conversie ASCII) = valoarea numerică a primului caracter dintr-un şir (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 şir (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 = lungime) = determină lungimea unui şir

MOD (cap.17,p.176) (de la modulus = modul) = determină restul împărţirii întregi (GW-BASIC)

RND (cap.3,p.23) (de la random = aliator) = furnizează un număr aleator între 0 şi 1

SPACE$ (cap.17,p.176) (spaţiu) = determină un număr întreg de spaţii (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 = şir) = şirul de caractere corespunzător unui număr

VAL (cap.15,p.150;cap.17,p.176) (de la value = valoare) = valoarea numerică a unui şir de caractere

ANEXA 4

Lista exemplelor de programe

BIBLIOGRAFIE

1. I. Diamandi: *Partenerul meu de joc calculatorul*, Recoop, Bucureşti, 1989.

2. I. Diamandi: *Cum să realizăm jocuri pe calculator*, Editura Agni, Bucureşti 1992.

3. I. Diamandi, Gh. Păun: *40 de jocuri logice în BASIC*, Editura Agni, Bucureşti 1992.

4. H. Georgescu, T. Bălănescu, V. Ştefănescu: *Matematică, manual de algebră pentru clasa a IX-a*, cap.VII, Editura Didactică şi Pedagogică Bucureşti, 1989.

5. H. Georgescu, T. Bălănescu, V. Ştefănescu: *Matematică, manual de algebră pentru clasa a X-a*, cap.V, Editura Didactică şi Pedagogică Bucureşti, 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, Bucureşti 1992.

8. S. Vickers: *ZX-SPECTRUM BASIC programming*, Sinclair Research, 1982.

9. *** Colecţia *Gazetei de informatică* pe anii 1991, 1992.

10. *** *GW-BASIC Interpreter, User's Reference.*

11. *** *GW-BASIC Interpreter, User's Guide.*

Cine eşti tu, BASIC?

este o introducere pentru începători în limbajul de programare BASIC. Poate fi folosit atît pentru calculatoarele compatibile Sinclair Spectrum, cît şi 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 Bucureşti. Predă cursuri de limbaje de programare: BASIC, Pascal şi C. Se ocupă de pregătirea lotului pentru olimpiadele internaţionale de informatică.

ISBN 973 - 95626 -1-2

Preţ: 1250 Lei