

*Biblioteca  
de  
Informatică*

**Ion DIAMANDI**

# **CALCULATORUL, coleg de bancă**



**Editura Agni**



Ion Diamandi

**Calculatorul,  
coleg de bancă**

Editura *Agni*  
București, 1993

Tehnoredactarea pe calculator: Ioana Varga, Florin Ilia  
Coperta: Iolanda Malamen

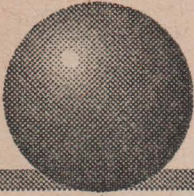
Toate drepturile rezervate Editurii Agni  
Copyright ©1993 Editura Agni  
C.P. 30-107, București

ISBN 973-95626-3-9

Ion Diamandi

**Calculatorul, coleg de bancă**



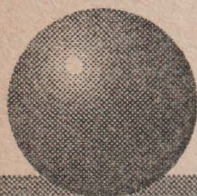


# Cuprins

☞ Cuvînt înainte . . . . .	7
☞ CAP. 1. Logica mașinilor inteligente . . . . .	10
☞ CAP. 2. Calculatorul de sticlă . . . . .	14
☞ CAP. 3. Rezolvarea problemelor cu calculatorul . . . . .	25
☞ Algoritmi . . . . .	25
☞ Rezolvarea problemei . . . . .	28
☞ CAP. 4. Probleme cu numere . . . . .	32
☞ Multiplii numerelor . . . . .	32
☞ Cel mai mic multiplu comun . . . . .	35
☞ Sume și produse de numere . . . . .	36
☞ Dacă șahul Persiei ar fi avut calculator... . . . .	44
☞ O problemă de intuiție . . . . .	46
☞ Divizibilitatea numerelor . . . . .	48
☞ Numere prime . . . . .	50
☞ Ciurul lui Eratostene . . . . .	53
☞ Cel mai mare divizor comun . . . . .	55
☞ Radicali . . . . .	59
☞ Rapid și precis - câteva probleme pentru dumneavoastră . . . . .	62
☞ O problemă privind grafica în mișcare . . . . .	65
☞ Să regăsim "proporția de aur" . . . . .	69
☞ Operații cu mulțimi de numere . . . . .	74
☞ CAP. 5. Din ce este format un calculator ? . . . . .	79

■	Octetul . . . . .	82
■	Să ne jucăm cu memoria . . . . .	84
■	Echipamente periferice . . . . .	86
■	<b>CAP. 6. Calculatoarele în competiție . . . . .</b>	<b>94</b>
■	Viteza calculatoarelor . . . . .	96
■	Ce n-a putut rezolva calculatorul . . . . .	98
■	Verificați-vă cunoștințele despre calculatoare . . . . .	99
■	<b>CAP. 7. Mecanismul hazardului . . . . .</b>	<b>106</b>
■	Un joc cu numere . . . . .	107
■	Să-i ajutăm pe cei mici . . . . .	108
■	Câteva probleme cu numere și grafică . . . . .	110
■	<b>CAP. 8. Probleme cu căutări și inversiuni . . . . .</b>	<b>114</b>
■	<b>CAP. 9. Probleme cu cuvinte . . . . .</b>	<b>119</b>
■	<b>CAP. 10. Funcții . . . . .</b>	<b>126</b>
■	7 probleme pentru funcția de gradul I . . . . .	126
■	Reprezentări grafice de funcții . . . . .	138
■	Reprezentarea grafică a unei funcții de o variabilă . . . . .	146
■	Limite . . . . .	150
■	<b>ANEXA A. Răspunsuri la problemele din capitolul 6 . . . . .</b>	<b>153</b>
■	<b>ANEXA B. Lista programelor prezentate în carte. Mic caiet de programe pentru vacanță . . . . .</b>	<b>154</b>
■	<b>ANEXA C. Lista figurilor prezentate în carte . . . . .</b>	<b>161</b>
■	<b>ANEXA D. Glosarul cuvintelor cheie . . . . .</b>	<b>164</b>
■	<b>Indexul cuvintelor cheie . . . . .</b>	<b>171</b>
■	<b>Bibliografie . . . . .</b>	<b>173</b>





## Cuvînt înainte

Introducerea recentă a informaticii în programa școlară pentru gimnaziu și liceu face necesară studierea unor probleme precum și elaborarea unor metodologii de utilizare a tehnicilor informatice în orele de școală. În acest sens sînt chemați să-și aducă contribuția atît profesori cît și informaticieni care au experiență în domeniul utilizării calculatoarelor de către tînăra generație. Nu mai puțin importante sînt și aspectele psihosociale legate de aceste abordări.

Este cunoscut faptul că în învățămîntul din țara noastră predomină aspectul informativ în defavoarea celui formativ. Introducerea informaticii ca materie școlară și, mai mult, introducerea tehnologiilor informatice în mediul școlar trebuie să aibă ca efect, în primul rînd, tocmai aspectul formativ și ar fi o greșeală ca acest lucru să fie considerat numai sub aspectul asimilării de către elev a unui plus de informații (termeni noi, instrucțiuni, comenzi). Lucrarea de față își propune să reprezinte un ghid care să ofere cîteva idei profesorilor din învățămîntul preuniversitar în vederea organizării unor ore de clasă avînd și calculatorul ca mijloc didactic. Scopul este deprinderea copiilor cu o gîndire ordonată și logică sau, cu alte cuvinte, formarea unei gîndiri algoritmice. Bineînțeles că, în aceeași măsură, lucrarea poate fi utilă elevilor din cursul gimnazial și liceal.

Pentru cel care dorește să folosească lucrarea (pentru a învăța utilizarea calculatorului, sau pentru a învăța la matematică cu ajutorul calculatorului, sau în vederea organizării unor ore de școală cu calculatorul), se presupune cunoașterea prealabilă a cîtorva elemente privind utilizarea calculatoarelor și, anume, modul de operare (de introducere a instrucțiunilor și comenzilor) precum și cîteva dintre cele mai cunoscute instrucțiuni și comenzi ale limbajului BASIC. În acest scop nu se recomandă introducerea clasică utilizată în informatică prin care se descrie forma unei instrucțiuni și apoi se dau exemple de folosire, fiind preferabil să se pornească de la rezolvarea unor probleme concrete de școală corespunzătoare anului de studiu respectiv. Este de asemenea indicat un ritm lent de introducere a termenilor și instrucțiunilor noi. Deși accentul se va pune pe

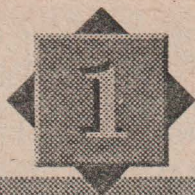
conceperea programelor (aspectul formativ), aceasta nu va reprezenta un scop în sine, ci, un mijloc de rezolvare a problemelor curente de școală. De aceea s-a ales ca temă principală rezolvarea unor probleme de matematică, dar lucrarea se dorește a constitui un stimulent pentru profesori, în ideea de a descoperi, ei înșiși, și de a integra în orele de clasă noi probleme (și nu numai de matematică) a căror rezolvare se pretează a se realiza cu ajutorul calculatorului. De fapt astfel de rezolvări de probleme (fiind strâns legate de programa școlară și nesolicitând cunoștințe suplimentare de matematică față de cele prevăzute acolo), se pot integra și în orele de la alte discipline școlare.

Modul de tratare a problemelor este următorul: se prezintă problema eventual cu încercarea de rezolvare pe cale clasică și explicarea necesității lucrului cu calculatorul. Necesitatea utilizării calculatorului în rezolvarea problemei poate proveni și din posibilitatea astfel creată de explorare și experimentare pe care o oferă acest nou instrument. Apoi se prezintă metoda (algoritmul) de rezolvare a problemei. S-a considerat că pentru acest tip de probleme nu este necesară și descrierea algoritmului prin tehnici ca scheme logice sau diagrame de structură, fiind suficientă înțelegerea pașilor algoritmului. În final se expune un exemplu de program scris în limbajul BASIC. Astfel sînt abordate probleme cunoscute din programa de matematică pentru gimnaziu și liceu: multipli numerelor, divizibilitatea numerelor, cel mai mare divizor comun și cel mai mic multiplu comun, numere prime, descompunerea în factori primi, șirul lui Fibonacci, funcția de gradul 1, grafice de funcții, limite de șiruri etc. Deseori sînt propuse spre rezolvare și anumite probleme distractive sau probleme pentru realizarea unor jocuri.

Orele de informatică se vor desfășura sub forma activităților practice cu calculatorul, prin problemele propuse urmărindu-se nu numai dezvoltarea unei gândiri logice și convergente (algoritmice) ci și formarea unor deprinderi de explorare și experimentare cu calculatorul. În unele exemple am căutat și dezvoltarea unor aptitudini pentru luarea unor decizii în condiții restrictive de timp (gîndire intuitivă), foarte necesare în rezolvarea ulterioară a unor probleme din viața de zi cu zi.

Paralel cu activitățile propuse am introdus și cîteva elemente fundamentale privind bazele proiectării și funcționării calculatoarelor. Informatica a ajuns la un stadiu de maturitate, care impune ca unele din rezultatele sale să intre în cultura generală a elevilor.

Programele exemplificate sînt funcționale pentru cele mai răspîndite calculatoare din țara noastră și, implicit, din sistemul de învățămînt preuniversitar și anume, calculatoarele compatibile Sinclair Spectrum: HC, CIP, JET, COBRA, TIM. S-a căutat de asemenea ca programele să fie realizate într-un BASIC standard pentru a putea fi lesne transpuse și pe alte calculatoare. Majoritatea programelor sînt funcționale și pe calculatoare de tip PC în versiunea GWBASIC. Cele care utilizează tehnici specifice versiunii SINCLAIR BASIC (în special cele care utilizează elemente grafice și șiruri de caractere), fiind, evident, nefuncționale pe calculatoare PC, sînt prezentate în două variante (A și B), pentru SINCLAIR SPECTRUM și, respectiv, GWBASIC. Pentru a veni în sprijinul celor care vor să se inițieze în utilizarea calculatoarelor, în final, se prezintă un index al cuvintelor cheie folosite în lucrare.



# Logica mașinilor inteligente

Pentru a obține rezultatul unor calcule, de exemplu  $3+4$ , știm că nu este suficient să cerem calculatorului  $3+4$  și nici  $3+4=$ . Este necesar să comandăm calculatorului să ne afișeze rezultatul operației respective, iar acest lucru se va realiza cu comanda PRINT (AFIȘEAZĂ). Pentru tipul de calculator avut în vedere PRINT se va obține prin acționarea tastei P. Deci PRINT  $3+4$  și apoi tasta **CR (ENTER)**. Vom obține afișat pe ecran rezultatul 7. În mod similar pentru calculul oricărei expresii se va folosi aceeași comandă. Încercați de exemplu:

```
PRINT 3.47 * (0.34 - 41.83/(SQRT 70 - 3)*2)
```

De asemenea pentru a obține afișarea unui text pe ecran se va folosi PRINT iar textul va fi încadrat între ghilimele.

```
PRINT "Am trecut in clasa a 8-a"  
PRINT " 2 + 3 = 4 "
```

*Dar ce se va întâmpla dacă vom introduce comanda PRINT  $2 = 2$  ?*

Evident, nu i-am cerut calculatorului nici să ne efectueze un calcul și să ne afișeze rezultatul și nici să ne afișeze un text ca  $2=2$ , deoarece nu l-am încadrat între ghilimele. Sigur, ați observat însă că la introducerea comenzii PRINT  $2 = 2$  pe ecran s-a afișat rezultatul 1, în cazul în care am lucrat cu HC și -1 în cazul PC-ului. Același rezultat îl vom obține și la comanda:

```
PRINT 2 + 4 = 6
```

În schimb pentru PRINT  $1 = 2$  vom obține rezultatul 0, la fel ca și pentru PRINT  $2 - 1 = 6$ . Acum lucrurile încep să se clarifice.

$2 = 2$  sau  $2 + 4 = 6$  reprezintă *propoziții adevărate*, iar acestea au pentru calculatorul HC valoarea 1 (Adevărat = 1, respectiv -1 pentru PC) în timp

ce  $1 = 2$  sau  $2 - 1 = 6$  reprezintă propoziții false care au pentru calculator valoarea 0 (Fals = 0).

*Dar ce rezultat vom obține cu PRINT  $1 = 1 = 1$ ?*

Observăm că obținem valoarea 1, ceea ce ne așteptam, deoarece am scris numai lucruri adevărate. În schimb cu PRINT  $4 = 4 = 2 + 2$  obținem valoarea 0.

Pentru a înțelege rezultatul obținut trebuie să știm cum *evaluatează* calculatorul ce urmează după comanda PRINT.

$4 = 4$  este o poziție adevărată pe care calculatorul o evaluează cu 1, apoi evaluează expresia  $2 + 2$  la 4 și, în sfârșit, evaluează propoziția  $1 = 4$  care, fiind falsă, va afișa rezultatul final 0.

De notat că mai întâi se evaluează expresiile aritmetice și apoi cele logice.

De exemplu pentru PRINT  $3+2 < 3*2-4$ , evaluează mai întâi  $3+2$  cu 5 apoi  $3*2-4$  cu 2 iar apoi comparându-l pe 5 cu 2 rezultă o propoziție falsă, afișând deci 0.

*Puteți explica de ce cu comanda PRINT  $1 = 1 = 3$  obținem rezultatul 0, iar cu PRINT  $1 = 1 > 3 < 2$  obținem valoarea 1?*

Cu comanda PRINT  $1 = 1 = 3$  obținem rezultatul 0 deoarece calculatorul evaluează  $1 = 1$  cu 1 fiind o propoziție adevărată, iar 1 obținut nefiind egal cu 3 (propoziție falsă) obținem rezultatul final 0.

Cu comanda PRINT  $1 = 1 > 3 < 2$  obținem rezultatul 1 deoarece:

$1 = 1$  propoziție adevărată  $\rightarrow$  valoarea 1

$1 > 3$  propoziție falsă  $\rightarrow$  valoarea 0

$0 < 2$  propoziție adevărată  $\rightarrow$  valoarea 1

În expresiile și propozițiile care vor fi evaluate după comanda PRINT se pot introduce nu numai numere, semne pentru operațiile aritmetice, funcții (de exemplu SQR = radical, SIN = sinus, etc.) și paranteze (rotunde), dar și semnele  $<$ ,  $>$ ,  $=<$ ,  $>=$  precum și operatorii logici OR (SAU) și AND (ȘI).

Să vedem cum putem verifica cu ajutorul calculatorului operațiile cu SAU care sînt date în următoarea tabelă de adevăr (Adevărat = A, Fals = F, SAU =  $\vee$ ):

a) $A \vee A = A$	sau	$1 \text{ OR } 1 = 1$
b) $A \vee F = A$	sau	$1 \text{ OR } 0 = 1$
c) $F \vee A = F$	sau	$0 \text{ OR } 1 = 1$
d) $F \vee F = F$	sau	$0 \text{ OR } 0 = 0$

	Verificare cu calculatorul	Rezultat
a)	PRINT 1 = 1 OR 2 = 1*2	1
b)	PRINT 1 = 1 OR 1 = 2	1
c)	PRINT 1 = 2 OR 1 = 1	1
d)	PRINT 1 = 2 OR 2 = 3	0

În aceste exerciții de logică se poate folosi și operatorul logic NOT care are funcția de a *nega* o propoziție. De exemplu, PRINT NOT (1 > 2) va da rezultatul 1 deoarece 1 > 2 este o propoziție falsă (rezultat 0), iar negarea ei va fi o propoziție adevărată (rezultat 1).

Putem verifica faptul că printr-o negare AND se transformă în OR iar OR în AND. Astfel cu PRINT NOT (1=2 AND 1=1) obținem ca rezultat 1 deoarece propoziția din paranteză este falsă, iar negată va fi adevărată; forma va fi echivalentă cu PRINT NOT 1=2 OR NOT 1=1 care dă ca rezultat final tot 1.

În mod similar verificării operațiilor cu SAU se pot verifica și operațiile cu ȘI (AND =  $\wedge$ ):

e) $A \wedge A = A$	sau	$1 \text{ AND } 1 = 1$
f) $A \wedge F = F$	sau	$1 \text{ AND } 0 = 0$
g) $F \wedge A = F$	sau	$0 \text{ AND } 1 = 0$
h) $F \wedge F = F$	sau	$0 \text{ AND } 0 = 0$

	Verificare cu calculatorul	Rezultat
e)	PRINT 1 = 1 AND 2 = 2	1
f)	PRINT 1 = 1 AND 1 = 2	0
g)	PRINT 1 = 2 AND 1 = 1	0
h)	PRINT 1 = 2 AND 2 = 3	0

Putem realiza și propoziții în care să combinăm cei doi operatori OR și AND.

De exemplu putem verifica următoarea relație care demonstrează distributivitatea lui AND față de OR :

$$(P1 \vee P2) \wedge P3 = (P1 \wedge P3) \vee (P2 \wedge P3)$$

De exemplu să considerăm că P1 este adevărată iar P2 și P3 sînt false. În acest caz  $(P1 \vee P2)$  va fi adevărată iar  $(P1 \vee P2) \wedge P3$  va fi falsă. În consecință rezultatul va fi o propoziție falsă (0).

Verificare cu calculatorul:

$$\text{PRINT } (1 = 1 \text{ OR } 1 = 2) \text{ AND } 1 = 3$$

Obținem într-adevăr rezultatul 0, la fel ca și în cazul:

$$\text{PRINT } (1=1 \text{ AND } 1=3) \text{ OR } (1=2 \text{ AND } 1=3)$$



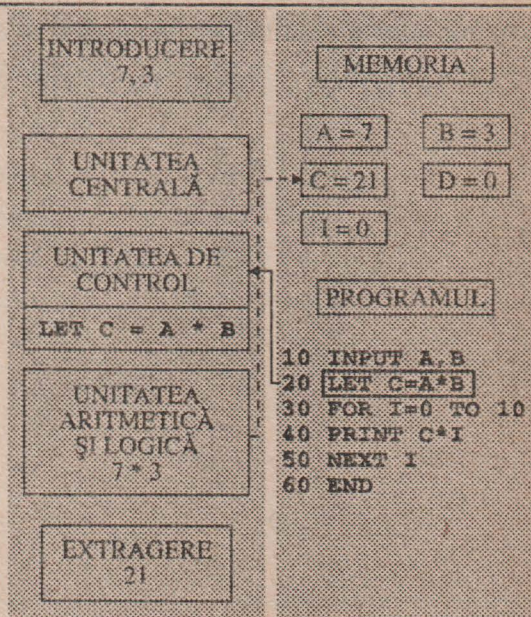
## Calculatorul de sticlă

Știind cum funcționează un calculator ne va fi mult mai ușor să realizăm programe și, de asemenea, să le îndreptăm (depanăm) astfel încât să obținem rezultate corecte.

Pentru a înțelege cum funcționează un calculator ne-ar fi foarte util dacă am putea, oricând am dori, să îl "secționăm" sau, cu alte cuvinte, dacă am avea la dispoziție un "calculator de sticlă" și să vedem ce se petrece în interiorul său. Astfel am vedea, pe de o parte, părțile componente ale calculatorului (unitatea centrală, unitatea aritmetică logică, unitatea de introducere a datelor, unitatea de extragere a rezultatelor, etc.), iar, pe de altă parte, memoria calculatorului care conține atât programul care se execută cât și locațiile (celulele) de memorie care conțin valorile variabilelor (parametrilor) din program.

fig.1

Calculatorul de sticlă





În figura 1 se distinge ce am "vedea" în "calculatorul de sticlă" sau în cazul "secționării" unui calculator într-un anumit moment al funcționării programului care se află în memorie.

O instrucțiune care se execută va fi preluată de unitatea centrală (și anume de unitatea de control) care o va decodifica și o va transmite la o altă unitate în funcție de rezultatul decodificării. Astfel dacă este vorba de o instrucțiune de introducere de date, atunci va fi implicată unitatea de introducere.

Prin unitate de introducere înțelegem de cele mai multe ori tastatura, dar poate fi și unitatea de discuri (sau casetofonul).

Astfel considerînd că în memorie s-a introdus un program, după ce s-a lansat comanda de execuție a programului, RUN, calculatorul va lua prima linie de program (10 INPUT A,B) iar instrucțiunea respectivă o va trimite unității de control. Aceasta o decodifică și, observînd că este vorba de o instrucțiune de introducere, (alte instrucțiuni de acest fel sînt READ, LOAD, etc.) va determina așteptarea tastării de către utilizator a unor valori numerice. Să presupunem că s-au introdus valorile 7 și 3. În acest caz aceste valori se vor depune în locațiile de memorie A și respectiv B după care controlul va fi trecut la următoarea linie de program (20 LET C=A\*B). Acest moment este oglindit chiar în figură. După ce instrucțiunea este decodificată, unitatea aritmetică și logică (UAL) va evalua expresia  $3 \times 7$ . Valoarea rezultată va fi depusă în locația de memorie C după care controlul va trece la următoarea linie de program. Dacă aceasta ar fi linia 25 PRINT C, fiind implicată o instrucțiune de extragere de rezultate, aceasta va avea efect asupra televizorului sau monitorului calculatorului afișîndu-se pe ecran numărul 21.

Instrucțiunile care activează unitatea aritmetică și logică sînt acelea care conțin *expresii care necesită a fi evaluate* (calculate) (LET, PRINT, GO TO, IF etc) sau care necesită comparații aritmetice sau logice (IF), în timp ce, instrucțiunile care acționează asupra unităților de extragere sînt: PRINT, PRINT AT, PLOT, DRAW, CIRCLE, LPRINT, LIST, LLIST, SAVE etc. De asemenea instrucțiunile care au efect asupra memoriei calculatorului (de fapt asupra locațiilor de memorie) sînt așa numitele instrucțiuni de atribuire, și anume, LET, READ-DATA, INPUT, FOR-NEXT, executarea lor avînd ca urmare depunerea unor valori în locații de memorie. Observăm că multe din instrucțiuni activează mai multe unități. De

exemplu LET activează unitatea aritmetică logică pentru a calcula o expresie iar apoi depune într-o locație de memorie valoarea rezultată.

Din păcate calculatoarele cu care lucrăm nu sînt "calculatoare de sticlă" iar dacă secționăm calculatorul nu vom vedea decît circuite electronice. Astfel încît nu ne rămîne decît să ne imaginăm ce se întîmplă într-un calculator atunci cînd se execută un program. Să încercăm acest lucru cu un program care realizează adunarea a **N** numere naturale:

```

10 INPUT N
20 LET S = 0
30 FOR I = 1 TO N
40 LET S = S + I
50 NEXT I
60 PRINT S

```

Pașii de program cu instrucțiunile care se execută, precum și acțiunile corespunzătoare efectuate de calculator sînt ogindite în tabelul 1.

Momentul (pași de program)	Instrucțiunea care se execută	Ce face calculatorul	Cum arată memoria după acțiunea calculatorului
1.	10 INPUT N	Calculatorul așteaptă introducerea unui număr de la tastatură. Să presupunem că utilizatorul a introdus valoarea 5 și apoi a acționat <b>ENTER</b> . Valoarea 5 va fi depusă în locația de memorie cu numele N.	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> S <input type="text"/>  N <input type="text" value="5"/>  I <input type="text"/> </div>

2.	20 LET S=0	Calculatorul utilizează o locație de memorie numită S și pune în ea valoarea 0.	S <input type="text" value="0"/> N <input type="text" value="5"/> I <input type="text"/>
3.	30 FOR I=1 TO N	Calculatorul utilizează o locație de memorie numită I și pune în ea valoarea 1.	S <input type="text" value="0"/> N <input type="text" value="5"/> I <input type="text" value="1"/>
4.	40 LET S=S+I	Calculatorul trebuie să evalueze expresia din partea dreaptă a semnului egal, adică S + I. Pentru aceasta citește valorile 0 și 1 din celulele de memorie S, și respectiv I, adunarea lor realizându-se în unitatea aritmetică logică (UAL). Valoarea obținută (1) se depune în locația de memorie S.	S <input type="text" value="1"/> N <input type="text" value="5"/> I <input type="text" value="1"/>

5.	50 NEXT I	<p>Calculatorul adaugă valorii din locația I o unitate (<math>I = I + 1</math>), adică trece la următorul I. Calculatorul citește valoarea din locația de memorie I și o compară cu valoarea lui N (<math>I &gt; N</math>?). Cum valoarea lui I este 2, deci mai mică decât 5 atunci calculatorul transferă controlul la linia numărul 30.</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>S <input type="text" value="1"/></p> <p>N <input type="text" value="5"/></p> <p>I <input type="text" value="2"/></p> </div>
6.	30 FOR I=1 TO N	<p>Calculatorul reia ciclul.</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>S <input type="text" value="1"/></p> <p>N <input type="text" value="5"/></p> <p>I <input type="text" value="2"/></p> </div>
7.	40 LET S = S+I	<p>Calculatorul evaluează expresia <math>S + I</math>. El citește din locația de memorie S valoarea 1 și din locația de memorie I valoarea 2, evaluând deci expresia <math>S + I</math> la valoarea 3. Această valoare este depusă în locația de memorie S.</p>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> <p>S <input type="text" value="3"/></p> <p>N <input type="text" value="5"/></p> <p>I <input type="text" value="2"/></p> </div>

8.	50 NEXT I	<p>Calculatorul adaugă valorii din locația I o unitate (<math>I = I + 1</math>). Compară apoi această valoare cu valoarea din locația N și, fiind mai mică (3), transferă controlul la linia cu numărul 30.</p>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">S <input style="width: 30px;" type="text" value="3"/></div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">N <input style="width: 30px;" type="text" value="5"/></div> <div style="border: 1px solid black; padding: 5px;">I <input style="width: 30px;" type="text" value="3"/></div>
9.	30 FOR I=1 TO N	Calculatorul reia ciclul	<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">S <input style="width: 30px;" type="text" value="3"/></div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">N <input style="width: 30px;" type="text" value="5"/></div> <div style="border: 1px solid black; padding: 5px;">I <input style="width: 30px;" type="text" value="3"/></div>
10.	40 LET S=S+I	<p>Deoarece <math>S = 3</math> și <math>I = 3</math> rezultă că <math>S + I = 3 + 3 = 6</math>. Calculatorul depune în locația S valoarea 6.</p>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">S <input style="width: 30px;" type="text" value="6"/></div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">N <input style="width: 30px;" type="text" value="5"/></div> <div style="border: 1px solid black; padding: 5px;">I <input style="width: 30px;" type="text" value="3"/></div>
11.	50 NEXT I	<p>Calculatorul trece la următoarea valoare din locația I. Aceasta este 4 fiind deci, mai mică decât cea din locația N care este 5. Calculatorul transferă controlul la linia cu numărul 30.</p>	<div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">S <input style="width: 30px;" type="text" value="6"/></div> <div style="border: 1px solid black; padding: 5px; margin-bottom: 5px;">N <input style="width: 30px;" type="text" value="5"/></div> <div style="border: 1px solid black; padding: 5px;">I <input style="width: 30px;" type="text" value="4"/></div>

12.	30 FOR I=1 TO N	Calculatorul reia ciclul.	S <input type="text" value="6"/> N <input type="text" value="5"/> I <input type="text" value="4"/>
13.	40 LET S=S+I	Deoarece S = 6 și I = 4 rezultă că S + I = 6 + 4 = 10. Calculatorul depune în locația S valoarea 10.	S <input type="text" value="10"/> N <input type="text" value="5"/> I <input type="text" value="4"/>
14.	50 NEXT I	Calculatorul adaugă valorii din locația I o unitate (I = I + 1), apoi compară valoarea din locația I cu valoarea din locația N. Valoarea din locația I nefiind mai mare decât cea din N (ele sînt egale), transferă controlul la linia cu numărul 30.	S <input type="text" value="10"/> N <input type="text" value="5"/> I <input type="text" value="5"/>
15.	30 FOR I=1 TO N	Calculatorul reia ciclul.	S <input type="text" value="10"/> N <input type="text" value="5"/> I <input type="text" value="5"/>

16.	40 LET S=S+I	<p>Calculatorul - evaluează expresia <math>S + I</math>. El citește din locația S valoarea 10 și din locația de memorie I valoarea 5 evaluând expresia <math>S + I</math> la valoarea 15. Această valoare este depusă în locația de memorie S.</p>	<p>S <input type="text" value="15"/></p> <p>N <input type="text" value="5"/></p> <p>I <input type="text" value="5"/></p>
17.	50 NEXT I	<p>Calculatorul adaugă valorii din locația I o unitate. Calculatorul compară valoarea din locația I cu valoarea din locația N și aceasta fiind mai mare (<math>I &gt; N</math>) nu mai transferă controlul la linia cu numărul 30, ci la linia următoare, care este linia cu numărul 60.</p>	<p>S <input type="text" value="15"/></p> <p>N <input type="text" value="5"/></p> <p>I <input type="text" value="6"/></p>
18.	60 PRINT S	<p>Calculatorul citește valoarea din locația de memorie S și afișează această valoare. Pe ecran va apare valoarea 15.</p>	<p>S <input type="text" value="15"/></p> <p>N <input type="text" value="5"/></p> <p>I <input type="text" value="6"/></p>

19.	—	Deoarece după linia 60 nu mai este nici o linie de program, calculatorul afișează un mesaj de terminare a programului: 0 OK 60: 1 ceea ce înseamnă că programul s-a terminat în linia 60 și a fost executat fără nici o eroare.	<table border="1"> <tr> <td>S</td> <td>15</td> </tr> <tr> <td>N</td> <td>5</td> </tr> <tr> <td>I</td> <td>6</td> </tr> </table>	S	15	N	5	I	6
S	15								
N	5								
I	6								

tabel 1

*Detalierea execuției programului pentru adunarea a N numere naturale*

După cum se poate observa pentru executarea acestui program au fost necesari nu mai puțin de 19 pași de program, rezultatul fiind cel corect ( $1 + 2 + 3 + 4 + 5 = 15$ ). Iar pentru fiecare pas de program calculatorul a avut de executat una sau mai multe operații. De exemplu, pentru pasul 7, adică pentru executarea liniei 40 `LET S = S + I` calculatorul a executat 4 operații:

1. a citit valoarea din locația S (care este 1)
2. a citit valoarea din locația I (care este 2)
3. a evaluat expresia  $S + I$  ( $1 + 2 = 3$ )
4. a depus valoarea obținută (3) în locația S

Poate vi se pare monotonă și plictisitoare funcționarea în acest mod a calculatorului. Tocmai în aceasta constă și tăria lui, și anume, în faptul că poate executa foarte repede și corect multe operații asemănătoare și, de fapt, foarte simple. În speță calculatorul, cu toate că în final poate să rezolve probleme grele și să execute lucruri foarte interesante, nu "știe" să facă decât cele 4 operații aritmetice.

Este bine ca pentru orice program (problemă) pe care îl dăm calculatorului să îl execute să ne imaginăm (în modul în care am făcut-o pentru programul care calculează suma primelor N numere naturale) felul în care va



funcționa calculatorul și care va fi evoluția parametrilor programului (celulelor de memorie).

De obicei această *simulare* a funcționării programului se face cu puține valori. Dacă rezultatul va fi cel așteptat atunci va fi de presupus că și cu valori mai multe (reale) rezultatele vor fi corecte sau, cu alte cuvinte, programul "merge".

Dacă totuși în cazul executării programului rezultatul nu este cel așteptat atunci se poate proceda la *depanarea* programului prin întreruperea funcționării acestuia în diferite momente și prin "fotografierea" conținutului locațiilor de memorie (variabilelor) pentru a se vedea dacă evoluția acestora este corectă. Întreruperea funcționării programului se poate face prin inserarea unor instrucțiuni STOP după calculul în cadrul programului a valorilor parametrilor sau, pur și simplu, prin întreruperea programului în mod comandă (prin BREAK sau STOP). Pentru "fotografierea" conținutului variabilelor se pot insera după instrucțiunile STOP instrucțiuni de afișare (PRINT) a valorilor parametrilor care ne interesează. În cazul întreruperii programului cu BREAK sau STOP se vor da în mod direct comenzi de afișare (PRINT) a valorilor parametrilor care ne interesează.

De exemplu, să presupunem că întrerupem programul nostru cu BREAK și apoi citim valorile parametrilor cu:

```
PRINT N —> obținem valoarea 5 (corect)
PRINT I —> (să presupunem că se afișează valoarea 3)
PRINT S —> obținem valoarea 6 (corect)
```

Concluzionăm că programul "merge" deoarece pentru primele 3 numere naturale ( $I=3$ ) suma este 6.

În ambele cazuri de întrerupere, după afișarea valorilor, execuția programului se poate continua cu comanda CONT (CONTINUE), atâta timp cât valorile parametrilor nu au fost modificate. Vizualizând astfel evoluția parametrilor ne vom putea da seama în ce loc programul "nu merge bine", adică, de fapt, în ce loc valorile parametrilor nu mai corespund cu valorile așteptate de noi.

Pentru unele calculatoare (nu și pentru cele compatibile Sinclair Spectrum din păcate) există posibilitatea ca printr-o anumită comandă (opțiune) execuția programului să fie însoțită de afișarea liniilor de program care se execută în acel moment. Această operație se numește **trasarea programului** iar comanda este TRACE (TRON pentru PC în cadrul sistemului GWBASIC). Se oferă o modalitate utilă de a se vizualiza în același timp atât rezultatele cât și numerele de linie respective ale căror instrucțiuni conduc la aceste rezultate, fiind deci, *o unealtă de depanare a programelor*.



# Rezolvarea problemelor cu calculatorul

Ne propunem să vă învățăm cum să rezolvați probleme de matematică (dar nu numai) folosind ca instrument de lucru calculatorul. Bineînțeles că vă vom propune și probleme de alt tip care se referă la alte domenii decât matematica, acestea incluzându-se în lucrare deoarece rezolvarea lor necesită de asemenea cunoștințe și metode matematice. Printre aceste probleme se pot enumera *problemele de grafică* și cele de tip *jocuri logice*.

Desigur în abordarea problemelor propuse vom căuta inițial o rezolvare uzuală (clasică) iar apoi vom încerca transpunerea acestei rezolvări în scopul tratării prin intermediul calculatorului. În situația în care nu găsim o rezolvare uzuală convenabilă se propune o nouă metodă mai apropiată de calculator. În ambele cazuri calculatorul este privit ca un instrument care ajută la rezolvarea problemei în special prin preluarea unor calcule repetate și plictisitoare care ar fi mari consumatoare de timp pentru rezolvitor.

Deci, în primul rând vom studia cum rezolvă calculatorul problemele. În acest scop va trebui să găsim o metodă pentru soluționarea problemei, iar apoi să concepem un program cu care se va rezolva efectiv problema.

Metoda de a rezolva o problemă se numește **algoritm**.



## Algoritmi

---

Algoritmul este reprezentat de o mulțime de reguli folosite într-o anumită succesiune, cu ajutorul cărora se poate obține soluția unei probleme prin executarea unui număr finit de operații. Aceasta este o definiție și nu este necesară memorarea ei. Mai importantă este înțelegerea ei, iar cum înțelegerea se realizează cel mai bine prin analogii, puteți să vă imaginați că o rețetă dintr-o carte de bucate este, de fapt, un algoritm. De ce? Deoarece o rețetă reprezintă o mulțime de reguli care trebuie urmate, adică folosite într-o anumită succesiune pentru a obține un anumit fel de mâncare sau o prăjitură, adică soluția problemei.

Trebuie să remarcăm faptul că deseori obținerea algoritmului (sau rețetei) nu este o treabă foarte ușoară. De exemplu pentru a se ajunge la rețetele unora dintre cele mai gustoase feluri de mâncare s-au făcut nenumărate încercări și s-a folosit experiența acumulată de gospodine și bucătari timp de zeci (uneori chiar sute) de ani. Având însă rețeta (sau algoritmul) nu va fi foarte dificil de a obține prăjitura dorită (sau *soluția problemei*).

O metodă prin care se pot găsi algoritmi pentru rezolvarea problemelor este **metoda de abordare de sus în jos a problemelor**. Prin această metodă problema este descompusă în mai multe subprobleme și, pe măsură ce aceste subprobleme vor fi mai mici, ele vor semăna din ce în ce mai mult cu pașii logici care pot fi reprezentați prin operații simple pe care le poate efectua calculatorul. Metoda este eficientă, dar, pentru rezolvarea problemelor care se descompun în foarte multe subprobleme, deseori este dificil de urmărit șirul logic al algoritmului. În scopul ușurării urmăririi șirului logic în aceste situații se utilizează scheme cum sînt diagramele de tip arbore. Prin *diagramele de tip arbore* se realizează o descriere grafică a subprogramelor și pașilor logici de urmat pentru rezolvarea problemei.

Iată un exemplu de început de diagramă de tip arbore realizată în scopul rezolvării problemei de realizare a operațiunilor logice cu două mulțimi A și B (diferență, intersecție și reuniune) problemă care va fi reluată într-un capitol dedicat realizării acestui program (figura 2).

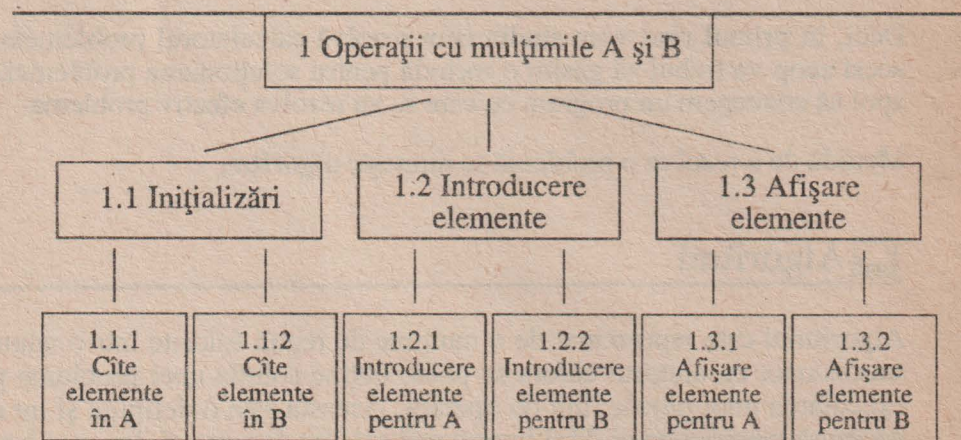


fig.2

Diagramă tip arbore pentru realizarea operațiilor logice diferență, intersecție și reuniune a două mulțimi

Pentru exemplificarea metodei de abordare prin care problemele se pot descompune ("sparge") în subprobleme mai mici, vom evidenția faptul că fiecare din subproblemele 1.2.1. și 1.2.2. mai pot conține câte o sarcină 1.2.1.1. și 1.2.2.1. de verificare a elementelor introduse pentru mulțimea A și respectiv B.

După realizarea unei astfel de diagrame se caută să se obțină o descriere a algoritmului într-un limbaj succint astfel încât pașii de urmat să fie ușor de înțeles și, în același timp, ușor de tradus în instrucțiuni pentru calculator, operație care se mai numește descrierea în **pseudocod** a algoritmului. Acest lucru va fi mult mai ușor de realizat din descrierea sarcinilor ce rezultă din diagrama de tip arbore. Bineînțeles, pentru a produce programul va trebui ca fiecare secțiune a pseudocodului să fie tradusă (translatată) în echivalentul său din limbaj BASIC. Iată cum arată descrierea în pași logici a diagramei de structură prezentate:

Pentru a distinge mai bine problemele și subproblemele de rezolvat același lucru îl vom scrie astfel (mai structurat):

⇒ 1 Operații cu mulțimile A și B

⇒ 1.1 Inițializări.

⇒ 1.1.1 Câte elemente în A.

⇒ 1.1.2 Câte elemente în B.

⇒ 1.2 Introducere elemente.

⇒ 1.2.1 Introducere elemente pentru A.

⇒ 1.2.1.1 Verificare elemente introduse pentru A (să nu se repete).

⇒ 1.2.2 Introducere elemente pentru B.

⇒ 1.2.2.1 Verificare elemente introduse pentru B (să nu se repete)

⇒ 1.3 Afișare elemente.

⇒ 1.3.1 Afișare elemente din A.

⇒ 1.3.2 Afișare elemente din B.



## Rezolvarea problemei

---

☞ *Găsirea metodei de a rezolva o problemă este, de obicei, cea mai dificilă parte a programării. De aceea, de la început este necesară organizarea și planificarea activităților.*

Pentru a produce algoritmul de rezolvare a unei probleme vor trebui parcurse mai multe etape pe care le descriem în continuare, folosind pentru acomodare tot o descriere în pași logici:

- ☞ 1 Privire generală asupra problemei
  - ☞ 1.1 Ce rezultate urmărești să obții
  - ☞ 1.2 Ce date este necesar să introduci
  - ☞ 1.3 Ce operații vei efectua cu aceste date
- ☞ 2 Analiza problemei
  - ☞ 2.1 Analiza problemei pentru identificarea modului în care poate fi rezolvată cu calculatorul
  - ☞ 2.2 Identificarea formulelor, datelor și relațiilor pe care le vei folosi
  - ☞ 2.3 Identificarea tuturor datelor implicate
- ☞ 3 Proiectarea algoritmului de rezolvare a problemei
  - ☞ 3.1 Descompunerea problemei în subprobleme
  - ☞ 3.2 Utilizarea unei diagrame de structură (de exemplu de tip arbore) ca ajutor
  - ☞ 3.3 Identificarea tipului modulelor sau părților ca fiind de introducere, prelucrare sau extragere
  - ☞ 3.4 Utilizarea structurilor de control

Pentru evidențierea celor mai importante structuri de control se pot utiliza **schemele logice**. Acestea reprezintă o metodă utilizată în realizarea de programe care constă din *blocuri logice* legate între ele. Fiecare *bloc* reprezentat prin diverse figuri geometrice, cu o anumită semnificație, descrie ce realizează programul în acel punct, *schema logică* descriind astfel fluxul controlului în algoritm și deci în program. Schema logică evi-

dențiază deci, cele mai importante *structuri de control* și ușurează astfel codificarea algoritmului în instrucțiuni BASIC, constituind, de asemenea, și o parte importantă a documentației programului.

Să exemplificăm o schemă logică care ar corespunde sarcinii 1.2.1.1. din pseudocod, și anume, verificarea elementelor introduse pentru mulțimea A (să nu se repete elementele). Algoritmul de rezolvare a acestei sarcini este următorul: fiecare element din mulțimea A introdus,  $a(i)$  se compară cu toate elementele din stînga sa, adică, cu  $a(1), a(2) \dots a(i-1)$ . Dacă se găsește că este egal cu unul din elementele din stînga lui atunci se afișează mesajul "Valoare eronată. Introduceți altă valoare", iar fluxul programului va merge în punctul în care se realizează introducerea valorii  $a(i)$ , punct care aparține sarcinii 1.2.1. Dacă elementul  $a(i)$  nu este egal cu nici unul din stînga sa atunci valoarea introdusă este corectă și se trece la introducerea valorii următoare,  $a(i+1)$ . Schema logică arată astfel (figura 3):

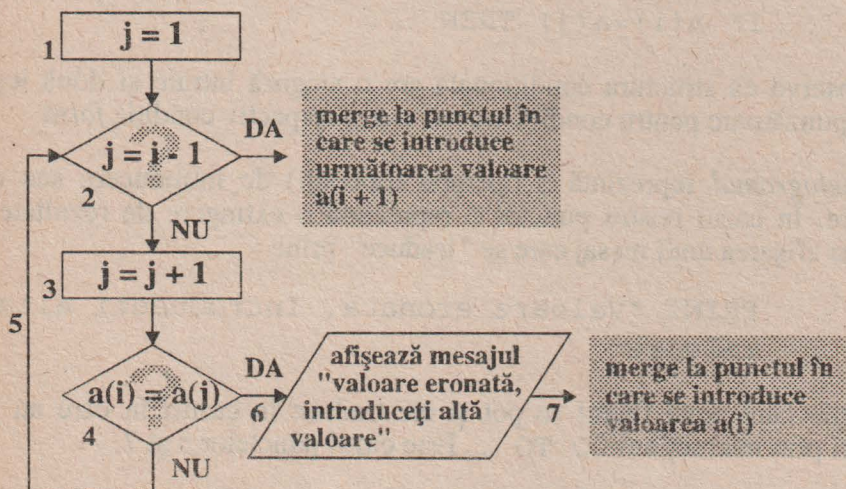


fig.3

Schema logică pentru verificarea elementelor introduse și care aparțin unei mulțimi date

Figurile geometrice din desen se mai numesc **blocuri**.

Dreptunghiurile reprezintă *structuri de prelucrare*. În cazul nostru dreptunghiurile 1 și 3 reprezintă *operații de atribuire*.

Acum putem să le "traducem" lesne în instrucțiuni BASIC. Astfel pentru punctul 1 se poate introduce o linie de program

```
LET j=1
```

iar pentru 3

```
LET j=j+1
```

Romburile reprezintă *structuri condiționale*. Astfel punctul 2 se poate "traduce" în BASIC prin:

```
IF j=i-1 THEN GO TO ...
```

iar punctul 4:

```
IF a(i)=a(j) THEN ...
```

Se observă că structura condițională are o singură intrare și două ieșiri corespunzătoare pentru condiție *adevărată* și respectiv condiție *falsă*.

*Paralelogramul* reprezintă o operație (operații) de introducere sau extragere. În cazul nostru punctul 6 reprezintă o extragere de rezultate și anume afișarea unui mesaj care se "traduce" prin:

```
PRINT "Valoare eronata. Introduceți alta  
valoare"
```

Săgețile care arată fluxul se pot și ele traduce în cazuri în care au loc salturi prin instrucțiuni GO TO .... Este cazul punctelor 5 și 7.

Deci următoarele etape pentru rezolvarea problemei vor fi:

- ⇒ 4. Realizarea programului de rezolvare a problemei (codificarea în limbaj BASIC sau alt limbaj de programare)
- ⇒ 5. Testarea (verificarea) programului
  - ⇒ 5.1. Verificarea sintactică (cu îndepărtarea erorilor dacă acestea există)



⇒ 5.2. Verificarea logică cu mai multe seturi de date (cu îndepărtarea erorilor dacă acestea există)

⇒ 6. Obținerea rezultatelor.

☞ *Lucrarea de față propune spre rezolvare numeroase probleme de matematică. În scopul clarificării modului de prezentare a rezolvării se pune mai mult accent pe analiza problemei, descrierea succintă (asemănătoare cu pseudocodul) a ideilor pentru rezolvarea ei și prezentarea programului. Din acest punct de vedere rezolvarea oferită în lucrare trebuie privită ca un îndrumar în vederea rezolvării problemei, recomandându-se ca atunci când se lucrează practic să nu se sară pașii logici descriși în acest capitol. Bineînțeles că soluțiile (programele) pe care le obțineți pot diferi de cele oferite în lucrare. Important este ca ele să fie corecte, adică prin intermediul lor să se obțină rezultatele corecte.*



## Probleme cu numere

### Multiplii numerelor

---



Să se realizeze un program cu care să se determine multiplii unui număr.

Pentru calculul multiplilor unui număr se pot utiliza două metode:

- calculul multiplilor prin adunare
- prin înmulțire.

Pentru metoda a) se memorează valoarea numărului ai cărui multipli se dorește să fie calculați într-o variabilă  $M$  ( $LET M = A$ ). Valoarea  $M$  va reprezenta, de fapt, valoarea primului multiplu. Următorii multipli se determină prin același procedeu, și anume se adună la numărul care reprezintă multiplul actual o valoare (pas) care reprezintă chiar valoarea numărului inițial căruia dorim să-i calculăm multiplii. Programul P1 este:

```
P1. 5  REM **  program de calcul al  **  
6  REM **  multiplilor unui numar  **  
7  REM **           prin adunare   **  
10  INPUT A  
20  LET M = A  
30  PRINT M  
40  LET M = M + A  
50  GOTO 30
```

Astfel linia 40 este cea în care se calculează multiplii și care evidențiază metoda folosită și anume aceea a adunării. Să presupunem că introducem la executarea programului valoarea 7. Se vor afișa la nesfârșit multiplii

numărului introdus, adică 7, 14, 21, 28, 35, 42,... . Pentru oprirea programului se va acționa

**BREAK (CS + SPACE)** pentru HC

**CTRL + PAUSE** pentru PC.

Pentru metoda b) multiplii se vor determina prin înmulțirea numărului inițial cu 1, 2, 3 și așa mai departe. De exemplu, pentru determinarea multiplilor lui 5, aceștia se obțin prin  $5 \times 1 = 5$ ,  $5 \times 2 = 10$ ,  $5 \times 3 = 15$  și așa mai departe. În acest scop va trebui să introducem o variabilă contor (C) care să ia valorile 1, 2, 3, ... și cu care să înmulțim numărul inițial pentru calculul fiecărui multiplu. Programul P2 este:

```
P2. 5. REM ** program de calcul al **
      6. REM ** multiplilor unui numar **
      7. REM ** prin inmultire **
      10 INPUT A
      20 LET C = 1
      30 LET M = A * C
      40 PRINT M
      50 LET C = C + 1
      60 GOTO 30
```

Linia 30 este cea în care se calculează multiplii și care evidențiază metoda folosită și anume cea a înmulțirii. La fel ca în cazul programului P1, introducând valoarea 7 se vor afișa multiplii lui unul sub altul: 7, 14, 21, 28, 35, 42,....



Să se modifice programele de calcul al multiplilor unui număr astfel încât fiecare multiplu să fie afișat împreună cu numărul de ordine respectiv (al câteia multiplu).

Pentru cazul b) trebuie doar modificată linia 40 astfel:

```
40 PRINT C , M
```

Pentru cazul a) în vederea afișării numărului de ordine trebuie introdusă o variabilă contor I după modelul folosit în cazul b).

Se adaugă deci liniile:

```
25 LET I = 1
45 LET I = I + 1
```

linia 30 se modifică astfel:

```
30 PRINT I , M
```

La execuția ambelor programe se vor afișa atât multiplii cât și numărul care reprezintă ordinea lor. De exemplu, dacă se introduce valoarea 7 se va obține:

```
1      7
2     14
3     21
4     28
⋮      ⋮
```

Dacă numărul de multipli pe care dorim să-i determinăm este cunoscut atunci programul se va putea realiza folosind un ciclu FOR-NEXT (programul P3):

```
P3. 5 REM ** program de calcul al **
6 REM ** multiplilor unui numar **
7 REM ** cu ciclu FOR-NEXT **
10 INPUT A
15 REM se introduce numarul de
16 REM multipli de calculat
20 INPUT N
30 FOR I = 1 TO N
40 LET M = A * I
50 PRINT I, M
60 NEXT I
```

În acest caz se va afișa un anumit număr de multipli ai unui număr. De exemplu, dacă se introduc valorile 17 și 4 se vor afișa primii 4 multipli ai numărului 17.


## Cel mai mic multiplu comun

---



Cum se poate determina cu ajutorul programului cel mai mic multiplu comun (CMMMC) a două numere?

Va trebui mai întâi să introducem cele două numere (A și B) și apoi să generăm multiplii acestor două numere. Generarea multiplilor o putem face înmulțind fiecare din numerele A și B cu 1, 2, 3 și așa mai departe folosind metoda de calcul a multiplilor prin înmulțire (cazul b). Apoi vom afișa pe o coloană numărul de ordine (pentru a ști al câtelea multiplu este), iar pe următoarele două coloane multiplii respectivi (M1 și M2). Programul P4 va arăta astfel:

```
 P4. 5 REM ** program de calcul al CMMMC **
6 REM ** al doua numere **
10 INPUT A
20 INPUT B
30 LET I = 1
40 LET M1 = A * I
50 LET M2 = B * I
60 PRINT I; " "; M1; " "; M2
70 LET I = I + 1
80 GOTO 40
```

Se vor genera multipli pînă cînd vom opri programul. De exemplu, introducînd numerele 3 și 4 (pentru A și respectiv B) vom putea citi din cele afișate pe ecran că al 7-lea multiplu al lui 3 este 21 iar al lui 4 este 28. Cînd s-a umplut un ecran întreg cu multipli și nu mai avem nevoie de generarea altor multipli vom opri programul acționînd tasta **N** sau **BREAK (CS + SPACE)** pentru HC **CTRL + PAUSE** pentru PC.

*Cum vom identifica cel mai mic multiplu comun al celor două numere?*

Simplu! Ne uităm pe coloana a 2-a și a 3-a, iar primul număr pe care îl vom identifica ca fiind în ambele coloane va reprezenta CMMMC, deoa-

rece el este multiplu atât pentru primul număr (**A**), deoarece este în a 2-a coloană, cât și pentru al doilea număr (**B**), deoarece este în a 3-a coloană și, în plus, el este primul dintre multipli (deci cel mai mic) care este același pentru ambele numere. De exemplu, pentru  $A=3$  și  $B=4$ , vom observa că numărul 12 este afișat pe coloana a 2-a (al 4-lea multiplu al lui 3) și, același număr este afișat și pe coloana a 3-a (al 3-lea multiplu al lui 4). Deci 12 reprezintă CMMMC pentru 3 și 4.

Ca alt exemplu, introducând numerele 12 și 42 vom observa numărul 84 afișat și pe coloana a 2-a (al 7-lea) și pe coloana a 3-a (al 2-lea). El reprezintă CMMMC pentru 12 și 42.

Dacă dorim să determinăm CMMMC pentru mai multe numere putem să afișăm multipli pe mai multe coloane și, în mod similar, primul număr care va fi afișat în toate coloanele (cu excepția primei care nu exprimă multipli) va reprezenta CMMMC al numerelor respective.

## Sume și produse de numere



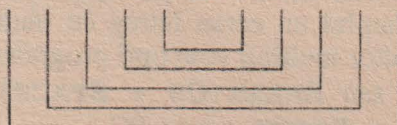
**Care este suma primelor 10 numere naturale ? Dar a primelor 100 de numere ?**

Desigur, pentru suma primelor 10 numere naturale putem calcula  $1 + 2 + 3 + \dots$  și așa mai departe pînă la 10. Mai ingenios însă putem calcula observînd că se pot asocia 0 cu 10, 1 cu 9, 2 cu 8, 3 cu 7 și 4 cu 6 toate aceste sume fiind 10 (vezi figura 4):

fig.4

*Reprezentarea grafică a metodei de adunare a primelor 10 numere naturale*

$$0+1+2+3+4+5+6+7+8+9+10$$



Avem 5 sume de 10 iar 5 din mijloc rămîne liber, deci:  $5 \times 10 + 5 = 55$ .

*Puteți da un răspuns aproximativ, dar rapid (fără să faceți efectiv calculul) la întrebarea care este suma primelor 100 de numere ?*

Putem folosi aceeași regulă, adică să adunăm pe 0 cu 100, pe 1 cu 99 ... și vom avea 50 de sume (de la 0 la 49) de câte 100 iar 50 va rămîne singur, deci:  $100 \times 50 + 50 = 5050$ .

Răspunsul vostru la întrebarea precedentă a fost cît de cît apropiat de cel real? Prin compararea rezultatului sumei primelor 10 numere cu cel obținut pentru 100 de numere, puteți spune cît va fi rezultatul sumei primelor 1000 de numere?

Desigur, aceasta va fi 500 500, iar al sumei primelor 10 000 va fi 50 005 000.

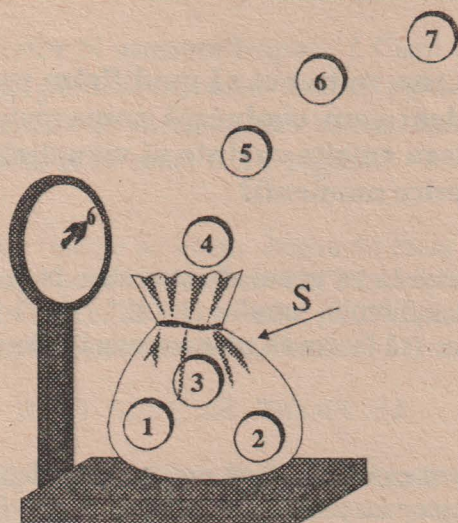


**Cum se poate face un program astfel încît să realizăm calculul unor astfel de sume cu calculatorul?**

Pentru aceasta trebuie să imaginăm un algoritm, adică să-l învățăm pe calculator o regulă pentru a calcula astfel de sume.

fig.5

Adunarea numerelor naturale



Vom proceda în felul următor: luăm un sac (sau, dacă vreți, o celulă de memorie) pe care îl botezăm S (îi punem o etichetă S). La început el este gol (deci LET  $S = 0$ ) și începem să punem în el câte un număr (FOR  $I = 1$  TO  $N$ ). De fiecare dată suma va fi egală cu suma existentă pînă în

acel moment în sac la care se mai adaugă numărul respectiv (deci  $LET S = S + I$ ). După ce am pus toate numerele ne uităm să vedem ce sumă avem în sac ( $PRINT S$ ) și astfel obținem rezultatul dorit. Programul P5 va fi:

```
P5. 5 REM ** program de calcul al sumei **
    6 REM ** primelor N numere naturale **
    10 INPUT N
    20 LET S = 0
    30 FOR I = 1 TO N
    40 LET S = S + I
    50 NEXT I
    60 PRINT S
```

Într-adevăr, acum, avînd afișate toate numerele putem verifica foarte ușor rezultatele pentru suma primelor 1000 sau 10 000 sau 100 000 de numere și să regăsim regula de formare a numărului care reprezintă suma.



**Cum va trebui să modificăm programul astfel încît să vedem cum evoluează suma pentru fiecare număr adunat (sau cu alte cuvinte să vizualizăm suma care este în sac în orice moment)?**

Pentru aceasta va trebui să intercalăm în programul P5 o linie de vizualizare a conținutului sacului ( $PRINT$ ) după fiecare introducere a unui număr în sac (să facem sacul transparent). Deci introducem linia

```
45 PRINT S
```

iar în acest caz linia 60 nu mai are rost, deoarece ultima sumă afișată va fi chiar suma care este în sac după ce au fost introduse toate numerele. Dacă dorim să știm de fiecare dată atît suma cît și numărul de numere care adunate dau suma respectivă va trebui să vizualizăm, în afară de sumă și numărul respectiv, care este, de fapt,  $I$ . Deci linia va fi :

```
45 PRINT I , S
```





Verificați acum cu programul de calcul al primelor N numere naturale următoarea formulă:

$$S = \frac{N \cdot (N + 1)}{2}$$

Vom observa că formula este valabilă pentru 1, pentru 2 și pentru orice număr I. Pentru această verificare vom modifica în programul P5 linia 45 astfel:

```
45 PRINT I; " "; S; " "; I * (I + 1) / 2
```

și vom vedea că pe fiecare rând, adică pentru orice I se vor afișa două numere egale.



Care este suma primelor N numere impare? Cum trebuie modificat programul pentru calculul sumei primelor N numere naturale astfel încât să calculeze numai suma numerelor impare? Există și în acest caz o regulă?

Pentru a calcula suma numerelor impare nu avem altceva de făcut decât să modificăm linia 30 făcând calculul sumei cu numere din 2 în 2 (STEP 2). De asemenea trebuie să avem în vedere că în intervalul [0, N] jumătate din numerele naturale (N/2) sînt impare, iar restul pare. Deci:

```
30 FOR I = 1 TO N STEP 2
```

Pentru primele 10 numere, deci, pentru primele 5 numere impare obținem suma 25, pentru primele 100 numere obținem suma celor 50 impare 2 500, iar pentru 1 000 obținem 250 000.

Evident, în acest caz regula este mai simplă și anume, de adăugare a două zerouri după grupul de cifre 25 pentru fiecare ordin de mărime. Deci, pentru 10 000 suma va fi 25 000 000.



Care este suma primelor  $N$  numere pare ? Care este în acest caz regula?

Singura modificare față de programul precedent P5 este faptul că se va începe contorizarea de la 2 și nu de la 1 ca în cazul numerelor impare. În acest caz linia 30 va arăta astfel:

```
30 FOR I = 2 TO N STEP 2
```

Pentru primele 10 numere, deci pentru primele 5 numere pare (fără să-l includem pe 0 deoarece am început cu 2) suma este 30, pentru 100 este 2 550, iar pentru 1 000 este 250 500. Pentru 10 000 suma va fi 2 500 500. Deci se adaugă câte un 0 alături grupului de cifre condus de 25 cât și celui condus de 5.



Cum se poate calcula media aritmetică a  $N$  numere?

Știm că media aritmetică a  $N$  numere se calculează împărțind suma numerelor la  $N$ . Pentru aceasta trebuie să știm de la început câte numere avem (linia 10). Se observă că în acest caz numerele pe care le avem de adunat nu le mai cunoaștem dinainte ci trebuie să le introducem. Acest lucru îl vom realiza prin inserarea liniei 35 în programul P5:

```
35 INPUT A
```

De exemplu, pentru a calcula media trimestrială la școală vom introduce separat mediile pentru fiecare materie. Linia 40 se va modifica în mod corespunzător având acum de adăugat de fiecare dată la vechea sumă noul număr introdus. În final nu vom fi interesați de suma numerelor ci de media lor care se obține prin împărțirea sumei la  $N$ . În consecință în linia 60 vom afișa rezultatul  $S/N$ . Programul P6 va fi :



```
P6. 5 REM ** program de calcul al **
    6 REM ** mediei aritmetice a **
    7 REM **          N numere          **
```

```

10 INPUT N
20 LET S = 0
30 FOR I = 1 TO N
35 INPUT A
40 LET S = S + A
50 NEXT I
60 PRINT S/N

```



**Care este suma pătratelor primelor  $N$  numere naturale? Puteți estima cam cât este, de exemplu, suma pătratelor primelor 100 de numere? Există vreo regulă de formare a numărului care reprezintă suma?**

Știm că pătratul unui număr reprezintă numărul obținut prin înmulțirea numărului cu el însuși, sau, cu alte cuvinte, aria unui pătrat a cărui latură este egală cu însuși numărul.

Pentru a calcula suma pătratelor numerelor va trebui să modificăm linia 40 în programul P5 de calcul al sumei numerelor în felul următor:

```
40 LET S = S + I * I
```

Pentru suma pătratelor primelor 10 numere vom obține 385, pentru primele 100 338 350, iar pentru primele 1 000 vom obține 333 833 500. Vă lăsăm să formulați singuri regula de obținere a numărului.

Se poate verifica și următoarea formulă:

$$S = \frac{N \cdot (N + 1) \cdot (2N + 1)}{6}$$

În acest scop vom introduce din nou linia 45 în felul următor:

```
45 PRINT I; " "; S; " "; I * (I + 1) * (2 * I + 1) / 6
```

Vom observa că pe fiecare rând afișat, adică pentru orice  $I$ , ultimele două numere sînt egale.



Care este suma cuburilor primelor  $N$  numere ? Există vreo regulă de formare a numărului care reprezintă această sumă?

Cubul unui număr este reprezentat de numărul înmulțit de 3 ori cu el însuși sau, cu alte cuvinte, volumul unui cub a cărui muchie este egală chiar cu numărul. Bineînțeles, pentru a calcula suma numerelor, linia 40 va avea următoarea formă:

```
40 LET S = S + I * I * I
```

Pentru primele 10 numere suma va fi 3 025, pentru 100, 25 502 500 iar pentru 1 000, 250 500 250 000. Vă lăsăm să formulați singuri regula de obținere a numărului sumă.



Care este suma inverselor primelor  $N$  numere naturale ? Puteți spune aproximativ cam cât este suma inverselor primelor 100 de numere?

Suma inverselor primelor  $N$  numere naturale va fi:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{N}$$

Observăm că numitorul fracțiilor care se adună este din ce în ce mai mare, deci și numărul mai mic. Pentru a realiza calculul vom modifica linia 40 a programului P5 de calcul al sumei primelor  $N$  numere astfel:

```
40 LET S = S + 1/I
```

Vom nota că suma inverselor primelor 2 numere este 1,5, suma inverselor primelor 3 numere este 1,83333 iar suma inverselor primelor 10 numere este 2,9289683. De asemenea, vom constata că suma inverselor primelor 100 de numere trece doar cu puțin de 5, iar suma inverselor a 1000 de numere este ceva mai mare de 7.



Care este suma inverselor pătratelor numerelor ? Puteți spune aproximativ cam cât este suma inverselor pătratelor primelor 1000 de numere?

Suma inverselor pătratelor numerelor va fi:

$$1 + \frac{1}{2 \cdot 2} + \frac{1}{3 \cdot 3} + \dots + \frac{1}{N \cdot N}$$

Deoarece numitorul crește mai repede și numerele adunate vor fi mult mai mici. Pentru program vom modifica linia 40 astfel:

$$40 \text{ LET } S = S + 1 / (I * I)$$

Veți constata cu mirare (poate) că suma inverselor pătratelor primelor 1000 de numere este de numai 1,6439346, iar pentru 10 000 de numere suma va diferi doar cu foarte puțin de precedenta (1,6448341).



Avînd la dispoziție programele de calcul al sumelor de numere, completați următorul tabel:

N	Suma primelor N numere naturale	Suma primelor N numere pare	Suma primelor N numere impare	Suma pătratelor primelor N numere naturale	Suma cuburilor primelor N numere naturale	Suma inverselor primelor N numere naturale
10	55	110	121	385	3025	2,9289683
100	5050	10100	10201	338350	25502500	. 5,
1000						
10000						
⋮	$\frac{N \cdot (N + 1)}{2}$			$\frac{N \cdot (N + 1) \cdot (2N + 1)}{6}$		

Încercați să regăsiți anumite reguli deja știute și să descoperiți altele.

De exemplu: este normal ca suma numerelor pare să fie dublă față de suma numerelor naturale ?

Da, deoarece fiecare număr care intră în componența sumei numerelor pare este dublu față de numărul omolog din cadrul sumei numerelor naturale.

De asemenea se poate observa că fiecare din numerele care intră în componența sumei numerelor impare este cu o unitate mai mare decât numărul omolog din cadrul sumei numerelor pare.



## Dacă șahul Persiei ar fi avut calculator...

---



Inventatorul jocului de șah a cerut șahului ca recompensă, să-i dea pentru prima căsuță a tablei de joc o boabă de grâu, pentru a 2-a, 2 boabe și așa mai departe, la fiecare căsuță să-i dubleze numărul de boabe (vezi figura 6). Șahul i s-a părut o recompensă modestă. Calculați cu ajutorul calculatorului câte vagoane de grâu trebuia să-i dea șahul recompensă inventatorului știind că 1000 de boabe de grâu = 1 kg și că 1 vagon = 10 t.

Va trebui să calculăm o sumă de forma:

$$1 + 2 + 2 \times 2 + 2 \times 2 \times 2 + \dots$$

adică:

$$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + \dots + 2^{63} \text{ (vezi figura 6)}$$

Ultimul termen al sumei este  $2^{63}$  deoarece am început numărarea căsuțelor de la 0 (de la 0 la 63 sînt 64 de căsuțe).

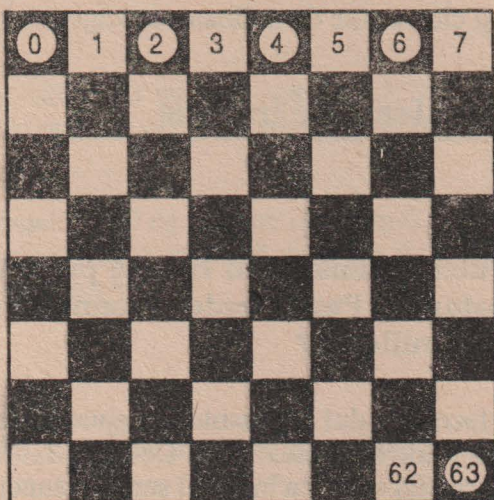
Pentru calculul sumei boabelor de grâu va trebui să modificăm linia 40 din programul P5 de calcul al sumei primelor N numere astfel:

$$40 \text{ LET } S = S + 2 \wedge I$$

fig.6

Reprezentarea grafică  
a metodei de calcul a  
recompensei inventa-  
torului jocului de șah

$$2^0 + 2^1 + 2^2 + 2^3 + 2^4 + 2^5 + 2^6 + 2^7 + \dots$$



$$\dots + 2^{62} + 2^{63}$$

Nu uităm să modificăm și linia 30 deoarece acum avem de repetat operația de adunare de un număr dat de ori și anume de 64 de ori (de la 0 la 63), adică:

```
30 FOR I = 0 TO 63
```

Bineînțeles nu vom mai avea nevoie de linia 10 deoarece acum **N** este definit în linia 30 și anume este egal cu 63. În final se va afișa numărul de vagoane. Acesta se va obține prin împărțirea numărului de boabe rezultat la 1000, se rezultând numărul de kilograme, apoi prin împărțirea acestui rezultat la 1000, obținându-se astfel numărul de tone și, în sfârșit, prin împărțirea rezultatului obținut la 10 se obține numărul de vagoane. Programul P7 va arăta astfel:



P7.

```
5 REM **      program de calcul al      **
6 REM **      recompensei inventatorului **
7 REM **      jocului de sah            **
20 LET S = 0
30 FOR I = 0 TO 63
40 LET S = S + 2^I
50 NEXT I
60 PRINT S/1000/1000/10
```

Vom obține ca rezultat  $1.8446744 \text{ E}+12$  ( $\text{E}+12$  înseamnă "înmulțit cu 10 la puterea 12"), adică 1 844 674 400 000 vagoane, adică un trilion, 844 miliarde și câteva sute de milioane de vagoane !

## O problemă de intuiție

---



**Puteți estima rapid cât este produsul primelor 10 numere naturale? Puteți crede că numărul respectiv este mai mare de 3 milioane?**

Pentru a face calculul produsului cu ajutorul calculatorului vom inițializa o variabilă P (de la produs) cu 1. De ce cu 1 ? Deoarece dacă am inițializa această variabilă cu 0 (ca în cazul sumei) atunci tot timpul produsul va fi 0 indiferent ce alte numere am mai înmulți cu el. Programul P8 va fi:

```
P8. 5 REM ** program de calcul al **
      6 REM ** produsului primelor N **
      7 REM **   numere naturale   **
      10 INPUT N
      20 LET P = 1
      30 FOR I = 1 TO N
      40 LET P = P * I
      50 NEXT I
      60 PRINT P
```

Pentru primele 10 numere vom obține 3 628 800, iar pentru 100 de numere calculatorul ne va răspunde că este un număr prea mare pentru a-l putea calcula.



**Care este produsul inverselor primelor N numere ?**

În acest caz cu cât N va fi mai mare cu atât numărul cu care se va înmulți va fi mai mic decât unitatea (deoarece numărul crește) și va avea ca efect



micșorarea numărului rezultat. Dar acesta nu va putea atinge practic niciodată 0. Linia 40 din programul P8 se va modifica astfel :

```
40 LET P = P / I
```

Pentru produsul inverselor primelor 10 numere obținem 2.7557319E- 7 (E-7 înseamnă înmulțit cu 10 la puterea -7) adică obținem de fapt numărul 0.0000002755.

Pentru produsul inverselor primelor 100 de numere obținem 0 ! Rezultatul este incorrect; calculatorul lucrează cu o anumită precizie și în limita ei identifică numerele (în speță, 0 cu  $9 \cdot 10^{-9}$ ).



Să se realizeze un program cu care să se calculeze expresia:

$$1 + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \dots + \frac{1}{1 \cdot 2 \cdot 3 \cdot \dots \cdot N}$$

Vom folosi atât metoda pentru calculul sumei (în primul rând este vorba de o sumă de numere) cât și metoda pentru calculul produsului a N numere. Este vorba, deci, de o *sumă de produse de numere*.

La fel cum pentru a construi o casă avem mai întâi nevoie de părțile ei componente (uși, ferestre etc.), tot așa pentru a calcula o sumă de produse vom calcula mai întâi produsele și apoi, pe baza lor suma. Programul P9 este:

 P9.

```
5 REM ** program de calcul al **
6 REM ** unei sume de inverse **
7 REM ** de produse **
10 INPUT N
20 LET S = 0
30 LET P = 1
40 FOR I = 1 TO N
50 LET P = P * I
60 LET S = S + 1/P
70 NEXT I
80 PRINT S
```

## Divizibilitatea numerelor

Modul de a testa cu calculatorul dacă un număr **A** este divizibil cu un alt număr **B** este următorul: dacă rezultatul împărțirii lui **A** la **B** este un număr întreg atunci **A** este divizibil cu **B**.

În limbajul calculatorului vom testa dacă  $A/B$  este egal cu  $INT (A/B)$ .

Într-adevăr 10 se divide cu 2, deoarece  $10/2$  este egal cu  $INT (10/2)$ , ambele expresii avînd valoarea 5.

Dar 9 nu se divide cu 2, deoarece:

$$9/2 = 4,5 \text{ iar } INT (9/2) = INT (4.5) = 4 \text{ și } 4.5 \neq 4$$

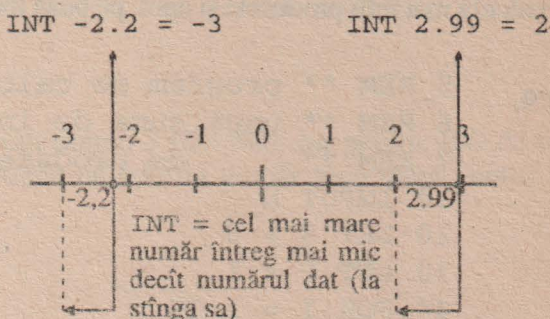
Se observă că pentru testarea divizibilității se folosește funcția **INT** (ÎNTREG). Prin aplicarea ei asupra unei expresii se obține valoarea întregă a numărului rezultat în urma calculării expresiei.

O definiție mai completă a funcției **INT** este însă: **cel mai mare număr întreg mai mic decît numărul dat**. Această definiție se impune deoarece:  $INT (2.99) = 2$  dar  $INT (-2.2) = -3$ .

Se observă, în acest ultim caz, că -3 este cel mai mare număr întreg mai mic decît numărul -2,1 (-2 ar fi fost un număr întreg mai mare decît -2,1 după cum se poate observa din figura 7).

fig.7

Reprezentare grafică  
pe axa numerelor pen-  
tru funcția **INT**  
(ÎNTREG)





Să se facă un program prin care calculatorul să afișeze dacă un număr introdus este sau nu par.

Să presupunem că introducem numărul  $N$ , despre care vrem să știm dacă este par sau nu (10 INPUT N). Apoi aplicăm modul de testare al divizibilității cu 2. Programul P10 va fi:

```
P10.  5 REM ** program de verificare **
      6 REM ** a paritatii unui numar **
      10 INPUT N
      20 IF N/2 = INT (N/2) THEN PRINT N;
        " PAR ": GOTO 10
      30 PRINT N; " IMPAR"
      40 GOTO 10
```

Introducând valoarea 1244 vom obține răspunsul PAR, iar pentru valoarea 1245 vom obține IMPAR.

Pentru a opri programul vom introduce

**STOP (CS + A)** pentru HC

**BREAK (CTRL + C)** pentru PC.



Să se facă un program prin care să se afișeze dacă un număr introdus este divizibil cu 3.

Programul va fi asemănător cu P10. În linia 20, va trebui schimbat 2 cu 3 precum și mesajele. În loc de "PAR" introducem "DIVIZIBIL CU 3", iar în linia 30 în loc de "IMPAR" punem "NU ESTE DIVIZIBIL CU 3".

Acum, introducând valoarea 1244 vom obține răspunsul "NU ESTE DIVIZIBIL CU 3", iar pentru valoarea 1245 vom obține "DIVIZIBIL CU 3".



Care este cel mai mic număr prim mai mare ca 1000 ? Dar ca 10 000 ? Să se facă un program cu care să se stabilească dacă un număr întreg este prim sau nu.

Știm că un număr este *prim* dacă nu are ca divizori decât pe 1 și pe el însuși. Deci, la început vom testa divizibilitatea numărului cu 2. Dacă numărul va fi divizibil cu 2 înseamnă că nu este prim și ne oprim cu mesajul " Nu e prim". Dacă nu este divizibil cu 2 continuăm testarea. Vom testa dacă numărul este divizibil cu 3 și, la fel, dacă este divizibil cu 3 ne oprim (nu e prim) și, dacă nu, continuăm testarea.

Nu vom testa dacă numărul este divizibil cu 4 deoarece dacă nu a fost divizibil cu 2 înseamnă că nu va fi nici cu 4. Practic va trebui să testăm divizibilitatea cu numerele (în continuare) care sînt prime (cele din *Ciurul lui Eratostene*).

Este însă dificil să introducem în prealabil numere care sînt în *ciur* deoarece nu numai că ocupă o parte însemnată din memoria calculatorului dar nici nu știm dinainte cîte numere prime anume să introducem în memorie, deoarece nu știm cît de mare va fi numărul pe care dorim să-l testăm.

Vom proceda astfel: începînd cu 3 vom testa divizibilitatea numărului cu numere impare (deci din 2 în 2: STEP 2). În acest caz chiar dacă vom face și unele operații inutile (de exemplu dacă am testat divizibilitatea cu 3 nu ar mai fi nevoie să o testăm și cu 9) vom asigura o posibilitate pentru calculator de a repeta un anumit tip de operații.

*Pînă unde vom merge cu testarea divizibilității ?*

Evident dacă am ajuns pînă la jumătatea numărului și nu am obținut ca rezultat al testării divizibilității un rezultat întreg (care ar fi trebuit să fie 2), în continuare, nu mai are sens să continuăm testarea deoarece, următorul număr întreg mai mic decât 2 este 1 care nu ne interesează deoarece orice număr întreg este divizibil cu 1. Cu alte cuvinte nu ne interesează să testăm divizibilitatea numărului dat cu numere mai mari decât jumătatea lui, neputînd fi posibil, în acest caz, să se obțină un număr întreg. Deci vom merge cu testarea divizibilității pînă la jumătatea numărului.

Programul P11 este:

```
P11. 5 REM ** program pentru numar prim **
      10 INPUT N
      20 IF N/2 = INT (N/2) THEN PRINT N;
         " NU ESTE PRIM": GO TO 10
      30 FOR I = 3 TO N/2 STEP 2
      40 IF N/I = INT (N/I) THEN PRINT N;
         " NU ESTE PRIM": GO TO 10
      50 NEXT I
      60 PRINT N; " PRIM"
      70 GOTO 10
```

Dacă s-a parcurs tot ciclul dintre liniile 30 și 50 și numărul  $N$  nu a avut ca divizor nici unul din numere, înseamnă că el este prim.

Să aplicăm programul încercând să răspundem la întrebarea *care este cel mai mic număr prim mai mare ca 1 000 ?*

Încercăm 1001: nu e prim, 1 003 nu e prim, 1 007 nu e prim, iar rezultatul corect este 1 009.

Într-un fel programul nu ne mulțumește pe deplin deoarece nu ne spune care a fost divizorul unui număr care nu este prim.

În linia 40 se testează divizibilitatea lui  $N$ . Dacă  $N/I$  este egal cu  $\text{INT}(N/I)$  înseamnă că numărul  $I$  este un divizor al lui  $N$ . Deci, pentru a afișa divizorul, linia 40 va deveni:

```
40 IF N/I = INT (N/I) THEN PRINT N;
   " NU ESTE PRIM. SE DIVIDE CU "; I : GO TO 10
```

Astfel vom afla că 1001 se divide cu 7; 1003 se divide cu 17 iar 1007 cu 19.



Care este cel mai mic număr prim mai mare ca 10 000?

Executând programul observăm că:

- ✗ 10 001 nu este prim; se divide cu 73
- ✗ 10 003 nu este prim; se divide cu 7
- ✗ 10 007 este prim. El este numărul căutat.



Programul astfel realizat ne indică primul divizor pentru un număr dar, deseori, sîntem interesați să găsim toți divizorii primi ai unui număr. Puteți, deci, să realizați un program cu care să se afișeze divizorii primi ai unui număr natural N?

Vom verifica dacă fiecare din numerele 2, 3, 5, 7, ...  $\text{INT}(N/2)$  reprezintă un divizor al lui N, iar dacă *da*, atunci se verifică dacă acest număr este prim sau nu, utilizîndu-se programul deja realizat. Dacă pentru acest număr se găsește un divizor înseamnă că nu este prim și se trece la testarea pentru următorul număr, iar dacă nu se găsește nici un divizor înseamnă că este prim și se afișează. Programul P12 este:

```

P12.  5 REM ** divizorii unui numar **
      10 INPUT N
      15 REM se cauta divizorii numarului N
      20 IF N/2=INT (N/2) THEN PRINT 2; " ";
      30 FOR I = 3 TO INT (N/2) STEP 2
      40 IF N/I = INT (N/I) THEN GO TO 100
      50 NEXT I
      60 STOP
      100 REM se verifica daca divizorul
      101 REM gasit este un numar prim
      110 FOR J = 3 TO INT (I/2) STEP 2
      120 IF I/J = INT (I/J) THEN GO TO 50
      130 NEXT J
      140 PRINT I ; " ";
      150 GO TO 50
  
```

Astfel aflăm că 1001 are ca divizori primi pe 7, 11 și 13; 1003 are ca divizori primi pe 17 și 59; 1007 pe 19 și 53, iar 10001 pe 73 și 137.



Ne punem problema să aflăm câte numere prime sînt mai mici decît un anumit număr  $N$ .

În acest scop se folosește algoritmul pentru determinarea faptului dacă numărul este prim sau nu, iar pentru calcularea numărului de numere prime mai mici decît numărul dat se folosește algoritmul de calcul a unei sume, adăugîndu-se de fiecare dată o unitate la suma precedentă atunci cînd se identifică un număr prim (calculul reușitelor).

Luăm în considerație faptul că 0 și 1 nu sînt numere prime, primul număr prim fiind 2. Vom inițializa în acest caz variabila  $C$  în care se păstrează numărul de numere prime găsit, cu 1 (avem inițial un număr prim și anume, 2) și vom impune să se introducă un număr întreg mai mare sau egal cu 3. De asemenea vom testa toate numerele începînd cu 3 pînă la  $N$  (ne interesează toate numerele prime mai mici sau egale cu numărul  $N$ ) luate din 2 în 2, deoarece numerele prime se găsesc printre cele impare. Programul P13 este:

 P13.

```
5  REM ** cite numere prime **
6  REM ** pina la un numar N **
10 INPUT "Introduceti un numar intreg
    >= 3 "; N
20 LET C = 1
30 FOR I = 3 TO N STEP 2
40 FOR J = 3 TO INT(I / 2) STEP 2
50 IF I / J = INT(I / J) THEN GOTO 80
60 NEXT J
70 LET C = C + 1
80 NEXT I
90 PRINT C; " Numere prime mai mici
    sau egale cu "; N
```

Folosind acest program vom putea observa cum variază numărul de numere prime atunci cînd  $N$  crește sau, cu alte cuvinte, care este distribuția

*numerele prime*. Astfel vom nota că sînt 4 numere prime mai mici ca 10 (deci 40% din numerele pînă la 10 sînt prime), 25 de numere prime mai mici ca 100 (deci 25% din numere pînă la 100), 168 de numere prime mai mici ca 1000 (17%) și 1 229 numere prime mai mici ca 10 000 (12%). Observăm că procentul numerelor prime scade pe măsură ce  $N$  crește sau, cu alte cuvinte, numerele prime sînt mai rare.

De asemenea putem să calculăm *cîte numere prime există într-un interval dat*. În acest scop vom introduce și marginea din stînga a intervalului (presupunem că acesta este un număr par, deoarece, de obicei dorim să aflăm cîte numere prime sînt între 10 și 20, între 100 și 200 etc.):

```
6 INPUT S
```

Inițializăm pe  $C$  cu 0, iar ciclul  $i$  vom începe de la  $S+1$  (trebuie să începem cu un număr impar), linia 30 devenind:

```
30 FOR I = S TO N STEP 2
```

De asemenea linia 90 va deveni:

```
90 PRINT C; "Numere prime între:"; S;  
" si "; N
```

Vom observa că sînt 4 numere prime în intervalul 10, 20; 21 numere prime în intervalul 100, 200 și 135 numere prime în intervalul 1000, 2000.



Să se modifice programul realizat astfel încît să se afișeze și numerele prime găsite sau, cu alte cuvinte, să se afișeze Ciurul lui Eratostene pînă la un anumit număr.

În acest scop în programul P13 inițial, se vor insera liniile 15 pentru afișarea lui 2 (care este primul număr prim) și 65 pentru afișarea numerelor prime  $I$  găsite :

```
15 PRINT 2; " ";  
65 PRINT I; " ";  
85 PRINT
```





Să se facă un program cu care să se calculeze cel mai mare divizor comun (CMMDC) al două numere.

Va trebui mai întâi să introducem cele două numere (**A** și **B**). Apoi ne vom gândi la o metodă (algoritm) cu ajutorul căreia vom putea calcula CMMDC al celor două numere. Vom folosi, de exemplu, *algoritmul lui Euclid* cu care se poate calcula CMMDC a două numere întregi nenule (cu  $A \leq B$ ) prin împărțiri repetate. Cum se procedează? Se împarte **B** prin **A**. Dacă **B** se divide cu **A** (restul este 0) problema este simplă întrucât este evident că CMMDC este chiar **A**, întrucât divide și pe **B** și pe **A** și nu poate exista alt număr mai mare decât **A** care să-l dividă pe **A**. Dacă la împărțirea lui **B** prin **A** restul nu este nul înseamnă că vom obține un cît întreg, să îl notăm cu **C** ( $C = B / A$ ) și un rest **R**. Aplicînd proba împărțirii putem calcula și valoarea lui **R**. Știm că

$$B = A \times C + R, \text{ deci } R = B - A \times C.$$

Dar de această dată în formulă am ținut cont de un cît întreg. Formula corectă este

$$R = B - A \times \text{INT } C.$$

Algoritmul lui Euclid ne indică apoi să înlocuim în împărțire pe **B** cu **A**, iar pe **A** cu **R** și să continuăm procedeul pînă cînd obținem un rest 0. În acest caz *CMMCD va fi ultimul rest nenul*.

Să exemplificăm funcționarea algoritmului pentru numerele 72 și 15. Am văzut că primul paș este împărțirea numărului mare (**B**) la numărul mic (**A**). În cazul nostru obținem cîtul 4 și restul 12. Mai departe, îl înlocuim pe **B** cu **A**, adică pe 72 cu 15 și pe **A** cu restul **R** adică pe 15 cu 12. Vom repeta procedeul împărțind pe 15 la 12 și obținem cîtul 1 și restul 3. Iarășî înlocuim pe **B** cu **A**, adică pe 15 cu 12 și pe **A** cu **R**, adică pe 12 cu 3, repetînd împărțirea. De data aceasta (12/3) obținem cîtul 4 și restul 0. Înseamnă că CMMDC este ultimul rest nenul, iar acesta a fost 3 pe care îl regăsim în valoarea actuală a lui **A**. Într-adevăr, putem verifica acest lucru

prin descompunerea numerelor inițiale în factori primi și aflarea CMMDC:

$$\begin{array}{r} 72 = 2^3 \times 3^2 \\ 15 = 3 \times 5 \\ \hline \text{CMMDC} = 3 \end{array}$$

Programul P14 va fi :

```
❓ P14. 5 REM ** program de calcul al **
        6 REM ** CMMDC al doua numere **
        10 INPUT A
        20 INPUT B
        30 LET C = B/A
        40 LET R = B - A * INT (C)
        50 IF R = 0 THEN PRINT "CMMDC="; A : STOP
        60 LET B = A
        70 LET A = R
        80 GOTO 30
```



Putem calcula cu ajutorul programului realizat și cel mai mic multiplu comun (CMMMC) al numerelor?

Da, cunoscînd valorile inițiale ale numerelor și CMMDC, totdeauna CMMMC va rezulta din împărțirea produsului numerelor inițiale la CMMDC. În acest scop va trebui să memorăm (salvăm) valorile inițiale ale numerelor A și B în alte două variabile, să zicem P și N, și, în final, să adăugăm în linia 50 o instrucțiune cu care se va calcula CMMMC. Deci adăugăm în programul P14:

```
25 LET P = B
27 LET N = A
50 IF R = 0 THEN PRINT "CMMDC="; A
51 PRINT "CMMMC="; P * N/A: STOP
```



Folosind tot o regulă (algoritm) de împărțire repetată să se arate cum se poate realiza descompunerea în factori primi a unui număr.

Ne propunem să găsim factorii primi ai unui număr întreg pozitiv. Știm că orice număr întreg pozitiv  $N$  poate fi exprimat ca produs de numere prime ridicate la puteri naturale (ultimele le vom numi **indici**). Indicele unui număr prim reprezintă puterea la care trebuie ridicat numărul pentru descompunerea în factori primi ai lui  $N$ . De exemplu:

pentru  $N = 360$

$$360 = 2^3 \times 3^2 \times 5 \text{ (indicii sînt 3, 2 și 1).}$$

Pentru a descompune în factori primi numărul  $N$  se va utiliza tot o metodă de împărțire repetată. Primul factor posibil este 2, deci putem scrie:

$$360 = 2 \times 180 = 2 \times 2 \times 2 \times 45 = 2^3 \times 45$$

Astfel, 2 este un factor al lui 360 și a lui 180 și al lui 90, dar nu este un factor al lui 45. Altfel spus împărțind pe 360 la 2 și pe 180 la 2 și pe 90 la 2 vom obține restul nul, adică:

$$N - 2 * \text{INT} (N / 2) = 0$$

În schimb dacă  $N = 45$  atunci  $N - 2 * \text{INT} (N / 2) \neq 0$

Vom încerca în continuare pe 3 :

$$360 = 2 \times 2 \times 2 \times 3 \times 15 = 2 \times 2 \times 2 \times 3 \times 3 \times 5 = 2^3 \times 3^2 \times 5$$

După cum se observă aplicarea metodei (algoritmului) se face astfel: se împarte succesiv  $N$  la divizorii  $D$  care sînt numere prime: (2, 3, 5, 7 etc). Dacă  $D$  divide pe  $N$  atunci se caută întregul  $I$  cel mai mare, astfel încît  $D^I$  să dividă pe  $N$ . În acest caz  $N$  fiind înlocuit în vederea împărțirii următoare prin  $N / D^I$ . Calculul se oprește cînd produsul tuturor factorilor deja găsiți este egal cu  $N$ .

Realizarea acestui program este ceva mai dificilă. Va trebui să memorăm mai multe valori ale variabilelor. De exemplu, va trebui să memorăm numărul

inițial  $N$ : LET  $A = N$ , deoarece în final avem nevoie de valoarea lui atât pentru afișare cât și pentru găsirea divizorilor care nu vor depăși ca valoare jumătate din numărul inițial. De asemenea va trebui să inițializăm cu 1 variabila  $K$  reprezentând valoarea produsului tuturor divizorilor găsiți la puterile respective și cu 0 puterea curentă (LET  $K = 1$  și LET  $E = 0$ ). Vom testa dacă 2 este divizor și apoi 3, 5, 7 etc pînă la jumătatea numărului (ca la testarea numărului prim). După alegerea fiecărui divizor potențial  $D$  vom apela o subrutină prin care cercetăm dacă  $D$  este divizor calculînd și puterea sa. Dacă numărul  $A$  se divide prin  $D$  - adică dacă  $A/D = \text{INT}(A/D)$  - atunci puterea lui  $D$  va crește cu o unitate (LET  $E = E+1$ ), valoarea lui  $K$  se va multiplica cu  $D$ , iar numărul se va înlocui cu valoarea  $A/D$  pe care am notat-o cu  $Q$ , repetîndu-se secvența cu noul număr. Dacă  $D$  nu este divizor, atunci se afișează divizorul găsit la puterea respectivă. Folosim o subrutină pentru a nu repeta de două ori aceeași secvență de instrucțiuni, atât în cazul divizorului 2 cât și în cazul unui divizor impar.

Partea de program de la liniile 90 la 180 realizează acest lucru și reprezintă practic un program de sine stătător (subprogram) în cadrul programului principal. De aceea, se numește **subrutină**. Ea se apelează cu instrucțiunea GOSUB (liniile 50 și 70), iar cînd ajunge la sfîrșit, instrucțiunea RETURN (linia 180) redă controlul programului la prima linie după instrucțiunea GOSUB care a apelat subrutina. Cînd  $K=N$ , adică atunci cînd produsul tuturor divizorilor găsiți la puterile respective este chiar  $N$ , programul se termină, deoarece nu mai putem găsi nici un alt factor prim. Programul P15 este:

```

P15. 5 REM ** program pentru descompunerea **
      6 REM ** unui numar in factori primi **
      10 INPUT N
      15 PRINT N; " = ";
      20 LET A = N
      30 LET K = 1
      40 LET D = 2
      50 GOSUB 90
      60 FOR D = 3 TO INT (N/2) STEP 2
      70 GOSUB 90
      80 NEXT D
      90 LET E = 0
     100 LET Q = INT (A/D)
     110 IF A/D <> Q THEN GOTO 160

```

```

120 LET E = E + 1
130 LET K = K * D
140 LET A = Q
150 GOTO 100
160 IF E <> 0 THEN PRINT D; "^"; E; "*";
170 IF K = N THEN STOP
180 RETURN

```

## Radicali

---

Știm că **pătratul** unui număr reprezintă numărul înmulțit cu el însuși. De ce se numește pătrat al numărului? Deoarece reprezintă *aria unui pătrat a cărui latură este chiar numărul*. În mod similar **cubul** unui număr reprezintă numărul înmulțit cu el însuși de 3 ori (la puterea a 3-a), deoarece reprezintă *volumul unui cub a cărei muchie este egală cu numărul*.

Se pune acum problema invers, adică, dându-se un număr, să se găsească altul care înmulțit cu el însuși să ne conducă la primul număr (*operația radical*). Sigur că, având niște numere pătrate, ne va fi ușor să identificăm numerele care, înmulțite cu ele însele, ne vor da ca rezultate pătratele respective. De exemplu, având 4, 9, 16, 25, 36 ... știm că  $2 \times 2 = 4$ ;  $3 \times 3 = 9$ ;  $4 \times 4 = 16$ ;  $5 \times 5 = 25$  și așa mai departe, cu alte cuvinte, numerele corespunzătoare vor fi 2, 3, 4, 5, 6 ... Deci având numere pătrate este relativ simplu.

*Dar cum putem găsi radicalul unui număr care nu este un pătrat perfect?*

În acest caz operația este mai dificilă, deoarece rezultatul nemaifiind un număr natural, vom aproxima rezultatul prin numere rafinate (exprimate zecimal).

Putem însă să ne ajutăm de calculator.



Problema care se pune, deci, este să realizăm un program cu ajutorul căruia să găsim valoarea aproximativă a radicalului unui număr pozitiv.

Vom realiza un program cu ajutorul căruia vom putea calcula mai multe valori aproximative ale căror pătrate să fie din ce în ce mai apropiate de numărul dat. Metoda aceasta se numește **metoda de rezolvare prin aproximații succesive**.

În primul rînd vom introduce numărul din care dorim să extragem radicalul (10 INPUT N). Apoi vom introduce o încercare a noastră, adică numărul care credem noi că înmulțit cu el însuși ne va da numărul inițial N (20 INPUT A). Pentru a vedea cît de reușită este încercarea noastră vom afișa valoarea încercării (A), rezultatul pe care îl obținem cu încercarea respectivă și numărul inițial N. Desigur, cu cît diferența dintre ultimele două numere este mai mică, cu atît încercarea a fost mai bună, iar dacă cele două numere sînt egale atunci înseamnă că încercarea respectivă reprezintă exact radicalul numărului. Vom introduce diverse încercări (repeșind procedeul: GO TO 20) în funcție de diferența observată între cele două numere.

Programul P16 este:

```
P16.  5 REM ** program de calcul al unui **
      6 REM ** radical prin incercari **
      10 INPUT N
      20 INPUT A
      30 PRINT A; " "; A * A; " "; N
      40 GOTO 20
```

Să aplicăm programul încercînd să găsim valoarea radicalului din numărul 10.

Știm că  $3 \times 3 = 9$  iar  $4 \times 4 = 16$ , deci numărul pe care îl căutăm va fi între 3 și 4. Să încercăm 3,5. Vom obține rezultatul 12,25 care este mai mare decît 10, deci, vom încerca o valoare mai mică: 3,4. Cu ea obținem rezultatul 11,56 care este tot mai mare decît 10, deci, vom încerca o valoare mai mică: 3,3. Cu ea obținem rezultatul 10,89 iar cu următoarea (3,2) obținem 10,24 care este tot mai mare ca 10. În continuare vom încerca o valoare mai mică (3,1) cu care obținem rezultatul 9,61. Acesta este însă mai mic decît 10. Rezultă că valoarea pe care o căutăm este situată între 3,1 și 3,2. Înseamnă că trebuie să căutăm o valoare cu 2 zecimale. Să încercăm valoarea 3,15. Obținem rezultatul 9,9225 care este mai mic ca 10, deci, vom încerca o valoare mai mare. Cu 3,16 obținem 9,9856 care este tot mai mică. Însă cu 3,17 obținem 10,0489 care este mai mare. Deci, valoarea căutată este între 3,16 și 3,17. Înseamnă că trebuie să căutăm o

valoare cu 3 cifre zecimale. Continuând în același mod raționamentul redăm rezultatele care se vor afișa pe ecran, măbind sau micșorând valoarea aproximației în funcție de relația de inegalitate relativă la 10 (pătratul este mai mic sau mai mare ca 10):

3,165	10,017225	10
3,164	10,010896	10
3,163	10,004569	10
3,162	9,998244	10

Deci valoarea căutată este între 3,162 și 3,163. Încercăm valoarea 3,1625;

3,1625	10,001406	10
3,1624	10,000774	10
3,1623	10,000141	10
3,1622	9,9995088	10

deci valoarea căutată este între 3,1622 și 3,1623. Să încercăm valoarea 3,16225:

3,16225	9,9998251	10
3,16226	9,9998883	10
3,16227	9,9999515	10
3,16228	10,000015	10

Deci valoarea căutată este între 3,16227 și 3,16228. Să încercăm valoarea 3,162275:

3,162275	9,9999832	10
3,162276	9,9999895	10
3,162277	9,9999958	10
3,162278	10,000002	10

Valoarea căutată este între 3,162277 și 3,162278. Să încercăm valoarea 3,1622775.

3,1622775	9,999999	10
3,1622776	9,9999996	10
3,1622777	10	10

Deci valoarea căutată este chiar 3,1622777. Aceeași valoare o putem obține (mult mai simplu) dacă am fi utilizat *funcția pentru radical* (SQR):

```
PRINT SQR 10
```

se va afișa valoarea 3,1622777.

De fapt, chiar și această valoare reprezintă o aproximație, calculatorul neputînd calcula o aproximație mai fină.



### Rapid și precis - cîteva probleme pentru dumneavoastră

---



Să se găsească perechile de numere a căror sumă este 1000, primul să fie divizibil cu 17 iar al doilea cu 19.

Pentru rezolvarea acestei probleme putem profita de viteza calculatorului și astfel să-l punem să testeze toate perechile de numere a căror sumă este 1000. Prima pereche va fi 1 și 999, a doua 2 și 998 și așa mai departe. Astfel cu un contor I care ia valori de la 1 la 999, primul număr din pereche va fi chiar I, iar al doilea 1000-I. Programul este P17:



```
P17. 5 REM ** program pentru rezolvarea **  
6 REM ** problemei cu numere **  
10 FOR I = 1 TO 999  
20 IF I/17=INT (I/17) AND  
 (1000-I)/19=INT ((1000-I)/19)  
 THEN PRINT I,1000-I  
30 NEXT I
```

Perechile de numere care întrunesc condițiile și care vor fi, deci, afișate sînt : 221 și 779, 544 și 456, 867 și 133.





Să se găsească perechile de numere a căror sumă este 1000, primul număr să fie divizibil cu 17 sau cu 13 iar al doilea cu 19 și cu 7.

Folosind același raționament programul va fi asemănător, linia 20 a programului 17 modificându-se conform noilor condiții:

```
20 IF (I/17=INT (I/17) OR I/13=INT (I/13))
   AND ((1000-I)/19=INT ((1000-I)/19)
   AND (1000-I)/7=INT ((1000-I)/7))
   THEN PRINT I , 1000-I
```



Să se găsească numărul  $\overline{ABC}$  pentru care  $A^2 + B^2 + C^2 = A + B + C$

Cifra A, fiind cea mai semnificativă, poate lua valori de la 1 la 9, în timp ce, cifrele B și C pot lua valori de la 0 la 9. Vom lua pe rând toate valorile posibile ale cifrelor A, B și C și vom testa care combinație îndeplinește condiția pusă. Programul P18 este:



```
P18. 5 REM ** program pentru gasirea **
      6 REM ** valorilor care satisfac **
      7 REM ** o identitate **
      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
```

Se obțin următoarele numere: 100, 101, 110 și 111. Prima cifră reprezintă valoarea lui A, a doua valoarea lui B, iar a 3-a valoarea lui C.



Să se genereze toate numerele de 4 cifre de forma  $\overline{3A2B}$  care se divid cu 9.

După aceeași metodă A și B pot lua valori de la 0 la 9. Vom lua toate combinațiile posibile iar calculatorul va testa pentru care se îndeplinește condiția divizibilității cu 9, și anume, când suma cifrelor numărului (adică  $3+A+2+B$ ) va reprezenta un număr divizibil cu 9. Programul P19 este:

```
P19. 5 REM ** program pentru gasirea **
      6 REM ** valorilor care satisfac **
      7 REM ** o conditie **
      10 FOR A = 0 TO 9
      20 FOR B = 0 TO 9
      30 IF (3+A+2+B)/9 = INT ((3+A+2+B)/9)
          THEN PRINT 3;A;2;B
      40 NEXT B
      50 NEXT A
```

Se obțin 11 soluții, prima fiind 3024, iar ultima 3924.



Să se găsească perechile de cifre A și B pentru care numărul  $\overline{7AB3}$  să fie divizibil cu 7 și cu 3.

De data aceasta pentru divizibilitatea cu 7 nu mai avem nici o regulă, de aceea, va trebui să testăm dacă numărul (care este egal cu  $7 \times 10^3 + A \times 10^2 + B \times 10 + 3$ ) este divizibil cu 7 după regula cunoscută.

Programul P20 este:



P20.

```
5 REM ** program de gasire a unor **
6 REM **   valori care satisfac   **
7 REM **           o conditie     **
10 FOR A = 0 TO 9
20 FOR B = 0 TO 9
30 IF (7*1000+A*100+B*10+3)/7=
    INT((7*1000+A*100+B*10+3)/7) AND
    (7+A+B+3)/3=INT((7+A+B+3)/3)
    THEN PRINT A,B
40 NEXT B
50 NEXT A
```

Vom obține ca rezultat perechile 2 și 0, 4 și 1, 6 și 2, 8 și 3. Deci numerele 7203, 7413, 7623 și 7833 sînt divizibile cu 7 și cu 3.



## O problemă privind grafica în mișcare

---



Să se facă un program cît mai scurt cu ajutorul căruia să se miște pe ecran un punct în 4 direcții (sus, jos, stînga, dreapta) și apoi un caracter grafic lăsînd urme în mișcarea sa.

Să presupunem că vrem să mișcăm punctul sau caracterul grafic care la început se află în centrul ecranului cu următoarele taste: 6 sau Q pentru sus; 7 sau A pentru jos; 8 sau P pentru dreapta și 5 sau O pentru stînga. Cel mai scurt program se va realiza nu prin mai multe linii IF ci prin calculul coordonatelor în funcție de tastele acționate folosind propozițiile logice.

De exemplu, pentru calculul coordonatei pe orizontală:

$$\text{LET } X = X + (\text{INKEY\$} = "8") - (\text{INKEY\$} = "5")$$

Dacă se acționează tasta 8 atunci  $\text{INKEY\$} = "8"$  va fi o propoziție adevărată și va lua valoarea 1, în timp ce  $\text{INKEY\$} = "5"$  este o propoziție falsă luînd valoarea 0. În concluzie noua coordonată va fi egală cu vechea coordonată la care se va adăuga o unitate (deci punctul se deplasează spre

dreapta). Dacă în schimb se acționează tasta 5 atunci  $INKEY\$ = "8"$  va lua valoarea 0 iar  $INKEY\$ = "5"$  valoarea 1 și, deci, valoarea coordonatei pe orizontală va scădea cu o unitate, asigurându-se deplasarea spre stînga. Programul P21 va asigura mișcarea punctului:

- ✗ la dreapta atunci cînd se acționează tasta 8 sau tasta P
- ✗ la stînga cînd se acționează tasta 5 sau O
- ✗ în sus cînd se acționează tasta 6 sau Q
- ✗ în jos cînd se acționează tasta 7 sau tasta A

```

P21.A 5 REM ** program pentru miscarea **
      6 REM **   unui punct pe ecran   **
      7 REM **   pentru calculator HC   **
      10 LET X = 255/2
      20 LET Y = 175/2 .
      30 PLOT X , Y
      40 LET X = X + (INKEY$ = "8" OR
                    INKEY$ = "p") - (INKEY$ = "5" OR
                    INKEY$ = "o")
      50 LET Y = Y + (INKEY$ = "6" OR
                    INKEY$ = "q") - (INKEY$ = "7" OR
                    INKEY$ = "a")
      60 GOTO 30
```

Totuși veți observa că, după ce punctul a atins deja marginea din stînga a ecranului, atunci cînd veți comanda mișcarea la stînga, punctul se va deplasa spre dreapta ! De asemenea, după ce punctul a atins deja o dată marginea de jos ecranului, atunci cînd veți comanda mișcarea în jos, punctul se va deplasa în sus ! Puteti explica această anomalie ?

Explicația este următoarea: cînd se aplică instrucțiunea de afișare a unui punct pe ecran  $PLOT X, Y$ , limbajul BASIC calculează automat valoarea absolută a valorilor coordonatelor X și Y (practic face  $PLOT ABS X, ABS Y$ ). În acest caz atunci cînd valoarea coordonatei pe orizontală (X), scade sub valoarea 0, atingînd valorile -1, -2, -3 ... punctul se va desena pe coordonatele pe orizontală cu valorile 1, 2, 3 ... realizîndu-se practic mișcarea la dreapta în loc de stînga. Pentru mișcarea în jos și în sus fenomenul este similar. Însă atunci cînd vom comanda mișcarea punctului la

dreapta sau în sus, atunci cînd coordonatele pe orizontală (X) și, respectiv, pe verticală (Y), vor depăși valorile marginii ecranului (255 și, respectiv, 175), atunci, punctul va ieși din ecran, iar programul se va întrerupe cu un mesaj eroare.

Desigur în acest caz, vor trebui adăugate și liniile care să asigure faptul că punctul nu va ieși din ecran. Deoarece calculatorul consideră parametrii după PLOT în valoare absolută, condiția ieșirii din ecran se va pune numai pentru partea dreaptă și de sus a ecranului ( $X > 255$  și  $Y > 175$ ) în timp ce, după cum am văzut acționînd în continuare după atingerea marginilor tastele pentru ieșirea prin partea stîngă și respectiv jos a ecranului, punctul se va depărta de aceste margini deoarece noile coordonate, deși sînt negative, ele sînt luate în considerare în valoare absolută, ceea ce face ca pînă în momentul în care sînt atinse valorile -255 și respectiv -175 punctele să poată fi afișate pe ecran.

Programul similar pentru calculatoare PC este P21.B.

```
P21.B. 5 REM ** program pentru miscarea unui **
7 REM **          punct pe ecran          **
8 REM **          pentru calculator PC     **
9 CLS
10 SCREEN 1
15 LET X = 320/2
20 LET Y = 200/2
30 PRESET (X,Y),7
40 LET X = X + (INKEY$ = "8" OR
INKEY$ = "p") - (INKEY$ = "5" OR
INKEY$ = "o")
50 LET Y = Y + (INKEY$ = "6" OR
INKEY$ = "q") - (INKEY$ = "7" OR
INKEY$ = "a")
60 GOTO 30
```

Se observă că s-a ales rezoluția medie în modul grafic ( $320 \times 200$  puncte). Originea este acum colțul din stînga sus al ecranului, iar punctul se desenează în culoarea albă (cod 7). Mișcarea punctului se va realiza invers ca la HC (stînga se transformă în dreapta, iar sus în jos) deoarece condiția de adevărat are valoarea logică -1.

Se observă că la atingerea marginilor ecranului nu se semnalează o croare, mișcarea continuându-se în afara marginilor.

Pentru mișcarea unui caracter grafic (să zicem cel obținut cu 8 și CS în modul grafic) programul va fi similar, obținându-se programul P22.A. pentru HC și P22.B pentru PC:

```
P22.A 5 REM ** program pentru miscarea **
      6 REM **   caracter pe ecran   **
      7 REM **   pentru calculator HC **
      10 LET X = 11
      20 LET Y = 16
      30 PRINT AT X , Y; "■"
      40 LET X = X + (INKEY$ = "6" OR
                    INKEY$ = "q") - (INKEY$ = "7" OR
                    INKEY$ = "a")
      50 LET Y = Y + (INKEY$ = "8" OR
                    INKEY$ = "p") - (INKEY$ = "5" OR
                    INKEY$ = "o")
      60 GOTO 30
```

Aceași remarcă făcută pentru programul scris pentru grafica de rezoluție înaltă este valabilă și pentru programul actual scris pentru grafica de rezoluție scăzută. În acest caz X va reprezenta numărul liniei, iar Y numărul coloanei. Se observă că atunci când X crește, practic, numărul liniei va crește, iar caracterul grafic se va deplasa în jos.

Programul similar pentru calculatoare PC este P22.B

```
P22.B 5 REM ** program pentru miscarea **
      6 REM **   unui caracter pe ecran   **
      7 REM **   pentru calculator PC   **
      8 CLS
      9 SCREEN 0
     10 LET X = 25/2
     20 LET Y = 20
     30 LOCATE X,Y: PRINT "◆"
```

```

40 LET X = X + (INKEY$ = "6" OR
  INKEY$ = "q") - (INKEY$ = "7" OR
  INKEY$ = "a")
50 LET Y = Y + (INKEY$ = "8" OR
  INKEY$ = "p") - (INKEY$ = "5" OR
  INKEY$ = "o")
60 GOTO 30

```

Se observă că s-a ales modul text la lăţimea de 40 de caractere (SCREEN 0). De asemenea s-a ales caracterul grafic **caro** obţinut cu tastele **CTRL** şi **D**. Ca şi în cazul precedent mişcarea se va realiza invers deoarece condiţia de adevărat se evaluează cu -1.

Ieşirea caracterului din cadrul ecranului are însă ca urmare întreruperea programului şi afişarea mesajului de eroare "**Illegal function call in 30**".

Se observă că în cadrul propoziţiilor logice care folosesc funcţia INKEY\$ se pot folosi numai operatorii logici OR care asigură posibilitatea alegerii unei singure taste pentru realizarea unei deplasări.

Operatorul logic AND nu se poate folosi în conjuncţie cu funcţia INKEY\$ deoarece aceasta nu funcţionează când se acţionează deodată mai multe taste. În acest scop la calculatoarele HC se poate folosi funcţia IN.

Aceste programe pot asigura şi o *animaţie* cu condiţia ca punctul (şi respectiv caracterul grafic) să fie şters de pe vechea poziţie înainte de punerea sa pe următoarea poziţie.

De asemenea în acest caz este necesar să se găsească şi o viteză adecvată prin inserarea între două afişări consecutive a unei mici pauze (PAUSE) sau a unei note muzicale de o anumită durată (de obicei mică) pentru a da senzaţia unei mişcări reale.



## Să regăsim "proporţia de aur"

---

Ştim că secvenţa de numere 0, 1, 1, 2, 3, 5, 8, 13, ... se numeşte *şirul lui Fibonacci*. Fiecare *nou termen* (număr) se calculează prin suma ultimelor două numere  $0+1=1$ ;  $1+1=2$ ;  $1+2=3$ ;  $2+3=5$ ;  $5+8=13$  şi așa mai departe.



Realizați un program cu ajutorul căruia să se calculeze primele  $N$  numere din șirul lui Fibonacci. Calculați și rezultatul împărțirii unui număr din șir la următorul. Ce puteți spune în legătură cu rezultatul obținut?

Va trebui să indicăm pînă la ce termen din șir vom calcula iar apoi să rezervăm un spațiu de memorie pentru memorarea acestor numere (20 DIM A (N)). Apoi vom introduce valorile primilor doi termeni din șir (0 și 1) iar următorii se vor calcula într-un ciclu de la 3 la  $N$  (adică de la al 3-lea termen pînă la al  $N$ -lea) după formula cunoscută. După acest calcul vom afișa valoarea contorului  $I$  (pentru a ști al cîtelea termen din șir este) apoi termenul respectiv și rația care îi corespunde (cîtul dintre termenul respectiv și cel curent). Programul P23 este:

```
P23. 5 REM ** Fibonacci 1 **
      10 INPUT N
      20 DIM A(N)
      30 LET A(1) = 0
      40 LET A(2) = 1
      50 FOR I = 3 TO N
      60 LET A(I) = A(I-1) + A(I-2)
      70 PRINT I ; " "; A(I) ; " "; A(I-1)/A(I)
      80 NEXT I
```

Din cele afișate pe ecran la rularea programului, vom observa, de exemplu, că al 21-lea termen din șirul lui Fibonacci este 6765 iar rația corespunzătoare este 0,618034. În legătură cu rația observăm că ia valori diferite în jurul lui 0,6 pentru primii termeni ai șirului, apoi valori foarte apropiate de 0,618034 pentru termenii situați după termenul 20, iar începînd cu termenul 23 se menține constantă la valoarea 0,61803399. Această rație se mai numește și **rația (sau punctul) de aur**, ea reprezentînd proporția cea mai naturală (s-a constatat că în natură dimensiunile sînt deseori bazate pe proporția de aur), dar și cea mai uzitată de oameni în construcții, în arhitectură, în artă etc. Prin analize s-a descoperit că în marile capodopere ale lumii (sculpturi, picturi, creații cinematografice) se respectă riguros proporția de aur.



*Cum s-a descoperit acest șir de numere și respectiv proporție?*

Pentru a afla câte perechi de iepuri de casă se nasc într-un an dintr-o singură pereche de iepuri, cineva a așezat o pereche de iepuri într-un loc îngrădit, știind că după o lună o pereche de iepuri aduce pe lume altă pereche. Numai prima pereche va avea descendenți și luna următoare, astfel încât în luna a doua vor fi trei perechi, iar dintre acestea în luna următoare două vor avea descendenți, astfel încât în luna a treia numărul de perechi de iepuri în această lună este de cinci. Dintre acestea în aceeași lună, vor avea urmași trei perechi, iar numărul de perechi în luna a patra va fi de opt. Începeți să recunoașteți șirul de numere?

Problema a fost formulată prin anul 1202, iar *Leonardo Fibonacci* (matematician italian, 1170-1250) a găsit *legea numerică* prin care se exprimă o însușire a materiei vii, și anume sub forma șirului de numere întregi pe care îl cunoașteți și care mai poartă denumirea de *legea creșterilor organice*, deoarece prin el se exprimă dezvoltarea materiei vii, realizată prin compuneri care se însumează succesiv.

Dintre exemple menționăm: distanțele dintre nodurile de creștere ale unei tulpini; dezvoltarea cochiliilor melcilor sau scoicilor; alungirea oaselor și a coarnelor animalelor etc.



**Modificați programul de calcul al termenilor și rației pentru șirul lui Fibonacci astfel încât să începeți secvența cu alte două numere dar folosind aceeași regulă pentru obținerea fiecărui termen nou. Se modifică rația?**

În acest scop se modifică liniile 30 și 40 din programul P23. De exemplu:

$$30 \text{ LET } A(1) = 5$$

$$40 \text{ LET } A(2) = 7$$

Se observă că termenii șirului vor fi diferiți dar rația este aceeași. În acest caz rația se stabilizează la valoarea cunoscută chiar mai repede (de la al 20-lea termen). Se poate încerca și cazul în care al doilea termen al șirului este mai mic decât primul termen. De exemplu:

$$30 \text{ LET } A(1) = 10$$

$$40 \text{ LET } A(2) = 2$$

Si în acest caz se poate observa că rația va fi aceeași, ea stabilindu-se de la al 23-lea termen al șirului ca în cazul inițial.



**Pentru calculul punctului de aur verificați următoarea formulă de calcul :**

$$r = \frac{1 + \sqrt{5}}{2}$$

Cu

```
PRINT (1 + SQR 5) / 2
```

obținem rezultatul 1,618034 care rezultă din împărțirea a doi termeni consecutivi :  $A(I) / A(I-1)$ .



**Modificați programul pentru calculul termenilor și rației din șirul lui Fibonacci astfel încât să puteți calcula doar al N-lea termen din șir și rația corespunzătoare.**

Se scoate linia 70 din programul P23 și se pune după NEXT în următoarea formă:

```
90 PRINT N; " " ; A(N) ; " " ; A(N-1) / A(N)
```

Această modificare devine utilă atunci când se solicită al N-lea termen din șirul lui Fibonacci și rația corespunzătoare.

Utilizând o variabilă multidimensională -  $A(N)$  - pentru calculul termenilor din șirul lui Fibonacci programul "merge" bine, dar este mare consumator de memorie atunci când vrem să calculăm termeni ai șirului de rang mare (de exemplu pentru  $N > 1000$ ). De la un anumit rang în sus, acest lucru nici nu mai este posibil, calculatorul neavând suficientă memorie.



**Realizați un program pentru șirul lui Fibonacci fără a utiliza o variabilă multidimensională.**

Pornim de la două valori inițiale  $X1$  și  $X2$ . În orice situație următorul termen din șir va fi  $X3=X1+X2$ . Apoi punem valoarea lui  $X2$  în  $X1$  și valoarea lui  $X3$  în  $X2$  și îl calculăm de fiecare dată după aceeași formulă pe  $X3$ . Repetăm operația de câte ori dorim. Programul P24 va fi:

```
P24. 5 REM ** Fibonacci 2 **
      10 LET X1 = 1
      20 LET X2 = 2
      30 PRINT X1 ; " "; X2 ; " "; X1/X2
      40 LET X3 = X2 + X1
      50 PRINT X3 ; " "; X2/X3
      60 LET X1 = X2
      70 LET X2 = X3
      80 GOTO 40
```

Pentru o reprezentare mai plastică putem da următoarele denumiri variabilelor:

$X1 = \text{BUNICUL}$   
 $X2 = \text{TATĂL}$   
 $X3 = \text{FIUL}$

Se observă că de fiecare dată FIUL, adică următorul termen din șir (de fapt următorul membru al familiei) va rezulta din  $X2$  și  $X1$  iar pentru următoarea generație locul BUNICULUI va fi luat de actualul TATĂ, iar locul TATĂLUI va fi luat de actualul FIU.



Să se facă un program cu care să se realizeze operațiile de reuniune, intersecție și scădere a două mulțimi.

Stim că în urma operației de *reuniune* a două mulțimi rezultă o mulțime care conține elementele din prima și din a doua mulțime, comune și necomune, luate o singură dată. De asemenea prin intersecție rezultă o mulțime care conține numai elementele comune, iar prin *scădere* o mulțime care conține toate elementele din prima mulțime dar care nu sînt în a doua.

Să notăm cele două mulțimi cu A și B. Vom parcurge următorii pași logici:

- ⇒ 1. **Inițializare.** Va trebui să știm câte elemente conține prima și a doua mulțime, valori pe care le vom atașa variabilei N (pentru prima mulțime) și respectiv M (pentru a doua mulțime). Vom dimensiona câte un vector pentru elementele mulțimii A și elementele mulțimii B.
- ⇒ 2. **Introducerea valorilor** A(I) pentru mulțimea A cu un ciclu FOR-NEXT precum și a valorilor B(I) pentru mulțimea B cu un alt ciclu FOR-NEXT
  - ⇒ 2.1. Deoarece o mulțime nu poate conține același element de mai multe ori, în ciclul de introducere (atît pentru mulțimea A cît și pentru mulțimea B) vom începe imediat după introducerea valorii un alt ciclu (imbricat) în scopul testării repetării valorii. Dacă aceeași valoare a mai fost introdusă atunci introducerea este respinsă solicitîndu-se altă introducere. De remarcat că indicele ciclului interior va lua valori de la 1 pînă la I-1 deoarece valoarea recent introdusă A(I) se compară cu valorile introduse pînă la acel moment, adică cu valorile A(1), A(2), ... , A(I-1). Dacă valoarea A(I) este diferită de valoarea A(K) , atunci verificarea se continuă, iar dacă se găsește o valoare egală se afișează

mesajul de eroare și se repetă introducerea. La fel se procedează și pentru ciclul de introducere a valorilor pentru mulțimea B.

⇒ 3. **Afișarea valorilor** mulțimii A și mulțimii B. Afișarea se va face pe un rând (valorile în continuare cu un spațiu între ele) iar, după terminarea afișării valorilor unei mulțimi, se va introduce o linie PRINT pentru trecerea la rândul următor.

⇒ 4. **Determinarea mulțimii diferență** A-B. În acest scop vom lua pe rând toate cele N elemente A(I) ale mulțimii A și le vom compara cu fiecare din elementele B(J) ale mulțimii B. Dacă, de exemplu, un element A(I) nu este egal cu nici unul din elementele B(J), adică dacă se parcurge tot ciclul J, atunci se afișează elementul A(I) în continuare, deoarece el aparține mulțimii A și nu aparține mulțimii B. Dacă însă se găsește că elementul A(I) este egal cu un element B(J), atunci el se dă la o parte (nu se afișează, neaparținând mulțimii A-B) și se trece la verificarea apartenenței la mulțimea A-B a următorului element din mulțimea A (NEXT I).

⇒ 5. **Determinarea mulțimii reuniune**  $A \cup B$ . În acest scop se vor afișa în primul rând toate elementele care aparțin mulțimii A. Deci:

⇒ 5.1. Se afișează toate elementele mulțimii A;

⇒ 5.2. Se iau pe rând elementele mulțimii B și se compară cu fiecare element din mulțimea A. Dacă se găsește o egalitate atunci nu se mai afișează și acest element deoarece el a mai fost odată afișat aparținând lui A (un element nu trebuie să apară de mai multe ori în reuniune) și se trece la analizarea următorului element din B (NEXT J). Dacă după compararea cu toate elementele din A nu se găsește nici o egalitate atunci elementul din B se afișează în continuare, el făcând parte din reuniune.

⇒ 6. **Determinarea mulțimii intersecție**  $A \cap B$ . Se iau pe rând toate elementele din A și se compară cu fiecare element din B. Dacă un element se găsește și în A și în B atunci el se afișează

deoarece face parte din intersecție. Apoi se trece la compararea următorului element din A cu fiecare din cele aparținând lui B (NEXT I). Programul P25 este:

```
P25.  5 REM **      program pentru calculul      **
      6 REM **      diferentei, reuniunii si      **
      7 REM **      intersecției a doua multimi  **
      10 REM initializare
      20 INPUT N , M
      30 DIM A(N) : DIM B(M)
      40 REM introducere valori
      50 FOR I = 1 TO N
      60 REM introducere valoare multime A
      70 INPUT A(I)
      80 REM verificare valoare introdusa
      90 FOR K = 1 TO I-1
     100 IF A(I)  A(K) THEN GOTO 130
     110 PRINT "Introduceti alta valoare"
     120 GOTO 70
     130 NEXT K
     140 NEXT I
     150 FOR I = 1 TO M
     160 REM introducere valoare multime B
     170 INPUT B(I)
     180 REM verificare valoare introdusa
     190 FOR K = 1 TO I-1
     200 IF B(I)  B(K) THEN GOTO 230
     210 PRINT "Introduceti alta valoare"
     220 GOTO 170
     230 NEXT K
     235 NEXT I
     240 REM afisare valori
     250 REM afisare valori multime A
     260 PRINT "A = ( ";
     270 FOR I = 1 TO N
     280 PRINT A(I) ; " " ;
```

```

290 NEXT I
300 PRINT " ) "
310 REM afisare valori multime B
320 PRINT "B = ( ";
330 FOR I = 1 TO M
340 PRINT B(I) ; " ";
350 NEXT I
360 PRINT " ) "
370 REM diferenta
380 PRINT " A - B = ( ";
390 REM determinare elemente multime
391 REM diferenta
400 FOR I = 1 TO N
410 FOR J = 1 TO M
420 IF A(I) = B(J) THEN GOTO 460
430 NEXT J
440 REM afisare element multime
441 REM diferenta
450 PRINT A(I) ; " ";
460 NEXT I
470 PRINT " ) "
480 REM reuniune
490 PRINT " A U B = ( ";
500 REM determinare elemente multime
501 REM reuniune
510 FOR I = 1 TO N
520 REM afisare element multime
521 REM reuniune (originar din A)
530 PRINT A(I) ; " ";
540 NEXT I
550 REM evitarea repetarii afisarii
551 REM unui element in reuniune
560 FOR J = 1 TO M
570 FOR I = 1 TO N
580 IF B(J) = A(I) THEN GOTO 620
590 NEXT I

```

```

600 REM afisare element multime
601 REM reuniune (originar din B)
610 PRINT B(J) ; " ";
620 NEXT J
630 PRINT " ) "
640 REM intersectie
650 PRINT " A X B = ( ";
660 REM determinare elemente multime
661 REM intersectie
670 FOR I = 1 TO N
680 FOR J = 1 TO M
690 REM afisare element multime
691 REM intersectie
700 IF A(I) = B(J) THEN PRINT A(I);" ";
      : GOTO 720
710 NEXT J
720 NEXT I
730 PRINT " ) "

```

Deseori este bine ca în cadrul programelor să se facă o verificare prealabilă a datelor introduse, iar în cazul în care o anumită dată nu corespunde, să se repete introducerea. Să presupunem, de exemplu, că  $A(I)$  sînt niște date care trebuie introduse și care reprezintă numere întregi și pozitive mai mici ca 1000.



**Cum trebuie continuat programul astfel încît să se realizeze o verificare a datelor introduse?**

Se va adăuga o linie programului P25 care face această verificare. Dacă data nu corespunde, se va afișa un mesaj prin care se va cere repetarea introducerii:

```

75 IF A(I) <> INT (A(I)) OR A(I) < 0 OR
   A(I) > 1000 THEN PRINT "data eronata;
   repetati introducerea" : GO TO 70

```



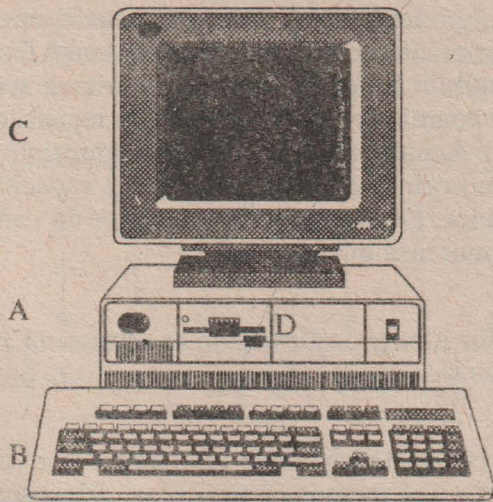
## Din ce este format un calculator ?

Calculatoarele au pătruns în viața de toate zilele ca un instrument indispensabil pentru fiecare domeniu de activitate.

Cele de tip PC (Personal Computer) - figura 8 - sînt și cele mai răspîndite și mai utilizate dintre calculatoare datorită gradului lor de accesibilitate dar și prețului relativ scăzut. Indiferent, însă, de tipul calculatorului, fie că este vorba de un sistem de dimensiuni mai mari (**mainframe**), un **minicalculator** sau unul de tip PC, un calculator este format din mai multe părți conectate între ele și care împreună alcătuiesc un tot, funcționarea fiecărei părți fiind integrată în funcționarea întregului sistem.

fig.8

*Calculatorul cu părțile  
lui componente*



**Unitatea centrală** (vezi partea A din figura 8) este componenta principală (creierul calculatorului) care practic conduce și controlează întregul proces din cadrul sistemului. În cadrul ei se găsește **unitatea aritmetică și logică** unde se desfășoară toate operațiile aritmetice și logice. De obicei, din punct de vedere fizic, unitatea centrală este, de fapt, cutia (sau dulapul) pe care în mod "impropriu" o numim calculator. Pentru calculatoarele per-

sonale, întreaga unitate centrală se găsește, de obicei, pe o singură placă ce conține mai multe circuite integrate. Unitatea aritmetică și logică, în acest caz, este unul din circuitele de pe placă și este un *microprocesor*, o bucățică de cristal de siliciu tratată special, încapsulată într-un înveliș de ceramică și care poartă denumirea de *cip*. Microprocesorul asigură practic toate operațiile în calculator, adică execută operații aritmetice și logice, decodifică instrucțiuni, transmite altor circuite din sistem semnale de control etc. Pentru calculatoarele pe care le cunoașteți cel mai bine (HC, CIP, COBRA, TIM, JET) microprocesorul utilizat se numește **Z-80** și conține câteva registre de memorie. Calculatoarele de tip PC sînt construite de obicei cu microprocesoare Intel 8088 sau 8086.

**Memoria internă** se găsește tot în cutia amintită nefiind accesibilă vederii noastre. Este formată tot din circuite electronice - registre - (ceva mai mici decît microprocesorul) care au proprietatea de a memora informații. Practic ele memorează secvențe de 0 și 1 (se folosește, deci, **sistemul de numerație binar**) care reprezintă date sau programe. Astfel 0 și 1 sînt cele mai mici unități de informație care circulă în calculator și ele se mai numesc **biți**.

Este important faptul că atît registrele microprocesorului cît și cele ale memoriei interne pot memora o secvență fixă de biți. Pentru calculatoarele compatibile Sinclair Spectrum mărimea secvenței este de 8 biți. Pentru calculatoarele de tip PC capacitatea regiștrilor este, de obicei, de 16 sau 32 de biți. Această secvență de 8 biți reprezintă un **octet** (sau **byte** în engleză) și el reprezintă unitatea de măsură a capacității (mărimii) memoriei calculatoarelor. Deci, practic în calculator un caracter ca A sau 9, de exemplu, se memorează într-un octet.

☞ *Nu uitați: informațiile se reprezintă prin biți, dar se transmit și se interpretează informațiile conținute în secvențe de 8 biți (octeți).*

De aceea toate registrele unui calculator (indiferent dacă sînt din microprocesor sau memorie) au aceeași dimensiune (8 biți la calculatoare Sinclair Spectrum) astfel încît ceea ce iese dintr-unul să poată intra în altul. Din acest motiv se mai spune că aceste calculatoare (Sinclair Spectrum) sînt calculatoare pe 8 biți, spre deosebire de PC-urile construite cu microprocesoare INTEL (8088) ale căror registre, după cum am văzut, pot memora 2 sau 4 octeți, fiind deci, calculatoare pe 16 sau 32 de biți. Bineînțeles că dacă un calculator va avea registre de capacitate mai mare,

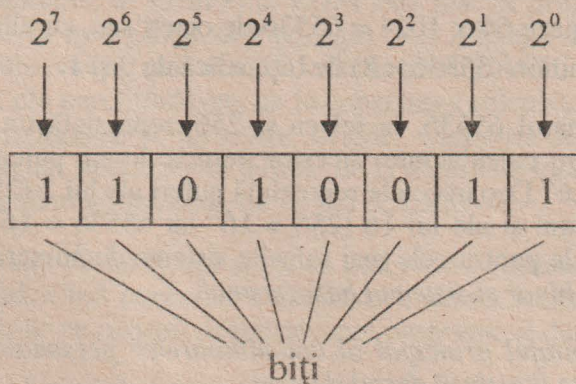
adică va putea memora și transmite o cantitate mai mare de informație, va fi mai puternic.

Mărimea memoriei este vitală pentru viteza și eficiența cu care sistemul va opera. Tehnologia circuitelor imprimate pe siliciu (așa numitele **cip-uri**) a evoluat foarte rapid ceea ce a permis o mare miniaturizare precum și posibilitatea ca o capacitate de memorie mare să fie cuprinsă într-o unitate centrală de dimensiuni mici. Așa că nu subestimați micile calculatoare pe care le aveți în față. Puterea lor (care este dată, în general, de doi factori importanți, și anume, *viteza de calcul și capacitatea de memorie*) este similară cu cea a sistemelor de calcul care erau folosite de specialiști la nivelul anilor 1975.

Am văzut că pentru calculatoarele cu care lucrăm, în fiecare circuit de memorie "intră" un octet sau, cu alte cuvinte, 8 biți. Fiecare din acești biți va avea o valoare (pondere) în funcție de poziția pe care o ocupă în octet (vezi figura 9), întreaga înșiruire a celor 8 biți formând un număr (o valoare) pe care îl putem calcula. Primul bit (cel mai din stânga) este *cel mai semnificativ*, având ponderea 128 ( $2^7$ ). Apoi ponderea biților scade pe măsură ce ne deplasăm spre dreapta (scăzând puterile lui 2), ultimul bit (cel mai din dreapta) având valoarea 1 ( $2^0$ ). Spunem că bitul cel mai din dreapta este *cel mai puțin semnificativ*. Valoarea pe care o reprezintă octetul o putem calcula în mod asemănător cu "citirea" unui număr (de fapt valoarea unui număr din sistemul zecimal, dar considerând baza 2).

fig.9

Octetul



## Octetul

---

De exemplu, înșiruirea de cifre 3524 din sistemul zecimal va reprezenta ca valoare 3 mii 5 sute 2 zeci și 4, sau, de fapt, 3 mii + 5 sute + 2 zeci + 4, adică:

$$3 \times 10^3 + 5 \times 10^2 + 2 \times 10^1 + 4 \times 10^0$$

În mod similar, putem calcula valoarea octetului din figura 9, și anume:

$$1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = \\ 128 + 64 + 16 + 16 + 2 + 1 = 211$$

Observăm că cel mai mic număr dintr-un octet se va forma atunci când toți biții vor fi "puși pe 0", în acest caz numărul avînd valoarea 0, iar, cel mai mare număr se va forma când toți biții vor fi "puși pe 1", iar acest număr va fi 255.

Pentru calculatoarele pe care le cunoașteți, capacitatea lor de memorie este de 64 Ko (kilo octeți). În sistemul zecimal, prefixul kilo indică 1000, dar în sistemul binar (care se utilizează, după cum am văzut pentru calculatoare), 1 kilo înseamnă 1024 deoarece trebuie să reprezinte o putere a lui 2 (și anume 2 la puterea a 10-a).

Acum putem să ne reprezentăm mai clar ce înseamnă capacitatea de memorie internă a calculatorului știut: el poate la un moment dat să înmagazineze  $64 \times 1024 = 65536$  de octeți sau, cu alte cuvinte, peste jumătate de milion ( $65536 \times 8$ ) de biți, adică de 0 și 1.

Numărul 65536, la fel ca și 256, reprezintă un număr foarte important pentru calculatoare. Acestor numere le-am putea spune **numere magice**. De ce? Deoarece ele reprezintă puteri ale lui 2 ( $256 = 2^8$ , iar  $65536 = 2^{16}$ ) precum și ale lui 16 ( $256 = 16^2$  iar  $65536 = 16^4$ ), iar 2 și 16 reprezintă bazele pentru cele mai folosite sisteme de numerație la calculatoare: *sistemul binar și sistemul hexazecimal*.

*Avantajul principal al calculatoarelor personale este acela că memoria poate fi extinsă destul de ușor.*

Multe dintre tipurile de calculatoare prezintă chiar în denumirea lor caracteristica referitoare la capacitatea de memorie. Astfel vechile sisteme de

calcul din țara noastră se numeau FELIX C32, FELIX C256, FELIX C512, FELIX C1024. Numerele respective reprezentau de fapt, capacitatea de memorie în Kiloocteți.

Observați că deși aceste sisteme erau de dimensiune mare (dulapuri), ele prezentau o capacitate de memorie comparabilă cu a PC-urilor actuale.

Similar ZX SPECTRUM 128 înseamnă un calculator tip SINCLAIR SPECTRUM cu o capacitate de memorie internă extinsă de la 64 Ko la 128 Ko, iar COMMODORE 64 reprezintă un calculator COMMODORE cu o capacitate de memorie internă de 64 Ko.

Pentru a putea fi referiți sau modificați, tuturor octeților memoriei interne le sînt asociate numere care reprezintă **adresele de memorie** ale octeților. La calculatoarele Sinclair Spectrum prima adresă de memorie este adresa 0, iar ultima este 65536, corespunzătoare capacității de memorie de 64 Ko.

Memoriile interne sînt de două feluri: unele care memorează programele (adică sînt deja "pline") care fac să "meargă" calculatorul și altele care sînt "goale" așteptînd ca noi să le "umplem" cu date sau programe. Primele memorii sînt de tip ROM (**Read Only Memory**) adică pot fi numai citite, noi putînd afla (citi) conținutul lor dar nu și scrie (adică memora) altceva în ele. Înscrierea conținutului acestor memorii se face de fabricant, iar operația de înscriere cu programe (de fapt tot cu secvențe de 0 și 1) se mai numește **arderea memoriilor**.

În cazul calculatoarelor amintite, în memoriile ROM este înscris programul care face calculatorul să meargă (și să înțeleagă) prin limbajul BASIC. De aceea aceste calculatoare se mai numesc **calculatoare BASIC** sau **cu BASIC-ul încorporat**, ele permițîndu-ne, de la conectarea calculatorului la rețeaua electrică introducerea de programe BASIC. Memoria de tip ROM la calculatoarele Sinclair Spectrum se întinde de la adresa de memorie 0 pînă la adresa 16383, fiind vorba, deci, de 16 Ko de memorie.

Celelalte memorii interne sînt de tip RAM și ele se mai numesc **memorii la dispoziția utilizatorului**. Cînd conectăm calculatorul, aceste memorii sînt "goale", ele umplîndu-se pe măsură ce introducem programe și date.

Putem inițializa calculatorul acționînd un buton de RESET, iar acest lucru realizează practic "golirea" memoriilor de tip RAM. Bineînțeles, memoriile de tip RAM reprezintă un volum de memorie mai mare decît cel reprezentat de memoriile tip ROM.

## Să ne jucăm cu memoria

---

Există două instrucțiuni pereche în limbajul BASIC prin intermediul căroră se poate lucra cu memoria: PEEK și POKE. Cu PEEK se poate citi ce este la o adresă de memorie, adică se poate vizualiza conținutul respectiv. Din acest motiv instrucțiunea PEEK se utilizează în asociație cu instrucțiunea PRINT.

PRINT PEEK 0 va afișa pe ecran conținutul primului octet al memoriei (bineînțeles un număr cuprins între 0 și 255).

Cu POKE se poate modifica valoarea unui octet dintr-o adresă de memorie.

Să experimentăm ce am învățat despre memoria ROM și RAM prin intermediul instrucțiunilor PEEK și POKE.

Să vizualizăm conținutul unei adrese de memorie (de exemplu adresa 1000) și apoi să încercăm să modificăm această valoare:

```
PRINT PEEK 1000
```

Se va afișa numărul 9, ceea ce înseamnă că octetul de la adresa 1000 are valoarea 9. Să încercăm să modificăm această valoare de la 9 la 10:

```
POKE 1000 , 10
```

Să vizualizăm, din nou, conținutul acestei adrese:

```
PRINT PEEK 1000
```

Observăm că a fost afișat tot numărul 9 deoarece adresa 1000 este o adresă de memorie ROM (aceste adrese sînt cele de la 0 la 16383) și, deci, conținutul său nu se poate modifica.

Să încercăm acum cu o adresă de memorie de tip RAM, deci, mai mare ca adresa 16384, să zicem 20000:

```
PRINT PEEK 20000
```

Obținem valoarea 0. Adresa 20000 referă un octet din memoria de tip RAM atașată ecranului grafic. Era normal să obținem valoarea 0 deoarece

ecranul este gol. Să modificăm această valoare printr-o nouă valoare, și anume, 170:

```
POKE 20000 , 170
```

Acum vizualizînd din nou valoarea conținută în adresa 20000, obținem, într-adevăr, noua valoare, și anume, 170, deoarece 20000 este o adresă de memorie RAM.

Mai mult, se poate observa chiar pe ecran consecința modificării (practic se vizualizează octetul) și anume o linie punctată, în partea de mijloc stînga a ecranului, corespunzătoare șirului de biți 10101010 care reprezintă în sistemul binar numărul 170. Vizualizarea modificării s-a produs deoarece adresa de memorie 2000 este o adresă asociată ecranului grafic.

Uneori este necesar să memorăm o informație a cărei valoare reprezintă un număr superior valorii 255. În acest caz vom reprezenta această valoare pe 2 octeți avînd, deci, nevoie de două adrese de memorie consecutive.

*Cum se face citirea informațiilor pentru valori mai mari ca 255?* Să presupunem, de exemplu, că vrem să știm care este adresa de început a unui program BASIC. Această adresă va fi un număr mai mare decît 16384, deoarece programul BASIC se situează (după cum am văzut) în partea de memorie RAM. Pentru memorarea numărului care reprezintă adresa sînt necesari 2 octeți, și anume, cei de la adresele 23635 și 23636. Citirea informației din acești 2 octeți prin care vom afla adresa de început a programului BASIC se face astfel:

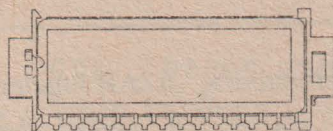
```
PRINT PEEK 23635 + 256 * PEEK 23636
```

Observăm că valoarea octetului de la adresa mai mare se înmulțește cu numărul 256. Obținem valoarea 23755 care reprezintă adresa de început a programului BASIC.

Dacă deschidem capacul "cutiei" calculatorului vom vedea placa pe care se situează circuitele: microprocesorul (care are dimensiuni ceva mai mari, de cîteva centimetri) și, ordonate sub formă de rînduri, mai multe circuite de memorie RAM de culoare închisă și de dimensiuni mai mici precum și 8 circuite de culoare deschisă (și ceva mai mari) de memorie ROM.

După cum am văzut, de cîteva ori ne-am referit în descrierea părților componente ale calculatorului și la programe. Într-adevăr, putem de la început să semnalăm faptul că un calculator este format practic din două

## Circuit ROM



părți: o parte de echipamente (este o parte fizică, palpabilă, care include unitatea centrală, memoriile, etc) și care se mai numește **hardware** (tocmai pentru că este o parte fizică, adică "tare") și o parte de programe care se mai numește **software** (pentru că nu este o parte fizică, adică este "moale", nepalpabilă).

*Programele înscrise de fabricant în memoriile ROM fac parte din software. În afara acestora tot din software mai fac parte și programele realizate de noi pe măsură ce le introducem în memoria calculatorului.*

## Echipamente periferice

Pînă în acest moment am descris elementele (componentele) centrale din cadrul hardware-ului. Celelalte părți componente ale calculatorului (adică tastatura, monitorul, casetofonul, unitățile de discuri, imprimanta etc) se mai numesc și **unități periferice**, în totalitatea lor alcătuiind periferia calculatorului.

**Tastatura** (vezi partea B din figura 8) este o componentă asemănătoare cu o mașină de scris. Prin intermediul ei instrucțiunile și datele se introduc în calculator, de aceea se spune că tastatura este un **echipament (dispozitiv) de introducere (sau intrare)**.

Deși la calculatoarele pe care le cunoașteți tastatura este pe "capacul" cutiei calculatorului ea face parte din *periferia calculatorului*. La multe calculatoare (de exemplu la cele de tip PC) ea se detașează de unitatea centrală fiind legată de aceasta printr-un **cablu**.

Tastatura este dotată cu taste care reprezintă litere, cifre, precum și alte simboluri. La calculatoarele pe care le cunoașteți prin acționarea unei taste se pot obține și cuvinte cheie care reprezintă **instrucțiuni sau comenzi**.

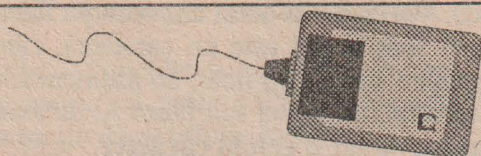
Alt dispozitiv de introducere poate fi cel de tip "**șoarece**" (figura 11), prin mișcarea sa și prin acționarea butoanelor putîndu-se introduce date sau comenzi. La calculatoarele de tip PC tastatura prezintă mai multe taste, iar



instrucțiunile (cuvintele cheie) se introduc prin tastarea lor caracter cu caracter.

fig.11

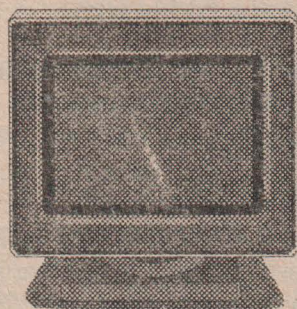
Mouse



**Monitorul** (sau televizorul) (vezi partea C din figura 8 și figura 12) este utilizat pentru a afișa pe ecranul său diverse rezultate, date sau mesaje. De aceea se mai numește **dispozitiv de afișare**. Spunem că el este un **echipament periferic de extragere** (sau ieșire) deoarece cu ajutorul lui nu putem decât să citim rezultatele sau datele de pe ecran, adică să le extragem. Spre deosebire de televizoare, monitoarele (tuburile cu raze catodice CRT) sînt proiectate special pentru folosirea la calculator dispunînd de posibilități complexe de afișare (o rezoluție mai mare).

fig.12

Monitor



Pe ecranul calculatorului se pot afișa caractere literale (litere, cifre, semne de punctuație, alte semne) și caractere grafice (în modul G), iar în acest caz spunem că ecranul este în **mod text**. În afară de caractere se pot realiza și desene (grafică), iar în acest caz ecranul va fi în **mod grafic**.

Pentru a afișa caractere sau grafică, toate aceste obiecte trebuie memorate. Deci ecranul calculatorului are un corespondent în memoria internă, acest corespondent numindu-se **memoria ecran**, adică memoria necesară reprezentării ecranului. Pentru a putea fi reprezentat în memorie ecranul este organizat într-un anumit mod în funcție de obiectele afișate pe ecran (caractere sau grafică). Acest mod de organizare a memoriei (ecran) se mai numește **maparea memoriei**.

Să vedem cum se face "corespondența" dintre ecran și memorie la calculatoarele HC în cazul în care se afișează caractere (modul text).

În acest caz ecranul este organizat pe linii și pe coloane. Puteți observa acest lucru și singuri, introducând, de exemplu, mai multe linii de caractere (text). Veți nota în acest caz cum pe fiecare linie încap 32 de caractere (atenție, și spațiul gol este un caracter), care se aliniază exact unele sub altele formînd linii. Veți nota, de asemenea, că fiecare caracter, indiferent de mărimea lui, un punct sau litera A, de exemplu, ocupă același spațiu pe ecran, spațiu numit și **celulă caracter**.

Astfel în *mod text* ecranul este organizat pe 32 de coloane (numerotate de la stînga la dreapta de la 0 la 31) și pe 22 de linii (numerotate de sus în jos de la 0 la 21). În plus mai sînt două linii în partea de jos a ecranului, pe care se afișează mesajele calculatorului. Concluzionăm că sînt 24 de linii și 32 de coloane astfel încît ecranul poate fi acoperit de  $24 \times 32 = 768$  caractere.

Ținînd cont de faptul că fiecărei celulă caracter îi corespunde în memorie 8 octeți, rezultă că întreaga memorie ecran are o mărime de  $768 \text{ celule caracter} \times 8 \text{ octeți} = 6144$  octeți.

Pentru a fi afișate pe ecran caracterele au fost în prealabil "desenate" prin puncte, iar punctele s-au obținut prin "punerea pe 1" a unor biți din cadrul octeților care alcătuiesc caracterul. Să nu vă fie teamă! Caracterele au fost proiectate în prealabil de către constructor, întreg setul de caractere folosite de calculator (se regăsesc pe tastatură) fiind și el memorat într-o anumită zonă a memoriei. Există însă posibilitatea ca în mod similar să vă construiți propriile caractere (pentru ș, î, ț, ă, de exemplu) și să le obțineți acționînd tasta atribuită caracterului în mod grafic (G).

Să vedem cum se face "corespondența" dintre ecran și memoria ecran în celălalt caz, adică atunci cînd vrem să afișăm desene (grafică). Evident va trebui să obținem aceeași mărime pentru memoria ecran.

În acest caz desenul se compune din mai multe puncte, iar ecranul va avea 256 de puncte (se mai numesc **pixeli**) pe orizontală (de la 0 la 255) și 176 puncte pe verticală (de la 0 la 175). Practic fiind necesar să se memoreze fiecare punct de pe ecran (pixel), iar fiecărui pixel corespunzîndu-i un bit numeric, vor fi necesari  $256 \times 176 = 45056$  biți, adică  $45056 : 8 = 5632$  octeți. Am văzut însă că mai sînt două linii în partea de jos a ecranului necesare pentru a afișa mesaje, pe această porțiune de ecran nerealizîndu-se în mod normal grafică. Această zonă ridică cu  $32 \text{ caractere} \times 2 \text{ linii} \times 8 \text{ octeți} = 512$  octeți mărimea memoriei, care devine astfel  $5632 + 512 = 6144$  octeți, adică la fel ca în cazul precedent.

Se observă că în primul caz – modul text – caracterele fiind obiectele care se puteau pune pe ecran, acesta se acoperea cu numai 768 de caractere. De aceea acest tip de grafică se mai numește de **rezoluție scăzută**.

În celălalt caz – modul grafic – punctele fiind obiectele care se puteau pune pe ecran, acesta se acoperea cu peste 45000 de puncte. De aceea acest tip de grafică se mai numește de **rezoluție înaltă**.

Cum se fac în acest caz reprezentări grafice pe ecran? În mod similar, când unul din biții corespunzători unui octet care reprezintă o adresă de memorie ecran este pus pe 1, atunci un punct se va desena pe ecran și spunem că punctul (bitul) este *aprints*. Dacă bitul este pe 0, atunci nu apare nimic pe ecran, punctul (bitul) fiind *stins*.

Concluzionăm că cele 6144 de adrese de memorie, și anume, cele cuprinse între 16384 și 22528 reprezintă adrese de memorie ecran.

*Cum se pot memora datele în afara memoriei interne a calculatorului ?*

Unitățile de **memorie externă** înmagazinează datele și instrucțiunile în afara unității centrale permițând ca acestea să fie utilizate repetat sau păstrate pentru cerințe ulterioare.

**Casetofonul** este o unitate periferică cu ajutorul căreia se încarcă în memoria calculatorului programe (de exemplu, jocuri de pe casete) sau date. De asemenea tot cu el se pot "salva" adică înregistra pe casete, programe sau date pe care în prealabil le-am introdus în memoria internă a calculatorului și pe care dorim să le folosim și altă dată. Dacă nu le-am "salva" pe casetă, le-am pierde odată cu deconectarea calculatorului de la rețea. Spunem că unitatea casetofon este o **unitate de introducere/extragere**, deoarece cu ea putem, așa cum am văzut, atât să introducem (să încărcăm) cât și să extragem (adică să salvăm) programe și date.

Salvarea (înregistrarea) programelor și datelor se realizează pe **suporturi magnetice**. Acestea joacă rolul de memorie externă deoarece sînt în afara calculatorului. În timp ce memoria internă, fiind formată din circuite electronice (scumpe) este mai limitată, cea externă este mai ieftină și poate înmagazina un volum mai mare de date și programe.

Suportul magnetic pentru casetofon este obișnuita *casetă magnetică*.

Capacitatea de memorie externă se măsoară tot în octeți (sau kilo octeți). După cum știți pe o casetă magnetică de o oră încap circa 10-12 jocuri de

---

*Din ce este format un calculator ?*

fig.13

Casetă magnetică

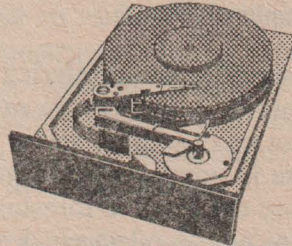


dimensiuni mari, adică din acelea care ocupă aproape întreaga memorie internă (de tip RAM) a calculatorului. Deci capacitatea de memorie a unei casete este de circa 10 ori mai mare decât capacitatea memoriei interne a calculatorului, adică, aproximativ 400-500 ko.

Cel mai frecvent utilizate unități de memorie externă pentru calculatoarele PC sînt **unitățile de discuri flexibile** (vezi partea D din figura 8 și figura 14). Acestea sînt unități periferice tot de introducere/extragere cu ajutorul cărora, exact ca în cazul casetofonului se pot memora sau încărca în memoria internă programe sau date.

fig.14

Unitate de disc flexibil

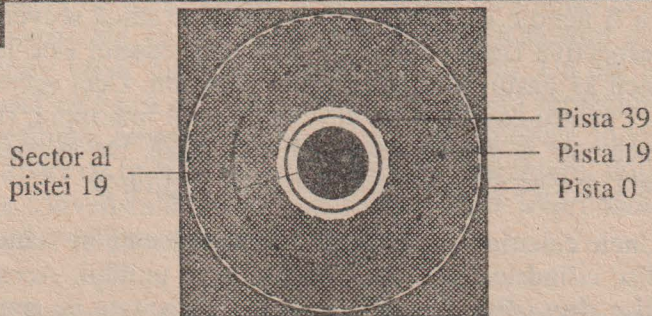


Suportul magnetic pentru unitatea de discuri este **discheta** sau **discul flexibil** (figura 15a, 15b) care este un disc magnetic care poate avea diverse dimensiuni (8 inch, 5,25 inch și 3,5 inch). Capacitatea de memorie a unei dischete este în funcție de tipul ei. Unele pot memora un volum de informații mai mic decât o casetă magnetică (256 Ko sau 360 Ko), altele pot memora un volum mai mare (720 Ko sau 1200 Ko). Acest volum variază în funcție de mărimea dischetei și densității înregistrărilor.

Unele discuri flexibile au o singură suprafață de înregistrare (față), altele prezentînd două suprafețe de înregistrare fiind deci, dublă față. Pe suprafața de înregistrare se află dispuse cercuri concentrice care se înfășoară dinspre centru spre margine. În figura 15a puteți observa modul în care este organizată o dischetă pe piste și pe sectoare, iar în figura 15b cum se

fig. 15a

*Organizarea unei dischete*



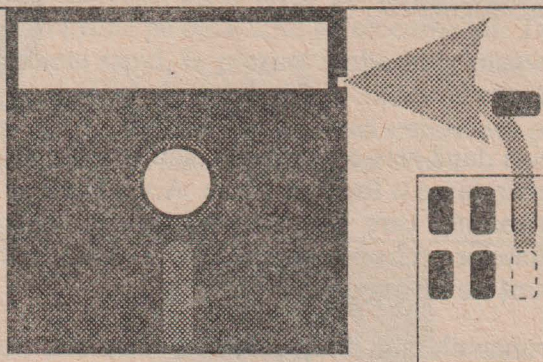
realizează protejarea unei dischete pentru ca informațiile înregistrate (date sau programe) să nu fie șterse din greșeală.

În interiorul unității de discuri se găsesc montate mai multe capete de citire/scriere care înregistrează și regăsesc (citesc) informații de pe **discul magnetic**.

Discurile magnetice ca și banda (casetă) magnetică reprezintă, deci, suporturi fizice pe care se înregistrează informațiile. Înregistrările sînt de tip magnetic.

fig. 15b

*Protejarea unei dischete la scriere*



Dacă ar fi să alegem ca memorie externă între casetă sau dischetă, ce ați prefera? Acasă probabil preferați casetofonul pickupului dar, desigur, discul magnetic este de preferat deoarece în cazul utilizării sale, *accesul la informație este direct*. Așa cum atunci cînd dorim să ascultăm o anumită melodie de pe un disc muzical poziționăm acul chiar la începutul său, tot la fel, în cazul unității de discuri de la calculator, atunci cînd dorim să încărcăm un anumit program de pe disc și dăm comanda respectivă, capul de citire al acelei unități de discuri se va poziționa exact pe acel program. Spre deosebire, pentru caseta magnetică, dacă dorim un anumit program

va fi nevoie să *parcurgem secvențial* (adică secvență cu secvență) banda respectivă (indiferent dacă derularea casetei este lentă sau rapidă) astfel încât să găsim începutul aceluia program. Deci accesul pentru disc față de cel pentru casetă, fiind direct este mult mai rapid, iar durata operațiilor de încărcare și salvare este mult mai mică. De asemenea informația memorată pe disc magnetic este relativ în siguranță.

Unele calculatoare de tip PC au încorporate în "cutia" calculatorului și un disc cilindric fixat, deci, în interiorul unității. Acest disc se mai numește **disc dur (disc "hard")** și are o capacitate de memorie foarte mare, de circa 10-100 ori mai mare decât capacitatea unei dischete. Aceste discuri au marele avantaj că pot reține multe programe și date chiar în interiorul calculatorului.

Unele calculatoare mai mari (sisteme sau minicalculatoare) mai folosesc ca unități de memorie externă și **unitățile de benzi magnetice**. Acestea deși sînt mai lente (accesul fiind tot secvențial) au o capacitate de memorie foarte mare. Ele folosesc ca suport de memorie tot banda magnetică (figura 16).

fig. 16

*Bandă magnetică*



Deseori după ce s-a lucrat cu calculatorul este necesar ca rezultatele să fie înregistrate pe o foaie de hîrtie. Acest lucru se realizează prin intermediul **imprimantei** care este un echipament periferic folosit pentru tipărirea rezultatelor prelucrate de calculator. Spunem că imprimanta este un **echipament de extragere**.

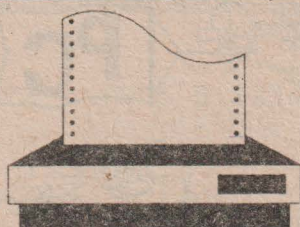
Imprimantele sînt foarte diferite între ele, deosebindu-se prin viteza de imprimare a caracterelor și putînd fi lente sau rapide (acestea din urmă putînd imprima pînă la mai multe mii de caractere pe secundă). Viteza de imprimare este condiționată de modul de imprimare folosit iar acesta poate fi lent, dacă se imprimă caracter cu caracter, sau mai rapid, dacă se tipărește o linie deodată. Unele imprimante permit tipărirea de la stînga la dreapta și apoi de la dreapta la stînga, ceea ce mărește foarte mult viteza de lucru prin eliminarea timpului pierdut la returul de car.

Foarte importantă este *metoda de imprimare*. Există imprimante care imprimă prin lovirea hîrtiei (*imprimare cu impact*) printr-un proces asemănă-

tor cu cel folosit la mașina de scris. Dintre imprimantele cu impact cele mai folosite sînt cele *matriciale*, la care capul de scriere este alcătuit din fire care reprezintă fiecare literă, cifră sau simbol (figura 17a). Un alt tip de imprimantă cu impact este cea cu "margaretă". Margareta reprezintă de fapt, un cap de imprimare compus din roțițe care au înglobate litere, cifre și alte caractere. Roata de tipărire se învîrte pînă cînd caracterul dorit se găsește într-o poziție convenabilă pentru imprimare. Aceste imprimante sînt mai lente dar asigură o calitate deosebită a tipăririi.

fig.17a

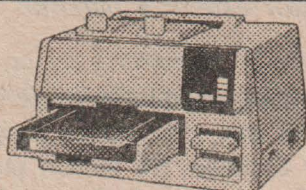
*Imprimantă cu impact*



Alte metode de imprimare sînt cele folosite la *imprimantele termice* care, după cum arată și numele, generează caracterele pe hîrtie sensibilă la căldură și imprimantele care imprimă pe baza unui *jet de cerneală*. Tehnologia imprimării evoluează foarte rapid, în prezent multe imprimante oferind facilități grafice ce pot genera ilustrații, hărți, planuri, etc. într-o diversitate de dimensiuni și culori. Tehnologia cea mai cunoscută pentru astfel de imprimări este cea pe bază de *laser* (vezi figura 17b).

fig.17b

*Imprimantă cu laser*



Imprimanta cu laser își găsește o întrebuințare și în realizarea cărții de față. Punerea în pagină (tehno-redactarea) acesteia a fost făcută pe un calculator PC (AT 80386), apoi cartea a fost listată pe o imprimantă cu laser, iar după efectuarea corecturilor, rezultatul a fost fotografiat și multiplicat de către tipografie.



## Calculatoarele în competiție

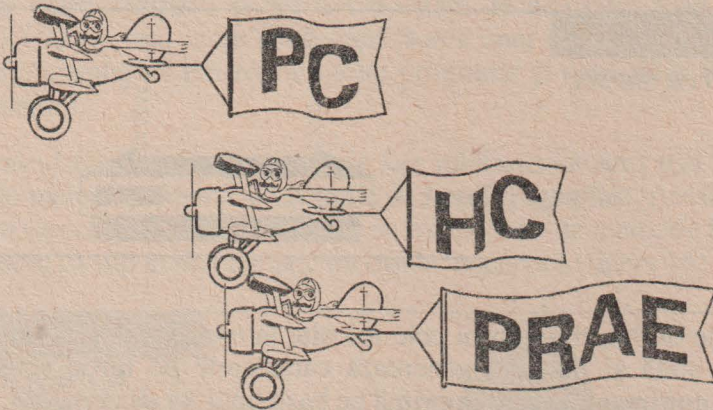


fig.18

### Calculatoarele în competiție

Deseori în utilizarea tehnicii de calcul se pot auzi expresii de genul: "acest calculator este mai puternic" sau "calculatorul X este cel mai rapid".

Deși aceste expresii au intrat aproape în limbajul curent al utilizatorilor de calculatoare, cei mai mici dintre aceștia au numai o înțelegere intuitivă asupra acestor expresii. Să intrăm puțin mai în adâncul chestiunii și să presupunem că avem în față câteva calculatoare de tipuri diferite. Să luăm un calculator compatibil Sinclair Spectrum (HC, de exemplu) și să tastăm:

```
PRINT (SIN 0.123456789)^2 + (COS 0.123456789)^2
```

Pentru PC nu uitați să puneți argumentele pentru SIN și COS între paranteze, regula generală la GWBASIC fiind să se utilizeze paranteze pentru argumentele funcțiilor.



Așa cum ne așteptam vom obține valoarea 1 (deoarece  $\sin^2 x + \cos^2 x = 1$ ). Să repetăm acum exemplificarea cu următoarea mică diferență (vom modifica argumentul la cosinus la a 9-a poziție zecimală cu o unitate):

```
PRINT (SIN 0.123456789)^2 + (COS 0.123456788)^2
```

Vom obține și de data aceasta valoarea 1 ceea ce nu ne mai mulțumește pe deplin deoarece argumentul lui COS este puțin diferit de cel al lui SIN, iar calculatorul nu a fost așa de puternic încât să sesizeze această mică diferență. În schimb cu un calculator PRAE (ei da, bătrînul PRAE care așa cum îi arată numele a fost primul calculator personal realizat în țară) vom obține cu :

```
PRINT (SIN 0.123456789)^2 + (COS 0.123456789)^2
```

valoarea 1, iar cu:

```
PRINT (SIN 0.123456789)^2 + (COS 0.123456788)^2
```

valoarea 1.0000000002, de unde deducem că mica diferență a fost sesizată.

În cazul expus, prin "puterea" calculatorului am înțeles, de fapt, precizia sa de calcul, care este dată de numărul de cifre semnificative cu care sînt exprimate numerele. Într-adevăr, HC are o precizie de 9 cifre semnificative iar PRAE-ul de 11 cifre. Acest lucru se poate constata punînd calculatoarele să afișeze diverse numere. De exemplu, la HC cu

```
PRINT 0.123456789012345
```

se va afișa 0.12345679 în timp ce la PRAE se va afișa 0.12345678901.

Deci PRAE este mai puternic? Să nu ne grăbim cu afirmația deoarece putem avea surprize. Iată una dintre ele: să efectuăm cu PRAE un simplu calcul de ridicare la pătrat:

```
PRINT 10^2
```

și spre stupoarea noastră vom obține rezultatul 99.99999999 care este foarte apropiat de 100 dar nu identic.

Iată altă surpriză și mai mare. Să luăm un calculator PC AT (care știm că este mult mai puternic fiind din altă generație și cu o capacitate de memorie mult mai mare) și să repetăm exercițiile făcute cu PRAE și HC prin

intermediul lui GWBASIC încărcat de pe o dischetă. Vom observa că în primul caz cu sinusul și cosinusul avînd același argument obținem valoarea 0.9999999, iar în al doilea caz (cu argumentul lui cosinus modificat cu o unitate la a 9-a zecimală) același rezultat, ceea ce înseamnă că nici nu a calculat grozav și nici nu a sesizat diferența. Iar acest calculator are o precizie la afișare de nu mai puțin de 16 cifre zecimale (se poate constata deoarece cu PRINT 10.1234567890123456789 se va afișa 10.123456789012345).

*Puteți explica de ce se obțin aceste rezultate afit de diferite și, oarecum, contradictorii?*



## Viteza calculatoarelor

---

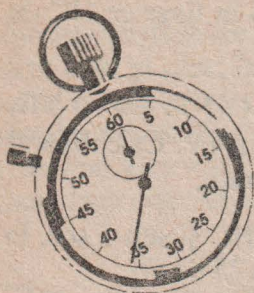
Cele două calculatoare (HC și PRAE) sînt comparabile deoarece:

- sînt construite pe baza aceluiași microprocesor și anume Z-80 ceea ce le conferă caracteristici asemănătoare (între care cea mai importantă este viteza de calcul)
- posedă aceeași capacitate de memorie (și anume 64 Ko).

De fapt aceste două caracteristici stau la baza a ceea ce numim puterea unui calculator. În schimb calculatorul PC AT este construit pe baza unui microprocesor 80286 și are o capacitate de memorie de 1Mo, deci are o putere mai mare.

*Și atunci cum se explică rezultatele obținute?*

Deoarece nu am luat în considerare un alt factor care determină puterea calculatoarelor, și anume, faptul că nu am apelat direct la posibilitățile calculatorului ci printr-un intermediar (în cazul nostru interpretorul BASIC) care este proiectat diferit la diferite calculatoare. În funcție de felul în care este construit acest interpretor numerele sînt reprezentate în simplă sau dublă precizie, memorîndu-se pe 4 octeți (la PRAE) sau pe 5 octeți (la HC) sau pe 8 (la PC în GW BASIC). În general proiectarea se face realizîndu-se un compromis între precizie și viteză (care este în funcție de modul în care se face analiza sintactică și semantică – interpretarea), modul de alocare și căutare a variabilelor etc.



Să punem acum *calculatoarele în competiție din punctul de vedere al vitezei lor în BASIC*. Pentru aceasta vom folosi un program care identifică dacă 10 007 este sau nu un număr prim. (În realitate așa cum am văzut este cel mai mic număr prim mai mare decât 10 000). Algoritmul folosit este cel clasic prin care se verifică dacă numărul este divizibil cu 2 și apoi începînd cu 3, din 2 în 2, pînă la jumătatea numărului (vezi programul P11):

 Ptest.

```
10 LET N = 10007
20 IF N/2 = INT (N/2) THEN PRINT N;
   " nu e prim": STOP
30 FOR I = 3 TO INT (N/2) STEP 2
40 IF N/I = INT (N/I) THEN PRINT N;
   " nu e prim": STOP
50 NEXT I
60 PRINT N ; " este prim"
70 STOP
```

Vom cronometra pentru fiecare calculator intervalul de timp dintre introducerea comenzii de executare a programului (RUN) și afișarea rezultatului.

Și iată rezultatele: PRAE dă răspunsul în 92 secunde, HC în 50 secunde, iar PCAT în 5 secunde.

Să calculăm aproximativ cam cîte operații (aritmetice și logice) face calculatorul la executarea acestui program :

- linia 10: 1 operație (de atribuire)
- linia 20: 3 operații (una de împărțire, una de calcul pentru aflarea întregului și una de comparare a celor două valori)
- linia 30: 2500 operații (5000:2 operații de atribuire)
- linia 40: 7500 operații (3 operații repetate de 2500 ori cît este ciclul)
- linia 50: 2500 operații (de comparare I cu N/2)
- linia 60: 1 operație (de afișare)

---

Total: 12505 operații

---

Astfel se poate spune că PRAE-ul are o viteză de 136 în timp ce HC de 250, iar PCAT de 2501 operații pe secundă.

Să optimizăm algoritmul modificînd în linia 30  $SQR(N)$  în loc de  $N/2$ , adică să mergem cu testarea divizibilității numai pînă la  $\sqrt{N}$ . Calculatorul va avea de făcut mai puțini pași și, deci, mai puține operații.

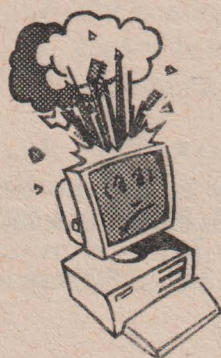
În acest caz rezultatele sînt: 3 sec. la PRAE, 2 sec. la HC și sub 1 sec. la PC AT, ceea ce înseamnă că eficiența a crescut de circa 25 de ori.

## Ce n-a putut rezolva calculatorul

---



Să se verifice dacă numărul 100 895 598 169 este prim sau nu.



Sigur că, la prima vedere, ne putem gîndi că, avînd la dispoziție programul optimizat pentru numere prime (mergînd cu testarea pînă la radical din număr), nu avem altceva de făcut decît să punem calculatorul la treabă și să așteptăm rezultatul. Dar, stupoare ! Atît calculatorul de tip HC cît și cel de tip PC AT ne răspunde că 100 895 598 169 este divizibil cu 2 !

Numărul de testat depășește limita de reprezentare pentru amîndouă calculatoarele.

Întrezăriți voi o altă metodă de rezolvare? Dacă după mai mult timp nu v-a venit nici o idee, nu vă impacientați, vă vom da noi răspunsul.

Numărul 100 895 598 169 nu este prim; el are ca divizori pe 898 423 și 112 303 (acest lucru se poate verifica și cu calculatorul, dar nu chiar așa de simplu, ci făcînd înmulțirea pe părți). Problema trimisă într-o scrisoare lui Fermat de către baronul Messner a fost rezolvată de celebrul matematician de pe o zi pe alta (fără calculator !), acesta nelăsînd, din păcate, și rezolvarea ei.


Calculatoarele sînt puse să rezolve și astfel de probleme, munca de cercetare nemapuțînd fi concepută fără ajutorul lor. Însă în astfel de cazuri se


folosesc calculatoare foarte puternice, de viteză și capacitate de memorie foarte mari și care, în plus, pot realiza și calcule *în paralel*. De exemplu, recent, cu unul din calculatoarele de acest fel s-a găsit cel mai mare număr prim descoperit pînă în prezent (am văzut că, numerele prime sînt din ce în ce mai rare cînd sînt căutate printre numere foarte mari). Numărul despre care este vorba are nu mai puțin de 227 832 cifre!


## Verificați-vă cunoștințele despre calculatoare


---


Puteți să vă verificați sau să vă îmbogățiți cunoștințele generale despre calculatoare: la ce sînt și cum sînt utilizate, cum sînt construite, istoricul lor. În acest scop veți selecta una din opțiunile A (adevărat) sau F (fals) pentru fiecare din afirmațiile din lista ce urmează verificînd, apoi, răspunsurile obținute cu cele exacte, indicate în ANEXA A de la pagina 151.


 1. Calculatoarele sînt inteligente. Ele pot răspunde practic oricărei întrebări. A F










 2. Calculatoarele pot lucra cu cuvinte la fel ca și cu numere. A F









 3. Un calculator poate realiza mii de operații pe secundă. A F











 4. Memoria calculatoarelor este foarte asemănătoare memoriei (creierului omenesc). A F

 5. Benzile magnetice pot fi utilizate ca unități de memorie ale calculatorului. A F


 6. Un program de calculator este un program TV despre calculatoare. A F


-  7. Un calculator poate avea o memorie suficient de mare pentru a memora o întreagă enciclopedie de informații. A F
-  8. Un calculator personal reprezintă un calculator aflat în proprietatea particulară a unei persoane. A F
-  9. Un calculator de buzunar este un calculator personal de dimensiuni foarte mici cu alimentarea prin baterii electrice și care poate fi transportat eventual în buzunarul unei haine. A F
-  10. Atît un calculator personal cît și unul de buzunar poate fi utilizat pentru rezolvarea unor probleme. A F
-  11. Atît un calculator personal cît și unul de buzunar poate prelucra atît numere cît și cuvinte. A F
-  12. Unul dintre primele dispozitive care ajutau la efectuarea calculelor matematice, abacul, a fost inventat în urmă cu aproape cinci mii de ani. A F
-  13. Primele calculatoare mecanice cu 4 funcții au fost construite la începutul secolului XVII-lea. A F
-  14. Cartelele perforate au fost utilizate în dispozitivele create de Jacquard pentru automatizarea muncii la războaiele de țesut la începutul secolului al XIX -lea. A F
-  15. Telefonul, automobilul, radioul și televiziunea au fost inventate înaintea calculatorului electronic. A F

-  16. În Anglia, în timpul celui de-al II-lea Război Mondial, Allan Turing a construit un calculator electronic cu care s-au decodificat mesajele secrete transmise de naziști. A F
-  17. Primul calculator electronic universal este considerat a fi ENIAC, construit în Statele Unite ale Americii în anul 1945. A F
-  18. Primul calculator electronic universal conținea tranzistoare și avea memoria internă pe ferite. A F
-  19. Primul calculator electronic utiliza tuburile de vacuum. A F
-  20. În anul 1947 erau deja instalate în întreaga lume câteva zeci de calculatoare electronice. A F
-  21. În anul 1947, IBM singurul producător de calculatoare electronice, a hotărât să nu mai investească în domeniul producției de calculatoare pe motiv că ele nu sînt cerute îndeajuns pe piață. A F
-  22. Un program pentru calculator reprezintă un set de instrucțiuni care arată ce pași trebuie parcurși pentru rezolvarea unei anumite probleme. A F
-  23. Sistemul de operare reprezintă un set de programe care are funcția de coordonare și control asupra calculatorului și care facilitează dialogul dintre utilizator și calculator. A F


-  24. Un program scurt poate fi utilizat uneori pentru a prelucra volume foarte mari de date. A F
-  25. Majoritatea roboților sînt cel puțin tot atît de inteligenți ca și o persoană medie. A F
-  26. Un elevator automat poate fi încadrat în categoria "mașinilor inteligente". A F
-  27. O formulă, de exemplu,  $S = V \times T$  reprezintă un model matematic. A F
-  28. Regăsirea informațiilor reprezintă utilizarea cea mai răspîndită a calculatoarelor. A F
-  29. Calculatoarele sînt utilizate inclusiv pentru realizarea de desene. A F
-  30. Dacă un program este destul de scurt (mai puțin de 20 de instrucțiuni) rularea (execuția) sa va dura mai puțin de o secundă. A F
-  31. O persoană care scrie instrucțiuni pentru calculator se numește **operator de calculator**. A F
-  32. O persoană care instalează și menține un calculator se numește **inginer de sistem**. A F
-  33. Înainte de a folosi un calculator este absolut necesară cunoașterea unor amănunte cum ar fi: cum este construit un calculator, care sînt părțile lui componente, pe ce tehnologie se bazează etc. A F



 34. Un program de calculator se poate utiliza și fără cunoștințe privitoare la modul în care a fost realizat. A F


 35. În urma executării următorului program, pe ecranul calculatorului se va afișa 3.6: A F

```
10 REM
20 PRINT "Rezultatul este: "
30 PRINT 10 - 2 * (5.3 - 2.1)
40 PRINT
45 CLS
50 PRINT 3.6
```

 36. În urma executării următorului program, pe ecran se va afișa: A F

5
3 + 2 = 5
3 + 2 = 6
3 * 2 = 6

```
10 PRINT 3 + 2
20 PRINT "3 + 2 = 5"
30 PRINT "3 + 2 = 6"
40 PRINT 3 * 2 = 6
```

 37. Ordinea corectă a liniilor următorului program este ACBDE. A F

```
A CLS
B LET A = B + 2
C INPUT B
D PRINT A , B
E STOP
```



38. Ordinea corectă a liniilor următorului program este A F  
ABCDEF G.

```
A CLS
B INPUT A
C LET B = A * B
D LET A = A + 2
E LET B = A + 1
F PRINT A , B
G STOP
```



39. După executarea următorului program care afișează A F  
suma primelor 10 numere naturale, introducând co-  
manda PRINT I se va afișa 1.

```
10 LET S = 0
20 FOR I = 1 TO 10
30 LET S = S + I
40 NEXT I
50 PRINT S
```



40. După executarea următorului program prin care se A F  
compară două numere A și B, pe ecran se va afișa:

A este egal cu B A nu este egal cu B
---

```
10 LET A = 3
20 LET B = 3
30 IF A = B THEN PRINT "A este egal cu B"
40 PRINT "A nu este egal cu B"
```



41. Instrucțiunile INPUT, LET și READ permit citirea A F  
unor valori dintr-o listă.



42. După executarea următorului program prin care se A F  
compară două numere A și B, pe ecran se va afișa:

```
A este egal cu B
A nu este egal cu B
```

```
10 LET A = 3
20 LET B = 3
30 IF A = B THEN PRINT "A este egal cu B":STOP
40 PRINT "A nu este egal cu B"
```



43. După executarea următorului program, pe ecran se va A F  
afișa:

```
A este mai mic ca B
```

```
10 LET A = 3
20 LET B = 3
30 IF A <=> B THEN PRINT "A este mai mic ca B":
PRINT "A este egal cu B":STOP
40 PRINT "A este mai mic sau egal cu B"
```



44. Instrucțiunile cuprinse în următoarea buclă FOR- A F  
NEXT se vor executa de 15 ori:

```
30 FOR I = 1 TO 30
40 PRINT I + 1
50 LET I = I + 1
60 NEXT I
```



45. Instrucțiunile din următoarea buclă se vor executa de A F  
un număr infinit de ori:

```
10 FOR X = 1 TO 10
20 LET X = X - 1
30 NEXT X
```



## Mecanismul hazardului

Descori pentru simularea unor situații sau fenomene este necesară generarea unor numere întâmplătoare (aleatoare). Putem realiza acest lucru prin intermediul calculatorului folosind funcția RND. Aceasta generează numere întâmplătoare, cuprinse între 0 și 1. Este de notat că valoarea 0 poate fi luată nu însă și valoarea 1.



Să se genereze cu ajutorul calculatorului numere aleatoare întregi și pozitive mai mici decât un număr dat.

Desigur va trebui ca numărul să fie înmulțit cu RND iar la rezultat să se aplice funcția întreg (INT). Programul P26 este:

```
P26. 5 REM ** program de generare numere **
      6 REM ** aleatoare intregi si **
      7 REM ** pozitive mai mici ca N **
      10 INPUT N
      20 PRINT INT (RND * N)
      30 GOTO 20
```

De exemplu, dacă dorim să generăm numere aleatoare întregi mai mici decât 20, vom introduce numărul 20 iar apoi:

- ✗ dacă cumva RND ia valoarea 0, atunci se va afișa valoarea 0, care este un număr mulțumitor pentru ceea ce se cere
- ✗ dacă RND ia o valoare apropiată de 1 (să presupunem că este chiar valoarea 1, cu toate că ea nu poate fi atinsă), atunci se va afișa valoarea 20, care este și ea mulțumitoare

× orice altă valoare pe care o ia RND va avea ca rezultat afișarea unui număr întreg cuprins între 1 și 20

Am văzut totuși că însăși valoarea 20 nu o va lua niciodată, cu toate că acest rezultat ar fi și el mulțumitor și n-ar trebui să-l trecem cu vederea. În acest scop vom modifica linia 20 a programului P26 astfel:

```
20 PRINT INT (RND *( N+1))
```

Motivul este lesne de înțeles. De data aceasta nu se va mai atinge în exemplul nostru valoarea 21 care nu ne interesează, în schimb va putea fi atinsă valoarea 20 dar și oricare altă valoare întreagă mai mică de 20, inclusiv 0 în cazul în care RND a luat valoarea 0.



## Un joc cu numere

---



Să se realizeze un program cu care să se poată juca jocul "Ghicește numărul". Calculatorul memorează un număr cuprins între 0 și 100 iar jucătorul trebuie să-l ghicească introducând încercări (numere) la care calculatorul va răspunde "Este prea mare" sau " Este prea mic".

Desigur se va folosi funcția RND pentru a se memora un număr aleator între 0 și 100. Programul P27 va fi:



```
P27. 5 REM ** joc ghiceste numarul **  
10 PRINT "GHICESTE NUMARUL"  
20 LET N = INT (RND * 101)  
30 INPUT X  
40 IF X<N THEN PRINT "Prea mic":GOTO 30  
50 IF X>N THEN PRINT "Prea mare":GOTO 30  
60 PRINT "Ai ghicit !"  
70 PRINT  
80 GOTO 10
```



Să se modifice jocul "Ghicește numărul" astfel încât atunci când numărul este găsit să se indice numărul de încercări din care a fost găsit.

Pentru a ține socoteala (contabilitatea) încercărilor, vom folosi o tehnică similară cu cea utilizată pentru realizarea sumelor cu calculatorul. Vom lua o variabilă (sac) S. La început în ea nu este nimic deci are valoarea 0:

```
25 LET S = 0
```

Apoi, de fiecare dată când se face o încercare vom pune în sac o bobită (adică adăugăm la vechea valoare încă o unitate):

```
35 LET S = S + 1
```

Vom modifica și linia 60 astfel încât să se afișeze numărul de încercări:

```
60 PRINT "Ai ghicit din ";S ; "incercari"
```

Această tehnică este des folosită în special la jocuri pentru ținerea evidenței scorului, pentru ținerea evidenței încercărilor și, de aceea, se mai numește tehnica "*numărării încercărilor*".



## Să-i ajutăm pe cei mici

---



Să se realizeze un program pentru antrenament la tabla înmulțirii.

Vom genera două numere aleatoare întregi cuprinse între 0 și 10 și vom întreba rezultatul înmulțirii lor. Dacă rezultatul este corect se vor genera alte 2 numere și altă întrebare va apare, iar dacă este incorect se va repeta întrebarea și se va introduce altă încercare. Programul P28 este:

**P28.**

```
5 REM ** program pentru antrenament **
6 REM **      la tabla inmultirii      **
10 CLS: PRINT "Tabla Inmultirii"
20 LET A = INT (RND*11)
30 LET B = INT (RND*11)
40 PRINT "Cit fac "; A; " x "; B; " ? "
50 INPUT C
60 IF C=A*B THEN PRINT " Bine": GOTO 20
70 PRINT "Nu e bine": GOTO 40
```

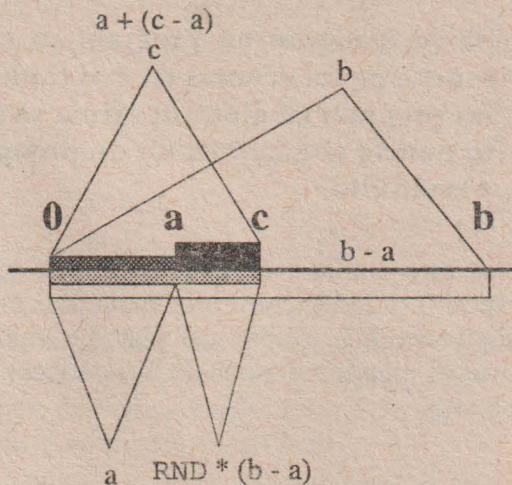


Să se genereze cu ajutorul calculatorului numere aleatoare întregi cuprinse într-un interval dat.

Să presupunem că dorim să generăm numere aleatoare întregi în intervalul  $[a, b]$ .

fig.19

Generarea de numere aleatoare întregi într-un interval dat



Desigur aceste margini vor trebui introduse. Problema este asemănătoare cu precedenta (de generare a unor numere aleatoare întregi și pozitive mai mici decât un număr dat) considerînd faptul că orice număr  $C$  din intervalul  $[A, B]$  are în componența sa valoarea  $A$  la care se adaugă o anumită

parte (depinde de numărul aleator generat) din valoarea  $B-A$  (vezi figura 19). Programul P29 este:

```
P29. 5 REM ** program de generare numere **
      6 REM ** aleatoare intregi cuprinse **
      7 REM **      intr-un interval dat      **
      10 INPUT A,B
      20 PRINT A + INT (RND * ( B - A + 1 ))
      30 GOTO 20
```

Cînd RND ia valoarea 0 atunci numărul generat va fi A, cînd RND ia o valoare apropiată de 1 atunci numărul generat va lua valoarea  $A + B - A = B$ , iar cînd RND ia orice altă valoare numărul generat va lua o valoare între A și B.

## Cîteva probleme cu numere și grafică

---



Să se realizeze un program cu ajutorul căruia să se deseneze pe tot ecranul cît mai multe puncte. Să se realizeze un program cu ajutorul căruia să se deseneze cît mai multe puncte situate într-un dreptunghi în partea dreaptă sus a ecranului.

Problema este similară cu generarea unor numere aleatoare întregi cuprinse între 0 și 255 (pentru orizontala punctului) și între 0 și 175 (pentru verticala punctului), pentru calculatoare compatibile Sinclair Spectrum, aceste valori menținînd punctele în interiorul ecranului grafic. Programul P30.A va fi:

```
P30.A 5 REM ** program de desenare puncte **
      6 REM **      pe ecran      **
      7 REM **      pentru calculator HC      **
      10 PLOT INT(RND * 256), INT(RND * 176)
      20 GOTO 10
```



Pentru calculatoare PC programul similar este P30.B. În acest caz numerele aleatoare întregi vor fi cuprinse între 0 și 320 (pentru orizontală) și 0 și 200 (pentru verticală):

```
P30.B 5 REM ** program de desenare puncte **
        6 REM ** pe ecran **
        7 REM ** pentru calculator PC **
        8 CLS
        9 SCREEN 1
       10 PSET (INT(RND*320),INT(RND*200)),7
       20 GOTO 10
```

În al doilea caz problema este similară tot cu generarea unor numere aleatoare cuprinse într-un interval dat. Pentru a se încadra într-un dreptunghi situat în partea dreaptă sus a ecranului să presupunem că orizontala punctelor va fi cuprinsă între limitele 170 și 220, iar verticala între 110 și 140. Programul P31 va fi:

```
P31.A 5 REM ** program de desenare **
        6 REM ** puncte într-un dreptunghi **
        7 REM ** pentru calculator HC **
       10 PLOT 170+INT(RND*51),110+INT(RND*31)
       20 GOTO 10
```

Valorile 51 și 31 s-au calculat astfel:

$$220 - 170 + 1 = 51$$

și

$$140 - 110 + 1 = 31$$

Pentru calculatoare PC programul similar este P31.B. Dacă se păstrează aceleași valori pentru coordonate ca în cazul programului pentru HC, atunci se formează un dreptunghi în partea dreaptă jos a ecranului deoarece acum punctul de origine este în colțul din stînga sus.

```
P31.B 5 REM ** program de desenare **
        6 REM ** puncte într-un dreptunghi **
        7 REM ** pentru calculator PC **
        8 CLS
```

```

9 SCREEN 1
10 PSET(170+INT(RND*51),
      110+INT(RND*31)),7
20 GOTO 10

```



Să se facă un program care să afișeze în diverse locuri pe ecran litera a. Apariția literei a va fi însoțită de o notă muzicală întîmplătoare.

Fiind vorba de afișarea unui caracter va trebui să generăm numere aleatoare întregi cuprinse între 0 și 21 (pentru a indica linia pe care se va afișa caracterul) și între 0 și 31 pentru coloana pe care se va afișa caracterul. Nota muzicală poate avea aceeași durată, dar înălțimea va fi aleatoare. Programul P32.A pentru calculatoare HC este:

```

P32.A 5 REM ** program pentru afisarea **
      6 REM **       unei litere       **
      7 REM **       pentru calculator HC **
      10 PRINT AT INT(RND*22),INT(RND*32);"a"
      20 BEEP .5, INT (RND * 70)
      30 CLS
      40 GOTO 10

```

Programul calculatoare PC programul similar este P32.B. S-a indicat modul text cu 40 de coloane (SCREEN 0) și 25 de rînduri linii numerotate de la 1.

```

P32.B 5 REM ** program pentru afisarea **
      6 REM **       unei litere       **
      7 REM **       pentru calculator PC **
      8 CLS
      9 SCREEN 0
      10 LOCATE 1+INT(RND*24), 1+INT(RND*39)
      20 PRINT "a"
      30 CLS
      40 GOTO 10

```



Să se genereze numere aleatoare întregi de N cifre.

Problema este similară generării numerelor aleatoare întregi într-un interval, dacă se observă că cel mai mic număr de N cifre (corespunzător limitei stînga a intervalului) este  $10^{(N-1)}$  iar cel mai mare număr de N cifre (corespunzător limitei dreapta a intervalului) este  $10^N - 1$ . Într-adevăr, pentru numere întregi de 3 cifre cel mai mic număr este 100 care este egal cu  $10^{(3-1)} = 10^2$ , iar cel mai mare număr este 999 care este egal cu  $10^3 - 1$ . Programul P33 va fi :

```
P33. 5 REM ** program de generare numere **
      6 REM **   aleatoare intregi   **
      7 REM **       de N cifre       **
      10 INPUT N
      20 PRINT 10^(N-1)+INT(RND*((10^N-1)-
          10^(N-1)+1))
      30 GOTO 20
```



## Probleme cu căutări și inversiuni



Să se genereze un șir de  $N$  numere aleatoare întregi pozitive mai mici ca 100, apoi cel mai mare număr să-și schimbe poziția cu cel mai mic și să se afișeze șirul.

După rezervarea spațiului de memorie pentru  $N$  numere și generarea numerelor va trebui să aflăm cel mai mare dintre numere (*maximul*) și, respectiv, cel mai mic (*minimul*).

Pentru aflarea maximului procedăm astfel: considerăm că maximul este chiar primul număr (poziția C a maximului fiind prima) și îl comparăm succesiv cu fiecare din numerele din șir. La o comparare dacă numărul cu care comparăm maximul este mai mic atunci se trece la următoarea comparare (cu numărul următor), dar dacă la un moment dat numărul din șir este mai mare decât maximul, atunci îl luăm pe el ca fiind maximul. În acest mod după compararea maximului cu toate numerele din șir, în final, în variabila care memorează maximul se va afla chiar cel mai mare număr din șir.

Pentru aflarea minimului procedăm invers față de modul de aflare a maximului (operațiile de aflare a minimului și maximului se vor realiza în paralel): considerăm că cel mai mic număr este chiar primul (poziția D fiind și ea prima), apoi, în mod asemănător comparăm acest minim, succesiv, cu toate numerele din șir. Dacă găsim un număr mai mic decât minimul îl luăm pe el ca minim. În final în variabila minim vom regăsi cel mai mic număr din șir.

Avînd minimul și maximul va trebui să le schimbăm între ele. În acest scop vom reține în prealabil atît poziția minimului cît și cea a maximului. Inversarea celor două numere se va face prin **metoda celor trei pahare**. Conform acestei metode, pentru a schimba între ele conținutul diferit a două pahare, avem nevoie de un al treilea pahar (gol) care va avea rolul de a primi (memora) conținutul unuia dintre cele două pahare. Acesta golindu-se, va putea "primi" conținutul celui de-al doilea pahar, care, la rîndul său golit va putea "primi" conținutul păstrat doar temporar în cel de-al 3-lea pahar.

Programul P34 va fi:

```
P34. 5 REM ** program 1 cu cautari **
      6 REM ** si inversiuni de numere **
      10 INPUT N
      20 DIM A(N)
      30 FOR I = 1 TO N
      40 LET A(I) = INT (RND*100)
      50 NEXT I
      60 LET MAX = A(1) : LET C = 1
      70 LET MIN = A(1) : LET D = 1
      80 FOR I = 2 TO N
      90 IF MAX < A(I) THEN LET MAX = A(I) :
          LET C = I
     100 IF MIN > A(I) THEN LET MIN = A(I) :
          LET D = I
     110 NEXT I
     120 LET P = A(C)
     130 LET A(C) = A(D)
     140 LET A(D) = P
     150 FOR I = 1 TO N
     160 PRINT A(I); " ";
     170 NEXT I
     180 PRINT
```



Să se realizeze un program cu ajutorul căruia să se caute un element dintr-o mulțime (de exemplu, un număr dintr-o listă de numere) și, dacă se găsește, să se treacă la sfârșitul listei în locul ultimului, toate celelalte elemente situate după el mutându-se înapoi cu o poziție.

Va trebui să introducem câte numere sînt în listă și să dimensionăm un spațiu de memorie pentru aceste numere. Apoi vom introduce numerele respective  $A(I)$ .

În exemplul dat vom genera numere aleatoare întregi cuprinse, de exemplu, între 0 și 100. Vom indica și numărul pe care vom încerca să îl găsim introducînd, deci, un număr întreg  $X$  cuprins între 0 și 100. Căutarea elementului se va realiza prin compararea numărului  $X$  cu fiecare număr  $A(I)$  din listă.

Acest tip de căutare se mai numește **lineară** și reprezintă singura metodă de căutare a unui element dintr-o mulțime de elemente care s-a format aleator.

Dacă lista de elemente ar fi fost generată după o anumită regulă, de exemplu, dacă numerele ar fi fost puse în ordine crescătoare, atunci s-ar fi putut utiliza o metodă mai eficientă (de exemplu, cea prin care se încearcă elementul situat în mijlocul listei, metodă care se mai numește **binară**, deoarece se bazează pe împărțirea listei în două subliste).

Dacă prin parcurgerea întregii liste nu se găsește nici un element identic cu cel introdus aceasta înseamnă că elementul căutat nu se găsește în listă și procesul se va repeta, adică, se va introduce de la tastatură alt număr pe care ne propunem să îl găsim. Dacă elementul este găsit atunci se memorează în variabila  $C$  poziția  $I$  în care se găsea elementul și se va începe procesul de mutare a elementelor care se găsesc după  $X$  (adică sînt în dreapta lui) cu o poziție spre stînga. Practic se va trece conținutul unei variabile (locație de memorie) aflate în poziția  $I+1$  în variabila (locația) din poziția  $I$ . Nu trebuie să ne temem de pierderea conținutului variabilei din poziția  $I$ , deoarece acesta este reprezentat de valoarea  $A(I)$ , pe care o cunoaștem, fiind egală cu valoarea  $X$ . Apoi prin transferul conținutului a

cîte unei variabile din dreapta în variabila situată cu o poziție spre stînga se eliberează locația pentru următorul transfer.

În final nu uităm să punem și pe ultima poziție  $N$ , valoarea elementului  $X$  și apoi să afișăm lista astfel obținută (vezi figura 20 în care se indică și ordinea logică de executare a operațiilor).

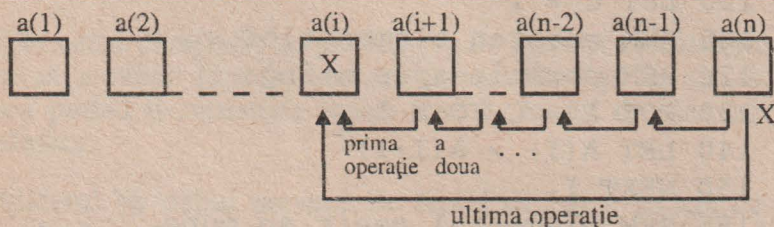


fig.20

Reprezentarea grafică cu indicarea ordinii logice a operațiilor pentru găsirea unui element dat într-o listă de elemente și trecerea sa la sfîrșitul listei

Programul P35 este:

**P35**

```

5 REM ** program 2 cu cautari **
6 REM ** si inversiuni de numere **
10 INPUT N
20 DIM A(N)
25 REM generare numere intregi si
26 REM pozitive mai mici
27 REM sau egale cu 100
30 FOR I = 1 TO N
40 LET A(I) = INT (RND * 101)
50 NEXT I
55 REM introducere element de cautat
60 INPUT X
65 REM cautare element
70 FOR I = 1 TO N
80 IF A(I) = X THEN GOTO 120
90 NEXT I
95 REM elementul cautat nu e in lista

```

```
100 PRINT "Elementul ";X; " nu se afla in
    lista. Introduceți pentru cautare
    alt element"
110 GOTO 60
115 REM elementul a fost gasit. Se
116 REM memoreaza pozitia lui in lista
120 LET C = I
125 REM mutarea elementelor cu o
126 REM pozitie spre stanga
130 FOR I = C TO N-1
140 LET A(I) = A(I+1)
150 NEXT I
155 REM elementul gasit se pune
156 REM pe ultima pozitie
160 LET A(N) = X
165 REM se afiseaza lista obtinuta
170 FOR I = 1 TO N
180 PRINT A(I) ; " ";
190 NEXT I
200 PRINT
```





## Probleme cu cuvinte

Ca și în cazul numerelor se pot rezolva cu ajutorul calculatorului și probleme cu cuvinte (sau în termeni informatici "șiruri de caractere"), acestea putând fi memorate (adică puse în locații de memorie) în mod asemănător.

Calculatorul va trebui prevenit de faptul că într-o locație de memorie urmează să se "pună" un cuvânt iar acest lucru se face prin numele atribuit locației (variabilei) care trebuie să fie o literă urmată de semnul dolarului (\$). Spunem că este o **variabilă alfanumerică** sau de **tip șir de caractere**. Deosebirea lucrului cu cuvinte față de cel cu numere constă în faptul că atunci când se rezervă un spațiu de memorie pentru cuvinte va trebui indicat nu numai numărul maxim de cuvinte care se intenționează a se memora ci și mărimea cuvântului maxim (din câte caractere va fi format cel mai lung cuvânt).

Să presupunem că citim cu o instrucțiune READ mai multe nume de familie (și anume 12) preluate dintr-o linie DATA, iar cel mai lung dintre nume presupunem (inițial) că nu va depăși 10 caractere (litere). Într-o astfel de linie după cuvântul cheie DATA se trec datele (valori, șiruri de caractere) care urmează a fi citite, unele după altele separate prin virgulă.

Printr-o instrucțiune DIM A\$ (12,10) vom rezerva un spațiu de memorie de 12 cuvinte (câte au fost citite din linia DATA) formate din maxim 10 caractere.

În figura 21 se poate observa cum va arăta memoria în cazul în care linia DATA arată astfel:

```
DATA "Popescu", "Popa", "Pamfil", "Anghel",  
"Protopopescu", "Paraschiv", "Papagheorghe",  
"Avram", "Nicolae", "Nistorescu", "Nitescu",  
"Ionescu"
```

fig.21

Reprezentarea unui spațiu de memorie prin utilizarea unei variabile alfanumerice bidimensionale pentru memorarea unor nume

a\$ (1)	Popescu			
a\$ (2)	Popa			
a\$ (3)	Pamfil			
a\$ (4)	Anghel			
a\$ (5)	Protopopes			
a\$ (6)	Paraschiv			
a\$ (7)	Papagheorg			
a\$ (8)	Avram			
a\$ (9)	Nicolae			
a\$ (10)	Nistorescu			
a\$ (11)	Nitescu			
a\$ (12)	Ionescu			

Se observă că toate cuvintele vor avea, de fapt, aceeași lungime pe care am declarat-o, și anume 10 caractere:

- ✗ dacă șirul de caractere inițial avea mai puțin de 10 caractere atunci locația de memorie se "umple" cu spații goale (blancuri) până la a 10-a poziție. Este cazul numelor Popescu, Popa, Pamfil, Anghel, Paraschiv, Avram, Nicolae, Nițescu și Ionescu
- ✗ dacă șirul de caractere inițial avea 10 caractere atunci el intră normal în locația de memorie. Este cazul numelui Nistorescu
- ✗ dacă șirul de caractere inițial avea mai mult de 10 caractere, atunci vor intra în locație numai primele 10 caractere, urmînd ca restul să nu mai fie memorate. Este cazul numelor Protopopescu și Papagheorghe. Spunem că acest fel de memorare este de **tip procustian**.

În legenda "Patul lui Procut", acesta tăia, după dimensiunea patului său, din picioarele oamenilor mai înalți care nu încăpeau în pat. În mod similar caracterele (literele) care nu mai au loc în locația de memorie dimensionată la o anumită lungime și o depășesc, nu se vor mai memora (vor fi "tăiate").

În ceea ce privește compararea cuvintelor în vederea punerii lor în ordine alfabetică, acest lucru se va realiza de către calculator în mod similar cu compararea numerelor și punerea acestora în ordine crescătoare (sau des-

crescătoare) deoarece caracterele sînt memorate sub forma codului ASCII, care este un cod numeric.



Să se realizeze un program prin care să se citească mai multe nume de familie apoi să se introducă de la tastatură o literă mare. Să se afișeze toate numele care încep cu acea literă.

Explicațiile pentru realizarea acestui program au fost deja date. Va trebui să rezervăm un spațiu de memorie de N locații și să imaginăm din câte litere este format cel mai mare nume. Dacă un nume nu va încăpea în locație acest lucru nu ne deranjează prea mult deoarece ne interesează numai începutul cuvîntului nu și sfîrșitul său. Primul caracter din fiecare nume se va compara cu litera introdusă X\$ și dacă acestea sînt identice atunci se va afișa numele. Programul P36.A pentru calculatoarele Sinclair Spectrum este:

```
P36.A 5 REM ** program cu care se pot **
      6 REM ** afisa numele care incep **
      7 REM ** cu o anumita litera **
      8 REM ** pentru calculator HC **
      10 INPUT N
      20 DIM A$(N,10)
      30 FOR I = 1 TO N
      40 READ A$(I)
      50 PRINT A$(I)
      60 NEXT I
      70 PRINT
      80 INPUT X$
      90 FOR I = 1 TO N
      100 IF A$(I,1) = X$ THEN PRINT A$(I)
      110 NEXT I
      120 PRINT
      130 GOTO 80
```

140 DATA "Popescu", "Popa", "Pamfil",  
"Anghel", "Protopopescu",  
"Paraschiv", "Papagheorghe",  
"Avram", "Nicolae", "Nistorescu",  
"Nitescu", "Ionescu"

Dacă introducem litera P obținem:

Popescu  
Popa  
Pamfil  
Protopopescu  
Paraschiv  
Papagheorghe

iar dacă introducem litera A obținem:

Anghel  
Avram



**Să se realizeze un program prin care să se introducă mai multe nume de familie apoi să se introducă de la tastatură o literă și să se afișeze toate numele care se termină cu acea literă.**

Pornind de la programul precedent pentru HC și avînd dimensionat un spațiu de memorie de N cuvinte a cîte 10 caractere practic am declarat fiecare cuvînt ca avînd o lungime de 10 caractere. În acest caz putem folosi funcția LEN pentru a memora lungimea fiecărui cuvînt (și astfel a identifica a cîta literă este ultima) deoarece toate cuvintele sînt de lungime 10. Nu putem face altceva decît să folosim un artificiu și, anume, să cercetăm locul (poziția) în cadrul cuvîntului unde apare primul spațiu liber (blancul), iar ultimul caracter al cuvîntului va fi precedentul. Folosind această metodă va trebui să avem însă grijă ca să dăm o lungime în orice caz acoperitoare pentru orice nume (de exemplu 20) astfel încît să fim siguri că în cadrul numelor (la coadă) va apare cel puțin un spațiu liber. Programul P37.A (numai pentru calculatoare HC) este:

```

P37.A 5 REM ** program pentru afisarea **
        6 REM ** numelor care se termina **
        7 REM ** cu o anumita litera **
        8 REM ** pentru calculator HC **
        10 INPUT N
        20 DIM A$(N,20)
        30 FOR I = 1 TO N
        40 INPUT A$(I)
        50 PRINT A$(I)
        60 NEXT I
        70 PRINT
        80 INPUT X$
        90 FOR I = 1 TO N
        100 FOR J = 1 TO 20
        110 IF A$(I,J) = " " THEN GOTO 130
        120 NEXT J
        130 IF A$(I,J-1) = X$ THEN PRINT A$(I)
        140 NEXT I
        150 PRINT
        160 GOTO 80

```

Altă metodă este să folosim totuși funcția LEN pentru a identifica ultima literă a cuvântului. Pentru aceasta se va introduce numele fără a se rezerva un spațiu de memorie și se va memora lungimea numelui. Apoi se va trece conținutul variabilei care memorează numele A\$ în altă variabilă B\$ pentru care s-a declarat un spațiu de memorie și o lungime suficientă. Fiecare ultimă literă a variabilei B\$ adică B\$(I, L(I)) se va compara cu litera căutată. Programul P38 este:

```

P38.A 5 REM ** program pentru afisarea **
        6 REM ** numelor care se termina **
        7 REM ** cu o anumita litera **
        8 REM ** folosind functia LEN **
        9 REM ** pentru calculator HC **
        10 INPUT N
        20 DIM B$(N,20) : DIM L(N)
        30 FOR I = 1 TO N

```

```

40 INPUT A$
50 PRINT A$
60 LET L(I) = LEN A$
70 LET B$(I) = A$
80 NEXT I
90 PRINT
100 INPUT X$
110 FOR I = 1 TO N
120 IF B$(I,L(I)) = X$ THEN PRINT B$(I)
130 NEXT I
140 PRINT
150 GOTO 100

```

Programul similar pentru calculatoare PC este P38.B:

```

P38.B 5 REM  ** program pentru afisarea  **
6 REM  **  numelor care se termina  **
8 REM  **    cu o anumita litera    **
9 REM  **  folosind functia LEN      **
10 REM **  pentru calculator PC     **
11 INPUT N
20 DIM B$(N) : DIM L(N)
30 FOR I = 1 TO N
40 INPUT A$
50 PRINT A$
60 LET L(I) = LEN (A$)
70 LET B$(I) = A$
80 NEXT I
90 PRINT
100 INPUT X$
110 FOR I = 1 TO N
120 IF MID$( B$(I),L(I),1) = X$ THEN
    PRINT B$(I)
130 NEXT I
140 PRINT
150 GOTO 100

```



Să se realizeze un program cu care să se facă un catalog, adică numele să fie trecute în ordine alfabetică.

După introducerea celor  $N$  nume de maximum  $M$  caractere se procedează la fel ca la ordonarea numerelor, comparându-se numele între ele pe perechi. Indicele  $I$  arată a câta trecere prin șirul de elemente se face, iar indicele  $J$  arată poziția cuvântului (al câtelea nume este). Vor fi necesare maximum  $N-1$  treceri. Dacă numele dintr-o pereche nu sînt în ordine bună, adică primul este mai mare decît al doilea, atunci se schimbă ordinea între ele, aplicîndu-se metoda celor trei pahare la fel ca la numere. Programul P39.A este:

```
? P39.A 5 REM ** program pentru afisare **
6 REM ** in ordine alfabetica **
7 REM ** pentru calculator HC **
10 INPUT N , M
20 DIM A$(N,M)
30 FOR I = 1 TO N
40 INPUT A$(I)
50 PRINT A$(I)
60 NEXT I
70 PRINT
80 FOR I = 1 TO N-1
90 FOR J = 1 TO N-I
100 IF A$(J) <= A$(J+1) THEN GOTO 140
110 LET B$ = A$(J)
120 LET A$(J) = A$(J+1)
130 LET A$(J+1) = B$
140 NEXT J
150 NEXT I
160 FOR I = 1 TO N
170 PRINT A$(I)
180 NEXT I
190 PRINT
```



## 7 probleme pentru funcția de gradul I




**Problema 1.** (puncte) în partea de jos a ecranului grafic al calculatorului (este de fapt axa OX).


*Rezolvare:* deoarece lungimea liniei este de 256 de pixeli, variabila de ciclare va avea limitele 0 și 255. Se va reprezenta grafic funcția

$$y = 0$$

Pentru trasarea punctelor se va modifica coordonata pe orizontală (X) care va crește de la 0 la 255, în timp ce coordonata pe verticală (Y) va rămâne constantă (0). Pentru o înțelegere mai bună, să notăm și în program variabila pe orizontală (abscisa) cu X, iar cea care reprezintă coordonatele pe verticală (ordonata) cu Y. Programul P40.A va fi:

```
 P40.A 5 REM ** program de trasare a unui **
6 REM ** segment prin puncte **
7 REM ** pentru calculatorul HC **
10 FOR X = 0 TO 255
20 PLOT X , 0
30 NEXT X
```

Pentru calculatoare PC programul similar este P40.B în care se ține seama de dimensiunile ecranului grafic:

```
 P40.B 5 REM ** program de trasare a unui **
6 REM ** segment prin puncte **
7 REM ** pentru calculatorul PC **
```



```

8 SCREEN 1
9 CLS
10 FOR X = 0 TO 319
20 PSET (X , 199) , 7
30 NEXT X

```



**Problema 2.** Să se deseneze prin puncte un segment de 100 pixeli la înălțimea de 80 de pixeli față de partea de jos a ecranului grafic.

*Rezolvare:* variabila de ciclare va avea limitele 1 și 100. Practic se va reprezenta dreapta care este graficul funcției

$$y = 80$$

Programul P41.A va fi:

```

P41.A 5 REM ** program de trasare a unui **
6 REM ** segment prin puncte **
7 REM ** la o anumita inaltime **
8 REM ** pentru calculatorul HC **
10 FOR X = 1 TO 100
20 PLOT X , 80
30 NEXT X

```

Pentru calculatoare PC programul similar care trasează o dreaptă la înălțimea de 50 pixeli de partea de jos a ecranului este P41.B:

```

P41.B 5 REM ** program de trasare a unui **
6 REM ** segment prin puncte **
7 REM ** la o anumita inaltime **
8 REM ** pentru calculatorul PC **
9 SCREEN 1
10 CLS
11 FOR X = 0 TO 319
20 PSET (X , 150) , 7
30 NEXT X

```



**Problema 3.** Să se deseneze prin puncte o linie cu lungimea de 100 de pixeli pe verticală începînd de la înălțimea de 50 de pixeli față de partea de jos din stînga a ecranului.

*Rezolvare:* variabila de ciclare va avea limitele 50 și 150. Prima limită are valoarea 50 deoarece segmentul începe să se traseze de la înălțimea de 50 de pixeli. A doua limită se obține prin adăugarea la valoarea inițială (50) a mărimii segmentului (creșterea pe verticală) care este 100. Linia desenată se va situa pe dreapta care este graficul funcției

$$x = 50$$

Programul P42.A este :

```

P42.A 5 REM ** program de trasare a unui **
      6 REM ** segment pe verticala prin **
      7 REM **           puncte           **
      8 REM ** pentru calculatorul HC    **
      9
     10 FOR Y = 50 TO 50+100
     20 PLOT 50 , Y
     30 NEXT Y
  
```

Pentru calculatoare PC programul similar este P42.B

```

P42.B 5 REM ** program de trasare a unui **
      6 REM ** segment pe verticala prin **
      7 REM **           puncte           **
      8 REM ** pentru calculatorul PC    **
      9
     10 SCREEN 1
     20 CLS
     30 FOR Y=200-50 TO (200-50)-100 STEP -1
     40 PSET (50 , Y) , 7
     50 NEXT Y
  
```





Să se deseneze prin puncte cel mai mare dreptunghi care

**Problema 4.** se poate desena pe ecranul grafic al calculatorului.

*Rezolvare:* prima latură a dreptunghiului o avem deja prin programul de la problema 1. Pentru dreapta următoare, abscisele punctelor vor fi aceleași (255), în timp ce ordonatele vor varia de la 0 la 175, adică atât cât permit mărimile coordonatelor ecranului pe înălțime, rezultând o dreaptă cu lungimea de 176 de pixeli. Liniile în continuarea programului P42.A sînt :

```
40 FOR Y = 0 TO 175
50 PLOT 255 , Y
60 NEXT Y
```

Pentru următoarea dreaptă și anume lungimea dreptunghiului situată în partea superioară a ecranului, punctul de pornire este cel corespunzător coordonatelor  $X = 255$  și  $Y = 175$ . Ordonata va rămîne constantă, iar cea pe orizontală va scădea de la 255 la 0 (deci STEP -1). Programul în continuare este:

```
70 FOR X = 255 TO 0 STEP -1
80 PLOT X , 175
90 NEXT X
```

În sfîrșit, pentru ultima latură, de data aceasta, abscisa va rămîne constantă ( $X = 0$ ), în timp ce, ordonata va scădea de la 175 la 0. Vom adăuga programului P42.A liniile :

```
100 FOR Y = 175 TO 0 STEP -1
110 PLOT 0 , Y
120 NEXT Y
```

Pentru ca programul să meargă mai "frumos" se pot insera instrucțiuni BEEP după fiecare punct care se înscrie pe ecran, deci la numerele de linie 25, 55, 85 și 115 :

```
BEEP 0.01,0
```

Se observă că durata sunetului este foarte mică, iar nota este **do**.





**Problema 5.** Să se deseneze prin puncte cel mai mare pătrat care se poate înscrie pe ecranul grafic al calculatorului și apoi să se traseze și diagonalele sale.

*Rezolvare:* deoarece laturile pătratului sînt egale, cel mai mare pătrat care se poate desena va avea latura egală cu 176 de pixeli. Programul P43.A este :

```
P43.A 5 REM ** program de desenare prin **
6 REM ** puncte a celui mai mare **
7 REM ** patrat care se poate **
8 REM ** inscrie in ecranul grafic **
9 REM ** pentru calculatorul HC **
10 FOR X = 0 TO 175
20 PLOT X , 0
30 NEXT X
40 FOR Y = 0 TO 175
50 PLOT 175 , Y
60 NEXT Y
70 FOR X = 175 TO 0 STEP -1
80 PLOT X , 175
90 NEXT X
100 FOR Y = 175 TO 0 STEP -1
110 PLOT 0 , Y
120 NEXT Y
```

Față de programul pentru dreptunghi se observă că diferă practic doar liniile de program 10 și 70 care dau o altă dimensiune maximă abscisei (176 față de 256).

Programul similar pentru calculatoare PC este P43.B:

```
P43.B 5 REM ** program de desenare prin **
6 REM ** puncte a celui mai mare **
7 REM ** patrat care se poate **
8 REM ** inscrie in ecranul grafic **
9 REM ** pentru calculatorul PC **
10 SCREEN 1
```

```

11 CLS
15 FOR X = 0 TO 199
20 PSET (X , 199) , 7
30 NEXT X
40 FOR Y = 199 TO 0 STEP -1
50 PSET (199 , Y) , 7
60 NEXT Y
70 FOR X = 199 TO 0 STEP -1
80 PSET (X , 0) , 7
90 NEXT X
100 FOR Y = 0 TO 199
110 PSET (0 , Y) , 7
120 NEXT Y

```

Programul se poate scrie și mai concis, desenându-se în același timp câte 2 laturi și folosind o singură variabilă pentru ciclare (programul P44.A și respectiv P44.B pentru PC):

```

P44.A 5 REM ** program de desenare prin **
6 REM ** puncte a celui mai mare **
7 REM ** patrat care se poate **
8 REM ** inscrie in ecranul grafic **
9 REM ** varianta imbunatatita **
10 REM ** pentru calculatorul HC **
15 FOR X = 0 TO 175
20 PLOT X , 0
30 PLOT 175 , X
40 NEXT X
50 FOR X = 175 TO 0 STEP -1
60 PLOT X , 175
70 PLOT 0 , X
80 NEXT X

```

```

P44.B 5 REM ** program de desenare prin **
6 REM ** puncte a celui mai mare **
7 REM ** patrat care se poate **
8 REM ** inscrie in ecranul grafic **

```

```

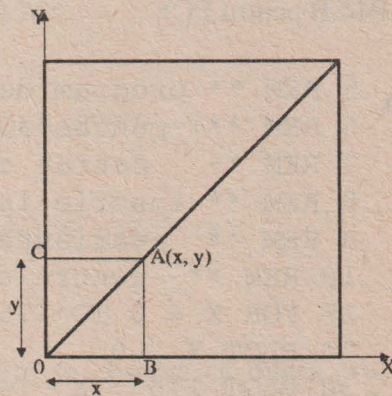
9 REM **   varianta imbunatatita   **
10 REM **  pentru calculatorul PC   **
11 SCREEN 1
12 CLS
15 FOR X = 0 TO 199
20 PSET (X , 199) , 7
30 PSET (199 , X) , 7
40 NEXT X
50 FOR X = 199 TO 0 STEP - 1
60 PSET (X , 0) , 7
70 PSET (0 , X) , 7
80 NEXT X

```

Pentru prima diagonală se pune problema coordonatelor unui punct de pe diagonala unui pătrat (vezi figura 22).

fig.22

*Coordonatele unui punct de pe diagonala pătratului*



Să luăm la îndeplinire un punct A de pe diagonala, de coordonate X și Y. Ne interesează să exprimăm coordonata Y în funcție de coordonata X sau, cu alte cuvinte, cât este ordonata cînd cunoaștem abscisa (abscisa o cunoaștem, deoarece este cea care va lua valori 0,1 ... 100, conform variabilei contor). Fiind vorba de un pătrat, *punctele de pe diagonală sînt egal depărtate de laturile pătratului* ( $AB=AC$ ). Dar  $AC = OB$  și  $AB = OC$ , iar OB este abscisa, iar OC este ordonata. Deci ordonata (Y) va fi egală cu abscisa (X). Diagonala se va situa pe dreapta reprezentată de funcția

$$y = x$$

În cazul nostru, pentru diagonală, coordonata pe orizontală crește de la 0 la 175 (FOR X = 0 TO 175).

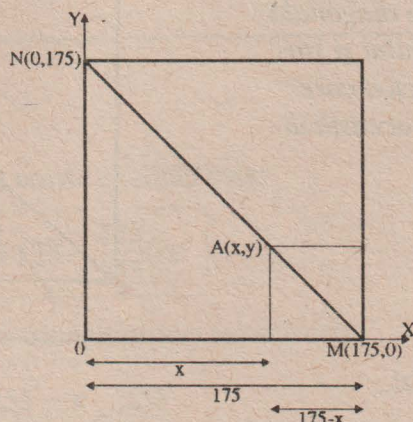
Programul P44.A se completează cu liniile:

```
90 FOR X = 0 TO 175
100 PLOT X , X
110 NEXT X
```

Acum să trasăm și cealaltă diagonală care va porni, de exemplu, din colțul pătratului de coordonate (175,0) și va ajunge în colțul de coordonate (0,175). Pe ce dreaptă se va situa această diagonală? (vezi figura 23).

fig.23

Reprezentarea grafică a diagonalei pătratului



O bună metodă pentru identificarea unei funcții este să cercetăm extremitățile acesteia. Astfel, în punctul M :  $x = 175$  și  $y = 0$ , iar în punctul N :  $x = 0$  și  $y = 175$ . Ecuația care verifică aceste egalități este

$$y = 175 - x$$

deoarece pentru  $y = 0$  obținem  $x = 175$ , iar pentru  $x = 0$  obținem  $y = 175$ . În cazul nostru, abscisa va scădea de la 175 la 0. Programul P44 se completează cu:

```
120 FOR X = 175 TO 0 STEP - 1
130 PLOT X , 175 - X
140 NEXT X
```

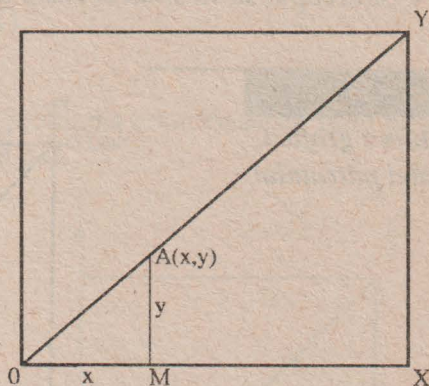


Să se deseneze prin puncte diagonalele dreptunghiului realizat la problema 4.

*Rezolvare:* problema care se pune, de fapt, este aceea de a se identifica funcția a cărei reprezentare grafică este dreapta pe care se situează diagonală. Dacă luăm un punct A la întâmplare pe diagonală, de coordonate  $x$  și  $y$ , va trebui să îl exprimăm pe  $y$  în funcție de  $x$  (considerând că vom cîmbia valoarea variabilei care exprimă abscisa) (vezi figura 24). Se observă că se formează un triunghi OAM asemenea cu triunghiul OYX (deoarece  $AM \parallel YX$ ).

fig.24

Coordonatele unui punct de pe diagonală dreptunghiului și stabilirea funcției care reprezintă această diagonală



În acest caz:

$$\frac{x}{y} = \frac{OX}{OY} \text{ adică } \frac{x}{y} = \frac{255}{175}$$

iar din această relație îl putem scoate pe  $y$  în funcție de  $x$ . Deci:

$$y = \frac{175}{255} \cdot x$$

În consecință, toate punctele de pe diagonală vor avea ordonatele egale cu  $175/255$  din valorile absciselor corespunzătoare. Continuarea programului P42 de la problema 4 va fi:

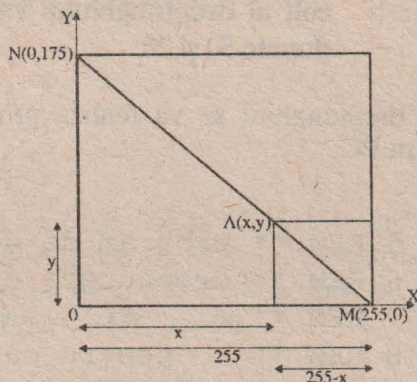
```
130 FOR X = 0 TO 255
140 PLOT X , X * 175/255
150 NEXT X
```



Pentru cealaltă diagonală să cercetăm extremitățile ei (vezi figura 25):

fig.25

Stabilirea funcției care reprezintă diagonală opusă a dreptunghiului



În punctul M:  $x = 255$  și  $y = 0$ ;

În punctul N:  $x = 0$  și  $y = 175$

Funcția de gradul I care verifică aceste egalități este:

$$y = 175 - \frac{175}{255} \cdot x$$

deoarece din figură (asemănarea triunghiurilor) rezultă că:

$$\frac{255 - x}{y} = \frac{255}{175}$$

de unde:

$$y = 175 - \frac{175}{255} \cdot x$$

În cazul nostru abscisa scade de la 255 la 0, iar programul se obține completându-se P42.A cu liniile:

```
160 FOR X = 255 TO 0 STEP -1
170 PLOT X , 175 - X * 175/255
180 NEXT X
```





**Problema 7.** Să se deseneze prin puncte un dreptunghi ale cărei laturi sînt 100 și 70 și apoi să se traseze diagonalele sale. Un colț al dreptunghiului va fi situat în punctul de coordonate 50 și 20.

*Rezolvare:* dreptunghiul se va realiza prin programul P45.A, respectiv P45.B pentru PC:

```
P45.A 5 REM ** program de desenare a unui **
6 REM ** dreptunghi cu diagonalele **
7 REM ** sale **
8 REM ** pentru calculator HC **
10 FOR X = 50 TO 100+50
20 PLOT X , 20
30 NEXT X
40 FOR Y = 20 TO 20+70
50 PLOT 150 , Y
60 NEXT Y
70 FOR X = 150 TO 50 STEP -1
80 PLOT X , 90
90 NEXT X
100 FOR Y = 90 TO 20 STEP -1
110 PLOT 50 , Y
120 NEXT Y
```

```
P45.B 5 REM ** program de desenare a unui **
6 REM ** dreptunghi cu diagonalele **
7 REM ** sale **
8 REM ** pentru calculator PC **
9 SCREEN 1
10 CLS
11 FOR X = 50 TO 100+50
20 PSET (X , 20) , 7
30 NEXT X
40 FOR Y = 20 TO 20+70
50 PSET (150 , Y) , 7
```

```

60 NEXT Y
70 FOR X = 150 TO 50 STEP -1
80 PSET (X , 90) , 7
90 NEXT X
100 FOR Y = 90 TO 20 STEP -1
110 PSET (50 , Y) , 7
120 NEXT Y

```

Pentru prima diagonală abscisa va crește de la 50 la 150, iar ordonatele punctelor de pe ea vor fi proporționale cu inversul raportului laturilor (70/100). De asemenea va fi necesar să se verifice condițiile de la extremitățile diagonalei:  $x = 50$  ;  $y = 20$  și  $x = 150$  ;  $y = 90$ .

În acest caz funcția este :

$$y = 20 + (x - 50) \cdot \frac{70}{100}$$

deoarece din figura 26 rezultă că :

$$\frac{x - 50}{y - 20} = \frac{100}{70}$$

Se adaugă programului P45.A liniile :

```

130 FOR X = 50 TO 150
140 PLOT X , 20 + (X-50) * 70/100
150 NEXT X

```

Pentru programul P45.B la linia 140 va fi:

```

140 PSET (X, 20 - (X-50) * 70/100) , 7

```

Pentru cealaltă diagonală, abscisa va scădea de la 150 la 50, iar pentru extremități se vor verifica condițiile:  $x = 150$  ;  $y = 20$  și  $x = 50$  ;  $y = 90$ .

Funcția va fi :

$$x = 90 - (x - 50) \cdot \frac{70}{100}$$

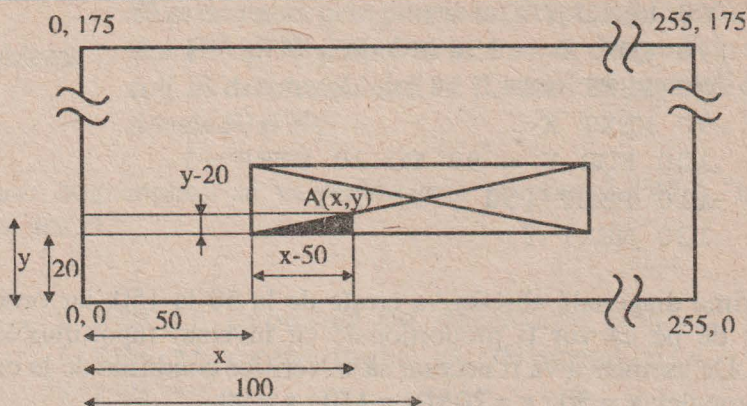


fig.26

Stabilirea funcției care reprezintă diagonală unui dreptunghi de coordonate și poziție date

Programul în continuare a lui P45.A este :

```

160 FOR X = 150 TO 50 STEP -1
170 PLOT X , 90 - (X-50) * 70/100
180 NEXT X

```

Pentru programul P45.B în linia 170 se va folosi PSET în loc de PLOT.



## Reprezentări grafice de funcții

Deseori avem nevoie să reprezentăm grafic diverse funcții. Cazul cel mai frecvent este cel în care avem mai multe puncte în sistemul de reprezentare prin coordonate, cunoscând, deci, câte două valori (corespunzătoare coordonatei pe orizontală și, respectiv, coordonatei pe verticală) și dorim să reprezentăm grafic aceste puncte pentru a putea observa tendința unui fenomen.

De exemplu, să presupunem că avem 10 puncte care reprezintă 10 înregistrări (informații) privind limita superioară a performanțelor calculatoarelor observate între anii 1971-1980. Prin limita superioară a perfor-

manțelor înțelegem viteza maximă atinsă de cele mai puternice calculatoare (supercalculatoarele) existente în anul respectiv, viteză măsurată în milioane de instrucțiuni pe secundă (MIPS).

De exemplu în anul 1971 cel mai puternic calculator existent în lume era STARAN care prezenta o viteză de 70 MIPS. În anul 1972 a fost lansat calculatorul ILIAC IV care îmbunătățea performanțele precedentului cu 5 MIPS. În anul 1973 nu a fost lansat nici un alt calculator care să prezinte performanțe superioare față de ILIAC IV. În anul 1976 firma CRAY a lansat supercalculatorul CRAY 1 care a fost primul calculator care a atins pragul celor 100 MIPS. În anul 1979 firma CDC a lansat calculatorul CYBER 203 care atingea performanțe de pînă la 400 MIPS iar, un an mai tîrziu, CYBER 205 a cărui viteză atingea 700 MIPS.

Performanțele descrise și anii corespunzători sînt oglindite în tabelul de mai jos:

Anii (X)	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980
Viteza – MIPS (Y)	70	75	75	83	83	100	100	200	400	700

Dorim să reprezentăm grafic aceste puncte (pe orizontală vom avea anii, iar pe verticală, vitezele) astfel încît să stabilim ce formă (evoluție) are curba care descrie creșterea vitezei (performanței) calculatoarelor în timp.



**Să se realizeze un program cu care să se poată reprezenta grafic punctele din tabelul reprezentînd creșterea performanțelor calculatoarelor și care exprimă valorile Y în funcție de X.**

Vom avea nevoie să introducem numărul N de puncte (înregistrări) ale căror valori le cunoaștem și să rezervăm un spațiu de memorie suficient pentru abscise – DIM X(N) – și ordinate – DIM Y(N). Apoi va trebui să introducem valorile perechilor X,Y și să le afișăm pe un rînd împreună cu numărul de ordine al înregistrării respective. Introducerea și afișarea (care se face pe măsură ce introducem valorile coordonatelor în vederea verificării corectitudinii introducerii datelor) se va realiza prin intermediul

unui ciclu FOR-NEXT de N pași. Pentru a putea reprezenta grafic punctele va trebui să realizăm o transformare a valorilor reale ale coordonatelor în valori proporționale cu coordonatele ecranului (*factor de scară*). În acest scop va trebui să cunoaștem care sînt valorile minime și maxime ale coordonatelor pe orizontală și respectiv pe verticală (XMIN, XMAX, YMIN, YMAX). Acest lucru îl putem realiza printr-un ciclu de N pași cercetînd valorile pentru fiecare punct. Regula (algoritmul) pentru găsirea maximului și respectiv minimului o cunoaștem deja.

Știind acum valorile minimă și maximă, putem să "punem" pe ecran toate punctele folosind o instrucțiune PLOT, valorile coordonatelor pe ecran fiind:

✕ coordonate pe orizontală:

HC	PC
$\frac{255}{XMAX - XMIN} \cdot (X(I) - XMIN)$	$\frac{319}{XMAX - XMIN} \cdot (X(I) - XMIN)$

✕ coordonate pe verticală:

HC	PC
$\frac{175}{YMAX - YMIN} \cdot (Y(I) - YMIN)$	$200 - \frac{199}{YMAX - YMIN} \cdot (Y(I) - YMIN)$

Programul P46.A este:

```

P46.A 5 REM **      program pentru      **
      6 REM **      reprezentarea grafica **
      7 REM **      a unei functii      **
      8 REM **      pentru calculatorul HC **
      10 INPUT N
      20 DIM X(N): DIM Y(N)
      25 REM introducere valori puncte
      30 FOR I=1 TO N
      40 INPUT "X=";X(I)
      50 INPUT "Y=";Y(I)
      60 PRINT I;" ";X(I);" ";Y(I)
      70 NEXT I
  
```

```

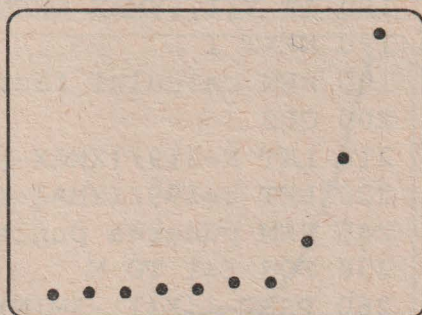
95 REM calculul valorilor maxime si
96 REM minime pentru coordonate
100 LET XMIN=X(1)
110 LET XMAX=X(1)
120 LET YMIN=Y(1)
130 LET YMAX=Y(1)
140 FOR I=2 TO N
150 IF X(I)<XMIN THEN LET XMIN=X(I)
160 IF X(I)>XMAX THEN LET XMAX=X(I)
170 IF Y(I)<YMIN THEN LET YMIN=Y(I)
180 IF Y(I)>YMAX THEN LET YMAX=Y(I)
190 NEXT I
195 REM calculul factorului de scara
200 CLS
210 LET X=255/(XMAX-XMIN)
220 LET Y=175/(YMAX-YMIN)
245 REM punerea punctelor pe ecran
250 FOR I=1 TO N
260 PLOT (X(I)-XMIN)*X, (Y(I)-YMIN)*Y
270 NEXT I

```

Vom obține rezultatul din figura 27 ceea ce ne sugerează o creștere exponențială a performanțelor calculatoarelor în timp.

**fig.27**

*Reprezentarea grafică prin puncte a creșterii performanțelor calculatoarelor electronice în timp*



Pentru calculatoare PC programul similar este P46.B:

**P46.B**

```
5 REM **      program pentru      **
6 REM **      reprezentarea grafica  **
7 REM **      a unei functii      **
8 REM **      pentru calculatorul PC **
9 CLS
10 SCREEN 1 , 0
15 INPUT N
20 X(N): Y(N)
25 REM introducere valori puncte
30 FOR I=1 TO N
40 INPUT "X=";X(I)
50 INPUT "Y=";Y(I)
60 PRINT I;" ";X(I);" ";Y(I)
70 NEXT I
95 REM calculul valorilor maxime si
96 REM minime pentru coordonate
100 LET XMIN=X(1)
110 LET XMAX=X(1)
120 LET YMIN=Y(1)
130 LET YMAX=Y(1)
140 FOR I=2 TO N
150 IF X(I)<XMIN THEN LET XMIN=X(I)
160 IF X(I)>XMAX THEN LET XMAX=X(I)
170 IF Y(I)<YMIN THEN LET YMIN=Y(I)
180 IF Y(I)>YMAX THEN LET YMAX=Y(I)
190 NEXT I
195 REM calculul factorului de scara
200 CLS
210 LET X=319/(XMAX-XMIN)
220 LET Y=199/(YMAX-YMIN)
245 REM punerea punctelor pe ecran
250 FOR I=1 TO N
260 PSET ((X(I)-XMIN)*X,
        199-(Y(I)-YMIN)*Y) , 7
270 NEXT I
```





Cum trebuie să modificăm programul astfel încât să trasăm și liniile dintre puncte?

Desigur vom folosi o instrucțiune DRAW. Începînd cu al doilea punct vom trasa mai întîi dreapta care unește punctul precedent ( $X(I-1)$ ,  $Y(I-1)$ ) cu cel actual ( $X(I)$ ,  $Y(I)$ ), iar pentru aceasta avem nevoie să calculăm creșterile (sau scăderile) pe orizontală și, respectiv, pe verticală. Acestea vor fi:

$$\begin{aligned}dX &= X \cdot (X(I) - XMIN) - X \cdot (X(I-1) - XMIN) = \\ &= X \cdot (X(I) - XMIN - X(I-1) + XMIN) = \\ &= X \cdot (X(I) - X(I-1))\end{aligned}$$

$$\begin{aligned}dY &= Y \cdot (Y(I) - YMIN) - Y \cdot (Y(I-1) - YMIN) = \\ &= Y \cdot (Y(I) - Y(I-1))\end{aligned}$$

Vom insera în programul P46 următoarea linie în program:

```
255 IF I=2 THEN DRAW X*(X(I)-X(I-1)),  
      Y*(Y(I)-Y(I-1))
```

În acest caz nu vom mai avea nevoie și de linia 260 pe care putem să o ștergem. Vom obține rezultatul din figura 28, care ne întărește convingerea unei creșteri exponențiale a vitezei calculatoarelor.

fig.28

Reprezentarea grafică  
a creșterii perfor-  
manțelor calculatoare-  
lor electronice în timp





Să se traseze pe ecran un cerc prin puncte. De exemplu să se marcheze orele pe cadranul unui ceas.

Să presupunem că vrem să trasăm un punct P pe cercul de rază R și, de asemenea, că centrul cercului este în origine (vezi figura 29).

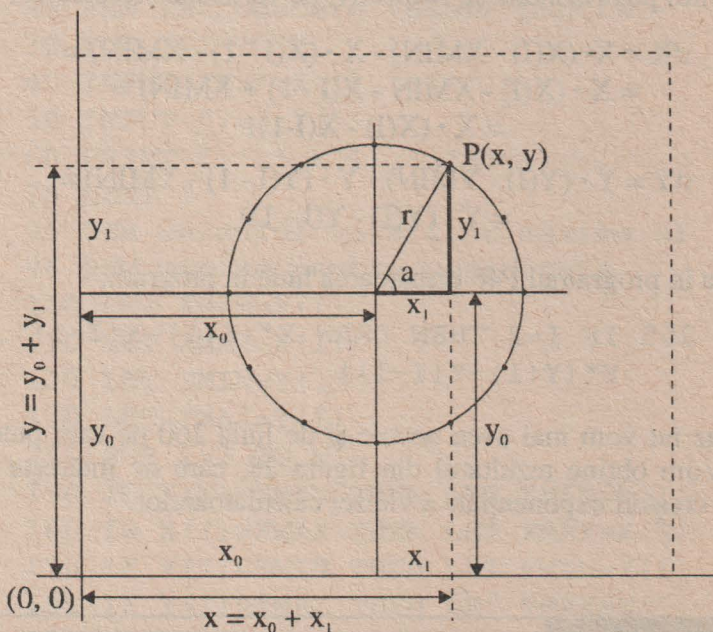


fig.29

Reprezentarea grafică a unui cerc

În acest caz distanța pe orizontală a punctului P, și anume X1, se calculează după formula  $X1 = R \cos A$ , unde A este unghiul la centru făcut de rază și orizontală (deoarece  $\cos A = X1/R$ ).

Similar, distanța pe verticală a punctului P, și anume Y1, se calculează după formula  $Y1 = R \sin A$ , deoarece  $\sin A = Y1/R$ . Astfel, în program, punctul P se va putea trasa prin:

PLOT R \* COS A , R \* SIN A

Dar în cazul real cercul nu este în origine. Originea se găsește pentru ecranul grafic de înaltă rezoluție în colțul din stînga jos al ecranului. Astfel, coordonata pe orizontală a punctului P, și anume X, va fi  $X = X_0 + X_1$ , unde  $X_0$  este distanța pe orizontală de la origine pînă la centrul cercului. Similar, coordonata pe verticală a punctului P va fi  $Y = Y_0 + Y_1$ . Acum vom putea trasa punctul P prin:

$$\text{PLOT } X_0 + R * \text{COS } A , Y_0 + R * \text{SIN } A$$

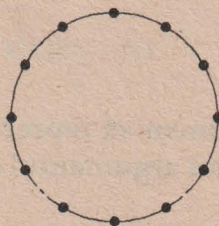
unde  $X_0$  și  $Y_0$  reprezintă coordonatele pe orizontală și pe verticală a centrului cercului.

Pentru a marca orele pe cadranul unui ceas va trebui să trasăm 12 puncte (vezi figura 30). Practic unghiul la centru va crește de la 0 la  $2\pi$  cu un pas

$$\text{de } \frac{2\pi}{12} = \frac{\pi}{6}.$$

fig.30

Cadran de ceas



Programul P47 este:

```

P47.A 5 REM ** Program de trasare a unui **
        6 REM **      cerc prin puncte      **
        7 REM **      pentru calculator HC    **
        10 INPUT "RAZA=" ;R
        20 INPUT "COORDONATELE CENTRULUI
              CERCULUI " ;X0 , Y0
        30 FOR A=0 TO 2*PI STEP PI/6
        40 PLOT X0 + R*COS A , Y0 + R*SIN A
        50 NEXT A
    
```

Pentru calculatoare PC programul similar este P47.B:

**P47.B**

```
5 REM ** Program de trasare a unui **
6 REM **      'cerc prin puncte      **
7 REM **      pentru calculator PC    **
8 CLS
9 SCREEN 1
10 LET PI=3.14
15 INPUT "RAZA=";R
20 INPUT "COORDONATELE CENTRULUI
    CERCLUI ";X0 , Y0
30 FOR A=0 TO 2*PI STEP PI/6
40 PSET (X0+R*COS(A), Y0+R*SIN(A)), 7
50 NEXT A
```



## Reprezentarea grafică a unei funcții de o variabilă

---



Ne propunem să reprezentăm grafic evoluția unei funcții atunci când argumentul său variază într-un interval.

În acest caz vom utiliza pentru reprezentare o funcție **DEF FN** urmînd ca, pentru fiecare caz în parte, să introducem în program forma funcției respective. Va trebui să introducem apoi valorile domeniului de definiție al funcției (XMIN și XMAX), numărul de puncte N prin care se reprezintă funcția și să rezervăm un spațiu de memorie Y(N) pentru toate valorile corespunzătoare ale funcției. Vom calcula și valoarea pasului pe orizontală XP prin formula:

$$XP = \frac{XMAX - XMIN}{N - 1}$$

deoarece pentru N puncte se obțin N - 1 intervale.

Apoi vom calcula valorile corespunzătoare pentru funcția Y mărirnd argumentul cu valoarea pasului. Vom calcula după metoda cunoscută valoarea minimă YMIN și valoarea maximă YMAX a funcției Y, apoi printr-un

ciclu de N pași vom trasa valorile funcției prin puncte, pentru valorile argumentului care cresc cu valoarea pasului.

Valorile funcției se vor calcula ținând cont de factorul de scară orizontal și vertical.

Programul P48 pentru trasarea graficului funcției sinus va fi:

```
P48.A 5 REM **      Program de trasare a      **
6 REM ** graficului functiei SINUS **
7 REM **      pentru calculatorul HC      **
10 DEF FN Y(X)=SIN X
20 INPUT "domeniul de definitie ";
      XMIN, XMAX
30 INPUT " N = "; N
40  Y(N)
50 LET X = XMIN
60 LET XP = (XMAX - XMIN)/(N - 1)
70 FOR I = 1 TO N
80 LET Y(I) = FN Y(X)
90 LET X = X + XP
100 NEXT I
110 LET YMIN = Y(1)
120 LET YMAX = Y(1)
130 FOR I = 2 TO N
140 IF Y(I) < YMIN THEN LET YMIN = Y(I)
150 IF Y(I) > YMAX THEN LET YMAX = Y(I)
160 NEXT I
170 CLS
180 LET X = XMIN
190 FOR I = 1 TO N
200 PLOT (X-XMIN)*255/(XMAX-XMIN),
      (Y(I)-YMIN)*175/(YMAX-YMIN)
210 LET X = X + XP
220 NEXT I
```

Pentru calculatoare PC programul similar va fi P48.B

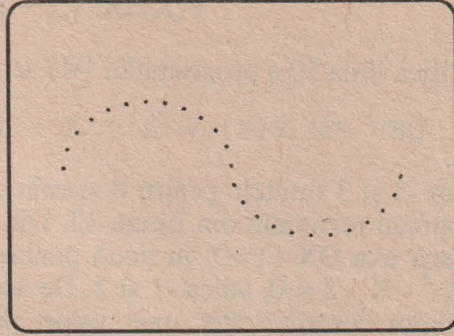
**P48.B**

```
5 REM **      Program de trasare a      **
6 REM ** graficului functiei SINUS **
7 REM **      pentru calculatorul PC   **
8 SCREEN 1 , 0
9 CLS
10 DEF FN Y(X)=SIN (X)
20 INPUT "domeniul de definitie ";
      XMIN, XMAX
30 INPUT " N = "; N
40 Y(N)
50 LET X = XMIN
60 LET XP = (XMAX - XMIN)/(N - 1)
70 FOR I = 1 TO N
80 LET Y(I) = FN Y(X)
90 LET X = X + XP
100 NEXT I
110 LET YMIN = Y(1)
120 LET YMAX = Y(1)
130 FOR I = 2 TO N
140 IF Y(I) < YMIN THEN LET YMIN = Y(I)
150 IF Y(I) > YMAX THEN LET YMAX = Y(I)
160 NEXT I
170 CLS
180 LET X = XMIN
190 FOR I = 1 TO N
200 PSET ((X-XMIN)*319/(XMAX-XMIN) ,
      200-(Y(I)-YMIN)*199/(YMAX-YMIN) ) , 7
210 LET X = X + XP
220 NEXT I
```

Introducînd pentru domeniul de definiție valorile 0 și 6,28 (adică  $2\pi$ ), pentru 100 de puncte obținem rezultatul din figura 31.

fig.31

Sinusoidă



Să se completeze programul astfel încât să se reprezinte și axele de coordonate.

Vom calcula  $X0$  și  $Y0$  astfel încât să "punem" punctele corespunzătoare dacă coordonatele lor "intră" în cele ale ecranului grafic. Vom folosi **OVER 1** pentru supratipărire, pentru HC. Adăugăm programului P48.A liniile:

```

230 LET X0=-XMIN*255/(XMAX-XMIN)
240 LET Y0=-YMIN*175/(YMAX-YMIN)
250 OVER 1
260 IF X0>=0 AND X0<=255 THEN
    PLOT X0,0 : DRAW 0,175
270 IF Y0>=0 AND Y0<=175 THEN
    PLOT 0,Y0 : DRAW 255,0
280 OVER 1

```

Pentru PC completarea similară este:

```

230 LET X0=-XMIN*320/(XMAX-XMIN)
240 LET Y0=-YMIN*199/(YMAX-YMIN)
260 IF X0>=0 AND X0<=319 THEN
    LINE (X0,0)-(X0,199)
270 IF Y0>=0 AND Y0<=199 THEN
    LINE (0,Y0)-(319,Y0)

```

Ca exemplu de aplicație să analizăm funcția

$$Y(X)=X^2 - X - 2$$

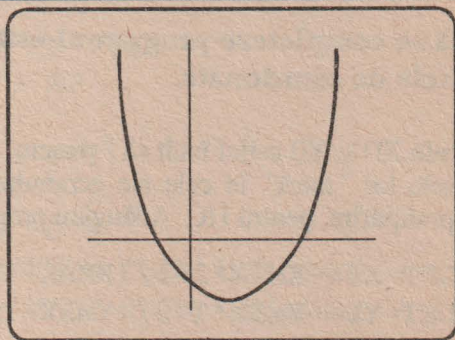
Vom modifica linia 10 a programului P48 astfel:

```
DEF FN Y(X) = X * X - X - 2
```

Introducem -2 și 3 limitele pentru domeniu de definiție iar pentru 100 de puncte obținem rezultatul din figura 32. Din figură se observă cum curba intersectează axa OX ( $Y=0$ ) în două puncte corespunzătoare rădăcinilor ecuației  $X^2 - X - 2 = 0$ , adică -1 și 2. De asemenea se observă că funcția are un minim corespunzător unei valori pozitive mai mici decât 1 (și anume valoarea 0,5) iar curba intersectează axa OY într-un punct corespunzător unui Y negativ (când  $X=0$  și  $Y=-2$ ).

fig.32

Reprezentarea grafică  
a unei funcții de  
gradul II date



## Limite

Calculatorul poate fi un instrument foarte eficient în calculul și studiul unor *limite de șiruri*. Afișarea rezultatelor expresiilor în cadrul unor repetiții ne conduce de obicei la determinarea corectă a limitei.



Să se calculeze cu ajutorul calculatorului:

$$\lim_{n \rightarrow \infty} \frac{n}{n^2 + 1}$$

Vom studia limita acestui șir prin intermediul următorului program (P49):



**P49.**    10 FOR I=1 TO 100  
           20 PRINT I , I/(I\*I+1)  
           30 NEXT I

Vom nota valorile obținute:

0,5	pentru I = 1
0,1	pentru I = 10
0,04	pentru I = 2
0,0099	pentru I = 100

Dacă încă nu este evident că limita este 0 vom putea continua experimentarea cu valori mai mari ale contorului. De exemplu cu I să ia valori de la 1000 la 1100.

Obținem 0,00099 pentru I = 1000 și 0,00090 pentru I = 1100.



Să se calculeze cu ajutorul calculatorului:

$$\lim_{n \rightarrow \infty} \frac{3n + 2}{2n + 2}$$

Vom folosi programul P49 pentru diverse valori ale contorului și cu linia 20 modificată astfel:

```
20 PRINT I , (3*I+2)/(2*I+1)
```

Vom obține :

1,6666	pentru I = 1
1,52	pentru I = 10
1,512	pentru I = 20
1,5049505	pentru I = 50
1,5024876	pentru I = 100
1,5002499	pentru I = 1000

Șirul va converge către valoarea 1,5.



Să se studieze cu ajutorul calculatorului:

$$\lim_{n \rightarrow 0} (1 + n)^{\frac{1}{n}}$$

Vom folosi programul P50:

```
P50. 10 FOR I=20 TO 1 STEP -1
      20 PRINT I , (1 + I)^(1/I)
      30 NEXT I
```

Vom obține :

1,1644235	pentru I = 20
1,2709816	pentru I = 10
1,5874011	pentru I = 3
2	pentru I = 1

Deoarece valorile nu sînt încă concludente vom studia limita în continuare folosind un pas mai mic. Linia 10 a programului P50 va deveni:

```
10 FOR I = 1 TO 0.1 STEP - 0.1
```

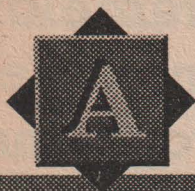
Vom obține :

2,0404513	pentru I = 0,9
2,25	pentru I = 0,5
2,5	pentru I = 0,1

Restrîngînd din nou intervalul pentru contor, vom lua acum pasul 0.01 ( $I = 0.100$  TO  $0.001$  STEP  $-0.01$ ) și obținem rezultatele:

2,6052534	pentru I = 0,09
2,65	pentru I = 0,05
2,7048138	pentru I = 0,01

Devine evident că limita este  $e$ .

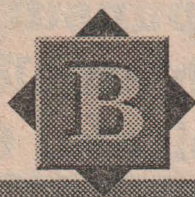


## Răspunsuri la problemele din capitolul 6

---

1. F  
2. A  
3. A  
4. F  
5. A  
6. F  
7. A  
8. F  
9. F  
10. A  
11. F  
12. A  
13. A  
14. A  
15. A  
16. A  
17. A  
18. F  
19. A  
20. F  
21. A  
22. A  
23. A

24. A  
25. F  
26. A  
27. A  
28. F  
29. A  
30. F  
31. F  
32. A  
33. F  
34. A  
35. A  
36. F  
37. A  
38. F  
39. A  
40. A  
41. A  
42. F  
43. F  
44. A  
45. A



# Lista programelor prezentate în carte. Mic caiet de programe

P1	Calculul multiplilor prin adunare . . . . .	32
P2	Calculul multiplilor prin înmulțire . . . . .	33
P3	Calculul multiplilor cu ciclul FOR-NEXT . . . . .	34
P4	Calculul CMMMC a două numere date . . . . .	35
P5	Calculul sumei primelor $N$ numere naturale . . . . .	38
P6	Calculul mediei aritmetice a $N$ numere . . . . .	40
P7	Calculul recompensei în vagoane de grâu a inventatorului șahului . . . . .	45
P8	Calculul produsului primelor $N$ numere naturale . . . . .	46
P9	Calculul unei expresii reprezentînd o sumă de inverse de produse de numere naturale . . . . .	47
P10	Verificarea parității unui număr . . . . .	49
P11	Determinarea dacă un număr este prim . . . . .	51
P12	Afișarea divizorilor unui număr dat . . . . .	52
P13	Afișarea numerelor prime pînă la un număr dat . . . . .	53
P14	Calculul CMMDC a două numere date . . . . .	56
P15	Descompunerea unui număr dat în factori primi . . . . .	58
P16	Calculul radicalului dintr-un număr dat prin metoda aproximărilor succesive . . . . .	60

<b>P17</b> Găsirea perechilor de numere a căror sumă este 1000, primul este divizibil cu 17 și al doilea cu 19. . . . .	62
<b>P18</b> Determinarea numărului $\overline{ABC}$ pentru care $A^2 + B^2 + C^2 = A+B+C$ . . . . .	63
<b>P19</b> Generarea numerelor de forma $\overline{3A2B}$ care se divid cu 9 . . . . .	64
<b>P20</b> Determinarea perechilor de cifre A și B pentru care numărul $\overline{TAB3}$ este divizibil cu 7 și cu 3 . . . . .	65
<b>P21A</b> Deplasarea unui punct pe ecran pentru calculatorul HC . . . . .	66
<b>P21B</b> Deplasarea unui punct pe ecran pentru calculator PC . . . . .	67
<b>P22A</b> Deplasarea unui caracter pe ecran pentru calculator HC . . . . .	68
<b>P22B</b> Deplasarea unui caracter pe ecran pentru calculator PC . . . . .	68
<b>P23</b> Șirul lui Fibonacci . . . . .	70
<b>P24</b> Șirul lui Fibonacci printr-o metodă recursivă . . . . .	73
<b>P25</b> Diferența, reuniunea și intersecția a două mulțimi . . . . .	76
<b>P26</b> Generarea de numere aleatoare întregi și pozitive . . . . .	106
<b>P27</b> Simularea jocului " Ghicește numărul" . . . . .	107
<b>P28</b> Antrenarea copiilor la învățarea tablei înmulțirii . . . . .	109
<b>P29</b> Generarea de numere aleatoare întregi cuprinse într-un interval dat . . . . .	110

<b>P30A</b> Desenarea de puncte aleatoare pe ecran pentru calculator HC . . . . .	110
<b>P30B</b> Desenarea de puncte aleatoare pe ecran pentru calculatorul PC . . . . .	111
<b>P31A</b> Desenarea de puncte aleatoare într-un dreptunghi dat pentru calculator HC . . . . .	111
<b>P31B</b> Desenarea de puncte aleatoare într-un dreptunghi dat pentru calculatoare PC . . . . .	111
<b>P32A</b> Afișarea în diverse locuri pe ecran a literei "a" (pentru HC) . . . . .	112
<b>P32B</b> Afișarea în diverse locuri pe ecran a literei "a" (pentru PC) . . . . .	112
<b>P33</b> Generarea de numere aleatoare întregi de $N$ cifre . . . . .	113
<b>P34</b> Generarea unui șir de $N$ numere aleatoare întregi și schimbarea maximului cu minimul . . . . .	115
<b>P35</b> Căutarea unui element dat într-o listă de numere și trecerea lui la sfârșitul listei . . . . .	117
<b>P36A</b> Afișarea numelor care încep cu o literă dată . . . . .	121
<b>P37A</b> Afișarea numelor care se termină cu o literă dată pentru calculator HC . . . . .	123
<b>P38A</b> Afișarea numelor care se termină cu o literă dată (pentru HC) folosind funcția LEN . . . . .	123
<b>P38B</b> Afișarea numelor care se termină cu o literă dată pentru calculatorul PC (folosind funcția LEN) . . . . .	124
<b>P39A</b> Afișarea numelor în ordine alfabetică pentru calculatorul HC . . . . .	125
<b>P40A</b> Trasarea unui segment prin puncte pentru calculator HC . . . . .	126

<b>P40B</b> <i>Trasarea unui segment prin puncte pentru calculator PC</i> . . . . .	126
<b>P41A</b> <i>Trasarea unui segment prin puncte la o înălțime dată pentru calculator HC</i> . . . . .	127
<b>P41B</b> <i>Trasarea unui segment prin puncte la o înălțime dată pentru calculator PC</i> . . . . .	127
<b>P42A</b> <i>Trasarea unui segment prin puncte pe verticală pentru calculator HC</i> . . . . .	128
<b>P42B</b> <i>Trasarea unui segment prin puncte pe verticală pentru calculator PC</i> . . . . .	128
<b>P43A</b> <i>Trasarea celui mai mic pătrat care se poate înscrie în ecranul grafic prin puncte (calculator HC)</i> . . . . .	130
<b>P43B</b> <i>Trasarea celui mai mare pătrat care se poate înscrie în ecranul grafic prin puncte pentru calculator PC</i> . . . . .	130
<b>P44A</b> <i>Trasarea celui mai mare pătrat care se poate înscrie în ecranul grafic prin puncte pentru calculator HC - variantă îmbunătățită</i> . . . . .	131
<b>P44B</b> <i>Trasarea celui mai mare pătrat care se poate înscrie în ecranul grafic prin puncte (calculator PC) - variantă îmbunătățită</i> . . . . .	131
<b>P45A</b> <i>Desenarea unui dreptunghi prin puncte pentru calculator HC</i> . . . . .	136
<b>P45B</b> <i>Desenarea unui dreptunghi prin puncte pentru calculator PC</i> . . . . .	136
<b>P46A</b> <i>Reprezentarea grafică a unei funcții prin puncte pentru calculator HC</i> . . . . .	140
<b>P46B</b> <i>Reprezentarea grafică a unei funcții pentru calculator PC</i> . . . . .	141
<b>P47A</b> <i>Trasarea unui cerc prin puncte pentru calculator HC</i> . . . . .	145

<b>P47B</b> <i>Trasarea unui cerc prin puncte pentru calculator PC</i> . . . . .	146
<b>P48A</b> <i>Trasarea graficului funcției SINUS pentru calculator HC</i> . . . . .	147
<b>P48B</b> <i>Trasarea graficului funcției SINUS pentru calculator PC</i> . . . . .	148
<b>P49</b> <i>Rezolvarea limitei șirului <math>\frac{n}{n+1}</math> când <math>n \rightarrow \infty</math></i> . . . . .	151
<b>P50</b> <i>Rezolvarea limitei șirului <math>(1+n)^{1/n}</math> când <math>n \rightarrow 0</math></i> . . . . .	152





### » Două programe pentru HC



#### CREION MAGIC

Cu ajutorul tastelor 5, 6, 7 și 8 puteți dirija o linie pe ecran și să desenați figurile pe care le doriți.

În timpul execuției, linia se alungește înspre ultima direcție indicată pînă cînd veți acționa altă tastă.

```
5 LET a$="": LET a=127: LET b=87
10 IF INKEY$="" THEN LET a$=INKEY$
20 LET b=b+(a$="7")-(a$="6")
30 LET a=a+(a$="8")-(a$="5")
40 IF a=256 OR a=0 THEN LET a=ABS (a-255)
50 IF b=176 OR b=0 THEN LET b=ABS (b-175)
60 PLOT a,b: GO TO 10
```



#### GRAFICĂ TRĂZNITĂ

Programul trasează linii de lungime întîmplătoare, de culori întîmplătoare, pornind din cele 4 colțuri de ecran.

```
10 PLOT 0,0: DRAW INK RND*6;RND*255,RND*175
20 PLOT 255,175: DRAW INK RND*6; -RND*255, -RND*175
30 PLOT 255,0: DRAW INK RND*6; -RND*255,RND*175
40 PLOT 0,175: DRAW INK RND*6;RND*255, -RND*175
50 GO TO 10
```

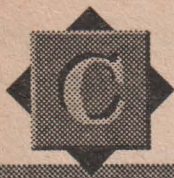
» *Cîteva programe cu sunete pentru PC*

```
10 REM SIRENA DE AMBULANTA
20 FOR I=400 TO 800 STEP 4
30 SOUND I,0.3
40 NEXT I
50 FOR I=800 TO 400 STEP -4
60 SOUND I,0.3
70 NEXT I
80 GO TO 20
```

```
10 REM SEMNAL
20 SOUND 5000 , 1
30 SOUND 10000 , 1
40 GO TO 20
```

```
10 REM SIRENA POLITIE
20 SOUND 523 , 5
30 SOUND 783 , 5
40 GO TO 20
```

```
10 REM LANSARE RACHETA
20 FOR I = 100 TO 3000 STEP 2
30 SOUND I , 100/I
40 NEXT I
```



# Lista figurilor cuprinse în carte

<b>Figura 1.</b> Calculatorul de sticlă . . . . .	14
<b>Tabelul 1.</b> Detalierea execuției programului pentru adunarea a $N$ numere naturale . . . . .	22
<b>Figura 2.</b> Diagramă tip arbore pentru realizarea operațiilor logice diferență, intersecție și reuniune a două mulțimi . . . . .	26
<b>Figura 3.</b> Schema logică pentru verificarea elementelor introduse și care aparțin unei mulțimi date . . . . .	29
<b>Figura 4.</b> Reprezentarea grafică a metodei de adunare a primelor 10 numere naturale . . . . .	36
<b>Figura 5.</b> Adunarea numerelor naturale . . . . .	37
<b>Figura 6.</b> Reprezentarea grafică a metodei de calcul a recompensei inventatorului jocului de șah . . . . .	45
<b>Figura 7.</b> Reprezentare grafică pe axa numerelor pentru funcția $INT$ (ÎNTREG) . . . . .	48
<b>Figura 8.</b> Calculatorul cu părțile lui componente . . . . .	79
<b>Figura 9.</b> Octetul . . . . .	81
<b>Figura 10.</b> Circuit ROM . . . . .	85
<b>Figura 11.</b> Mouse . . . . .	87
<b>Figura 12.</b> Monitor . . . . .	87
<b>Figura 13.</b> Casetă magnetică . . . . .	90
<b>Figura 14.</b> Unitate de disc flexibil . . . . .	90

<b>Figura 15a.</b> Organizarea unei dischete . . . . .	91
<b>Figura 15b.</b> Protejarea unei dischete la scriere . . . . .	91
<b>Figura 16.</b> Bandă magnetică . . . . .	92
<b>Figura 17a.</b> Imprimantă cu impact . . . . .	93
<b>Figura 17b.</b> Imprimantă cu laser . . . . .	93
<b>Figura 18.</b> Calculatoarele în competiție . . . . .	94
<b>Figura 19.</b> Generarea de numere aleatoare întregi într-un interval dat . . . . .	109
<b>Figura 20.</b> Reprezentarea grafică cu indicarea ordinii logice a operațiilor pentru găsirea unui element dat într-o listă de elemente și trecerea sa la sfârșitul listei . . . . .	117
<b>Figura 21.</b> Reprezentarea unui spațiu de memorie prin utilizarea unei variabile alfanumerice bidimensionale pentru memorarea unor nume . . . . .	120
<b>Figura 22.</b> Coordonatele unui punct de pe diagonala pătratului . . . . .	132
<b>Figura 23.</b> Reprezentarea grafică a diagonalei pătratului . . . . .	133
<b>Figura 24.</b> Coordonatele unui punct de pe diagonala dreptunghiului și stabilirea funcției care reprezintă această diagonală . . . . .	134
<b>Figura 25.</b> Stabilirea funcției care reprezintă diagonala opusă a dreptunghiului . . . . .	135
<b>Figura 26.</b> Stabilirea funcției care reprezintă diagonala unui dreptunghi de coordonate și poziție date . . . . .	138
<b>Figura 27.</b> Reprezentarea grafică prin puncte a creșterii performanțelor calculatoarelor electronice în timp . . . . .	141

<b>Figura 28.</b> <i>Reprezentarea grafică a creșterii performanțelor calculatoarelor electronice în timp</i> . . . . .	143
<b>Figura 29.</b> <i>Reprezentarea grafică a unui cerc</i> . . . . .	144
<b>Figura 30.</b> <i>Cadran de ceas</i> . . . . .	145
<b>Figura 31.</b> <i>Sinusoidă</i> . . . . .	149
<b>Figura 32.</b> <i>Reprezentarea grafică a unei funcții de gradul II date</i> . . . . .	150



## Glosarul cuvintelor cheie

---

Prin **cuvinte cheie** se înțeleg cuvintele existente în memoria calculatorului și care prezintă o anumită semnificație pentru acesta. Lucrarea de față face referire la cuvintele cheie ale limbajului **BASIC** utilizate pentru doua categorii de calculatoare: cele **compatibile ZX Spectrum** (HC, CIP, JET, COBRA, TIM, Sinclair) care utilizează versiunea Sinclair a limbajului **BASIC** și cele **compatibile IBM PC** care utilizează versiunea **GW BASIC**.

Dacă pentru parcurgerea și experimentarea lucrării în cazul calculatoarelor compatibile ZX Spectrum este necesară doar pornirea calculatorului, **BASIC**-ul fiind rezident în memoria acestor calculatoare, în cazul calculatoarelor compatibile IBM PC este necesară în prealabil încărcarea fișierului **GW BASIC**.

Atragem atenția asupra modului diferit de introducere a cuvintelor cheie pentru cele două tipuri de calculatoare: *pentru cele compatibile ZX Spectrum orice cuvânt cheie se introduce prin acționarea unei anumite taste sau a unei combinații de taste, iar pentru cele compatibile IBM PC prin tastarea cuvântului sau combinației de cuvinte, caracter cu caracter (acesta este, de fapt, și modul standard de lucru).*

Orice cuvânt cheie poate face parte din următoarele categorii:

- ✗ **comandă** - cuvânt cheie al cărui introducere are ca urmare execuția imediată de către calculator a unei acțiuni. Exemplu: RUN, LIST, LOAD, SAVE, etc.
- ✗ **instrucțiune** - cuvânt cheie care are ca urmare o acțiune executată de calculator și care poate fi utilizat într-o linie de program. Acțiunea este îndeplinită numai atunci când programul va fi executat. Exemplu: INPUT, PRINT, etc.

Un cuvânt cheie care reprezintă o comandă va funcționa ca o instrucțiune atunci când este folosit într-o linie de program.

De exemplu: 1000 LIST, cuvîntul cheie LIST reprezintă o instrucțiune, acțiunea de listare a programului realizîndu-se cînd execuția programului va ajunge la linia 1000.

Invers, o instrucțiune poate fi considerată comandă cînd va fi folosită ca atare. De exemplu, PRINT 3+4, cuvîntul cheie PRINT reprezintă o comandă, deoarece acțiunea de afișare a rezultatelor se realizează imediat după introducere.

În indexul cuvintelor cheie se va indica cu prioritate categoria cea mai reprezentativă în care se poate încadra cuvîntul cheie respectiv. De exemplu, RUN se va trece în categoria comandă/instrucțiune, deoarece, deși se poate utiliza și ca instrucțiune, se utilizează în primul rînd ca și comandă. Invers, INPUT se va trece la categoria instrucțiune/comandă, deoarece, deși se poate folosi și în mod comandă, se utilizează în primul rînd în linii de program, deci ca instrucțiune.

✗ **funcție** - cuvînt cheie a cărui folosire produce o valoare. Acest cuvînt se utilizează în combinație cu o comandă sau instrucțiune.

Exemple: INT, RND, ABS, SIN, SQR. Exemple de utilizare:  
PRINT INT 8.1.

Atragem atenția asupra unei deosebiri esențiale în utilizarea cuvintelor cheie funcție pentru cele două tipuri de versiuni BASIC aflate în discuție: în cazul versiunii GW BASIC este obligatorie folosirea parantezelor pentru încadrarea argumentului unei funcții, în timp ce, în cazul versiunii Sinclair BASIC folosirea parantezelor este opțională. Nefolosirea parantezelor în acest caz, deși simplifică utilizarea, poate conduce la alte rezultate decît cele așteptate.

De exemplu cu PRINT INT 8.1+2.2 se va obține rezultatul 10.2, deoarece se va evalua INT 8.1 obținîndu-se rezultatul parțial 8 iar acesta se va aduna cu 2.2 obținîndu-se rezultatul final 10.2. Dacă se dorește întregul din rezultatul lui 8.1+2.2 atunci se va introduce PRINT INT (8.1+2.2).

✗ **operator logic** - cuvînt cheie care este utilizat într-o comandă sau instrucțiune și care determină dacă o condiție este adevărată sau falsă.

Exemple de operatori logici sînt: AND, OR, NOT.

**X alte cuvinte cheie** - cuvinte cheie care sînt utilizate, în cadrul unor comenzi sau instrucțiuni și care reprezintă particule de propoziții care fac o legătură între cuvinte. Exemple: THEN, AT, STEP etc.

În scopul cunoașterii și folosirii cuvintelor cheie, adică a realizării de programe, vor trebui cunoscute :

- localizarea cuvintelor cheie pe tastatură (pe ce tastă se găsește și în ce mod se obține), fapt important pentru calculatoarele compatibile Spectrum, pentru cele compatibile IBM PC introducerea cuvintelor cheie făcîndu-se, așa cum am văzut, prin introducerea caracter cu caracter;
- ce fel de cuvînt cheie este (categoria);
- scopul utilizării cuvîntului cheie (în ce caz se folosește);
- formatul (sintaxa) cuvîntului cheie.

Deoarece în lucrare sînt oferite numeroase exemple de folosire a cuvintelor cheie în program, deseori fiind indicată și modalitatea de obținere s-a considerat util să se facă doar o trecere în revistă (memento) în ordine alfabetică a tuturor cuvintelor cheie folosite, indicîndu-se doar semnificația (din care rezultă și scopul principal de folosire), categoria în care se încadrează cuvîntul, pentru ce tip de calculator se folosește (dacă nu se indică, se subînțelege că este utilizabil pentru amîndouă tipurile), precum și paginile din lucrare în care puteți regăsi cuvîntul cheie respectiv într-un anumit program. Indexul nu conține toate cuvintele cheie din limbajul BASIC, ci doar majoritatea lor. Pentru restul de cuvinte, precum și indicații referitoare la formatul complet (sintaxa) în care pot fi folosite, mai ales că acesta poate fi diferit pentru cele două tipuri de calculatoare avute în vedere (și de asemenea poate conduce la rezultate neidentice), se pot consulta manuale de referință privitoare la limbajele Sinclair BASIC și GW BASIC sau lucrări citate în bibliografie.

**ABS.** *Funcție.* Produce valoarea absolută a unei valori numerice, adică valoarea fără semn pozitiv sau negativ.

**AND.** *Operator logic.* Leagă două sau mai multe condiții în cadrul unei instrucțiuni. Dacă toate condițiile legate prin AND sînt adevărate, atunci combinația va fi adevărată.



**AT.** *Particulă de legătură.* Se folosește în instrucțiunile PRINT, LPRINT și INPUT pentru a localiza o afișare pe ecran sau o imprimare pe hârtie. Se utilizează în Sinclair BASIC.

**BEEP.** *Instrucțiune/comandă.* Produce o notă muzicală de o anumită durată și înălțime în Sinclair BASIC. În GW BASIC produce un sunet.

**BIN.** *Funcție.* Transformă un număr binar într-unul zecimal. Se utilizează în Sinclair BASIC.

**CLS.** *Instrucțiune/comandă.* Șterge de pe ecran orice text sau grafică, lăsând întregul ecran cu culoarea fondului.

**CONT (INUE).** *Comandă.* Reia execuția unui program după o comandă BREAK. În Sinclair BASIC forma este CONTINUE.

**COS.** *Funcție.* Introduce cosinusul unui unghi.

**DATA.** *Instrucțiune.* Specifică o listă de date. Se folosește în program în conjuncție cu instrucțiunea READ.

**DEF FN.** *Funcție.* Definește o funcție.

**DIM.** *Instrucțiune.* Rezervă un spațiu de memorie.

**DRAW.** *Instrucțiune/comandă.* Desenează pe ecran linii drepte sau curbe în Sinclair BASIC. Desenează un obiect definit printr-o secvență de subcomenzi grafice în GW BASIC.

**FOR.** *Instrucțiune/comandă.* Execută o serie de instrucțiuni de un număr specificat de ori (ciclu).

**GO SUB.** *Instrucțiune/comandă.* Realizează saltul la subrutina care începe la linia specificată.

**GOTO.** *Instrucțiune/comandă.* Execută salt necondiționat la instrucțiunea de la linia specificată.

**IF.** *Instrucțiune/comandă.* Determină o execuție condiționată de valoarea logică a unei expresii.

**INKEY\$.** *Funcție.* Detectează ultimul caracter introdus de la tastatură. (citește tastatura).

**INPUT.** *Instrucțiune/comandă.* Se introduc datele de la tastatură în timpul execuției programului.

**INT.** *Funcție.* Returnează cel mai mare întreg care este mai mic sau egal cu valoarea care reprezintă argumentul.

**LEN.** *Funcție.* Returnează lungimea șirului de caractere specificat.

**LET.** *Instrucțiune/comandă.* Atribuie unei variabile valoarea expresiei specificate.

**LINE.** *Instrucțiune/comandă.* Desenează pe ecran o linie sau un dreptunghi. În GW BASIC. În Sinclair BASIC se utilizează împreună cu comanda SAVE pentru salvarea unui program cu lansare automată de la numărul de linii indicat.

**LIST.** *Comandă/instrucțiune.* Afișează lista instrucțiunilor programului din memoria calculatorului.

**LOAD.** *Comandă/instrucțiune.* Încarcă un program de pe un suport extern în memoria calculatorului.

**LOCATE.** *Instrucțiune/comandă.* Poziționează cursorul în poziția specificată. Se folosește în GW BASIC.

**MID\$.** *Funcție/instrucțiune.* Ca funcție returnează un subșir al șirului indicat, iar ca instrucțiune, înlocuiește o parte a unui șir cu un alt șir. Se folosește în GW BASIC.

**NEW.** *Comandă/instrucțiune.* Șterge din memorie programul curent și toate variabilele.

**OR.** *Operator logic.* Leagă două sau mai multe condiții în cadrul unei instrucțiuni. Dacă cel puțin o condiție este adevărată, atunci combinația va fi adevărată.

**OVER.** *Instrucțiune/comandă.* Afișează un caracter pe deasupra altuia prin suprapunere.

**PAUSE.** *Instrucțiune/comandă.* Suspendă execuția unui program pe o perioadă de timp. Se folosește în Sinclair BASIC.

**PEEK.** *Funcție.* Returnează conținutul octetului aflat la adresa de memorie specificată.

**PI.** *Funcție.* Introduce valoarea lui PI ( $\pi = 3,141592$ ). Se folosește în Sinclair BASIC.

**PLOT.** *Instrucțiune/comandă.* Desenează un punct într-o anumită poziție pe ecran. Se folosește în Sinclair BASIC.

**POKE.** *Instrucțiune/comandă.* Introduce valoarea unui octet în locația de memorie specificată.

**PRINT.** *Instrucțiune/comandă.* Afișează pe ecran valoarea expresiilor indicate.

**PSET.** *Instrucțiune/comandă.* Desenează un punct pe ecran la coordonatele specificate. Se folosește în GW BASIC.

**PRESET.** *Instrucțiune/comandă.* Desenează/șterge un punct pe ecran la coordonatele specificate. Se folosește în GW BASIC.

**READ.** *Instrucțiune/comandă.* Citește valorile din instrucțiunea DATA și le atribuie variabilelor din lista de variabile.

**REM.** *Instrucțiune.* Lansează comentarii într-un program.

**RETURN.** *Instrucțiune/comandă.* Termină execuția unei subrutine și revine în programul apelant.

**RND.** *Funcție.* Generează un număr aleator mai mare sau egal cu 0 și mai mic ca 1.

**RUN.** *Comandă/instrucțiune.* Începe execuția unui program.

**SAVE.** *Comandă/instrucțiune.* Memorează (salvează) fișierele (programele sau datele) pe un suport extern.

**SCREEN.** *Instrucțiune/comandă.* Servește la declararea modului de utilizare a ecranului (text, grafică, culori). Se folosește în GW BASIC. Tot în GW BASIC se mai folosește la returnarea codului ASCII corespunzător caracterului din rândul și coloana specificate. Forma SCREEN\$ se utilizează cu LOAD sau SAVE în Sinclair BASIC pentru a încărca/salva o imagine ecran.

**SIN.** *Funcție.* Returnează sinusul unghiului specificat.

**SQR.** *Funcție.* Returnează valoarea radicalului expresiei indicate.

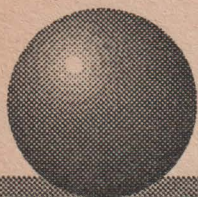
**STEP.** *Particulă.* Se utilizează în cadrul unei linii de program FOR pentru a evidenția pasul cu care se modifică valoarea variabilei contor.

**STOP.** *Instrucțiune/comandă.* Termină execuția unui program într-un anumit punct.

**THEN.** *Particulă.* Se utilizează într-o instrucțiune IF. Dacă este adevărată condiția atunci instrucțiunea (instrucțiunile) care urmează după THEN se vor executa, iar dacă nu este adevărată se vor executa instrucțiunile începând cu linia următoare de program.

**TO.** *Particulă.* Se utilizează într-o instrucțiune FOR pentru a determina valorile indicelui din ciclul FOR-NEXT.

**TRON.** *Comandă.* Determină listarea numerelor liniilor executate. Se folosește în GW BASIC.



# Indexul cuvintelor cheie

## A

ABS 66  
AND 11 - 12, 65, 149  
AT 68, 112

## B

BEEP 112, 129  
BREAK 23

## C

CLS 67, 109, 111,  
127 - 128, 131, 136,  
141, 146 - 147  
COS 94, 144 - 145

## D

DATA 119  
DEF FN 146 - 148, 150  
DIM 70, 76, 115, 117, 121,  
123 - 125, 140  
DRAW 143, 149

## F

FOR 16, 34, 38, 41, 45 - 47,  
51 - 53, 58, 62 - 63,  
65, 70, 76, 97, 115,  
117 - 118, 121, 123,  
125, 128, 136

## G

GO SUB 58  
GOTO 32, 49, 53, 56, 60,  
66 - 67, 69, 73, 76,  
106 - 107, 110 - 111,  
117, 121, 123, 125

## I

INKEY\$ 65, 68  
INPUT 16, 32 - 33, 38, 41,  
46 - 47, 51 - 52, 56,  
58, 70, 113, 115, 124,  
145  
INT 48 - 49, 51, 53, 56, 59,  
62, 64, 78, 97, 106,

109, 111 - 113, 115,  
117

## L

LEN 124  
LINE 149  
LIST 15

## O

OR 13, 66, 68, 78  
OVER 149

## P

PAUSE 69  
PEEK 84 - 85  
PI 145  
PLOT 66, 110, 126,  
128 - 130, 133 - 134,  
138, 140, 144 - 145

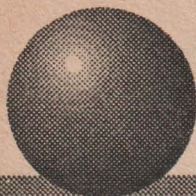
POKE 84  
PRESET 67  
PSET 111, 127 - 128, 132,  
137, 142, 148

## R

READ 121  
RETURN 59  
RND 106 - 107, 110 - 112,  
115

## S

SCREEN 67 - 68, 111 - 112,  
127 - 128, 132, 142,  
146, 148  
SIN 94, 144, 147  
SQR 10, 62  
STOP 23, 52, 56, 59, 97



## Bibliografie

**Sinclair ZX Spectrum User's Guide**, Dorling Kindersley Ltd. and Sinclair Research Ltd.

**Informatica pentru elevi**, Microinformatica S.R.L., Cluj, 1991

**Calculatoare personale, de la hardware la software**, Electronica NIS S.R.L., Cluj 1991

**Gazeta de informatică nr. 1 - 3**, Editura Petrion, București, 1991; **nr. 4 - 10**, Editura Libris, Cluj, 1992

**Culegere de probleme pentru concursuri**, *Bălan D.M., Bălan G.*, Editura Licurici, Suceava, 1992

**Limbaj BASIC**, Editura Licurici, Suceava, 1992

**Are You Computer Literate?**, *Billings K., Moursound D.*, Dilithium Press, Portland, Oregon, 1979

**Partenerul meu de joc, calculatorul**, *Diamandi I.*, RECOOP, București, 1989, 1990

**Cum să realizăm jocuri pe calculator**, *Diamandi I.*, Editura AGNI, București, 1992

**La conduite du ZX Spectrum**, *Hartnell T., Jones D.*, Edition Eyrolles, 1993

**L'enfant aux commandes de l'ordinateur**, *Krieger D.*, Edition Eyrolles, 1984

**ABC microcalculatoare**, *Petrescu A. și colectiv*, Editura Tehnică, București, 1990

**La conduite de l'IBM-PC**, *Plonin A.*, Editions Eyrolles, 1984

**Programarea structurată în BASIC**, *Stan I., Șerban M., Sandor O., Singer H.*, Editura Libris, Cluj, 1992

**Hello, BASIC**, *State Lumnița*, Editura AGNI, București, 1992

- Microsoft GW BASIC**, *Vasiu L., Grama R.*, Editura Libris, Cluj, 1992
- Programe de instruire – BASIC**, *Vlad A. I.*, Editura Științifică și Enciclopedică, București, 1989
- Să învățăm să programăm în BASIC**, *Zaharia C., Zaharia M.*, Editura Tehnică, București, 1992



NUMELE \_\_\_\_\_

PRENUMELE \_\_\_\_\_

LOCALITATEA \_\_\_\_\_

STR \_\_\_\_\_

NR \_\_\_\_ BL \_\_\_\_ ET \_\_\_\_ AP \_\_\_\_

JUDETUL \_\_\_\_\_ COD \_\_\_\_\_

Vă rog să-mi expediați prin colet poștal cu ramburs \_\_\_\_\_

\_\_\_\_\_exemplare din cartea (cărțile) \_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

DATA \_\_\_\_\_ SEMNATURA \_\_\_\_\_



În curînd vor apărea:



**Mihaela Cârstea , Ion Diamandi**  
*Un PC pentru fiecare*



**Adrian Atanasiu**  
*Cum se scrie un algoritm? simplu*



**Marian Gheorghe**  
*Cine ești tu, BASIC ?*



**Răzvan Andonie, Ilie Gârbacea**  
*Algoritmi fundamentali în C++*



## **Despre carte :**

Cartea este un îndrumar pentru utilizarea instructivă a calculatoarelor de tip SPECTRUM (HC, JET, COBRA, CIP) atât la școală, cât și acasă. Conține rezolvarea a circa 50 de probleme de matematică, în limbajul BASIC (pentru calculatoarele de tip SPECTRUM) și în versiunea GW-BASIC (pentru calculatoarele compatibile IBM-PC).

*Instrument extrem de util pentru orele de informatică.*

## **Despre autor :**

Dl. **Ion Diamandi** este șeful compartimentului de Instruire Asistată de Calculator de la SOFTWARE-ITC-SA.

**Cărți publicate:** *Dialog cu viitorul* (1988); *Partenerul meu de joc, calculatorul* (1988, 1989, 1990); *LOGO - o nouă metodă de a învăța cu ajutorul calculatorului* (1991); *Din spectacolul informaticii - calculatorul personal* (1992); *Cum să realizăm jocuri pe calculator* (1992); *40 jocuri logice în BASIC* (1992).