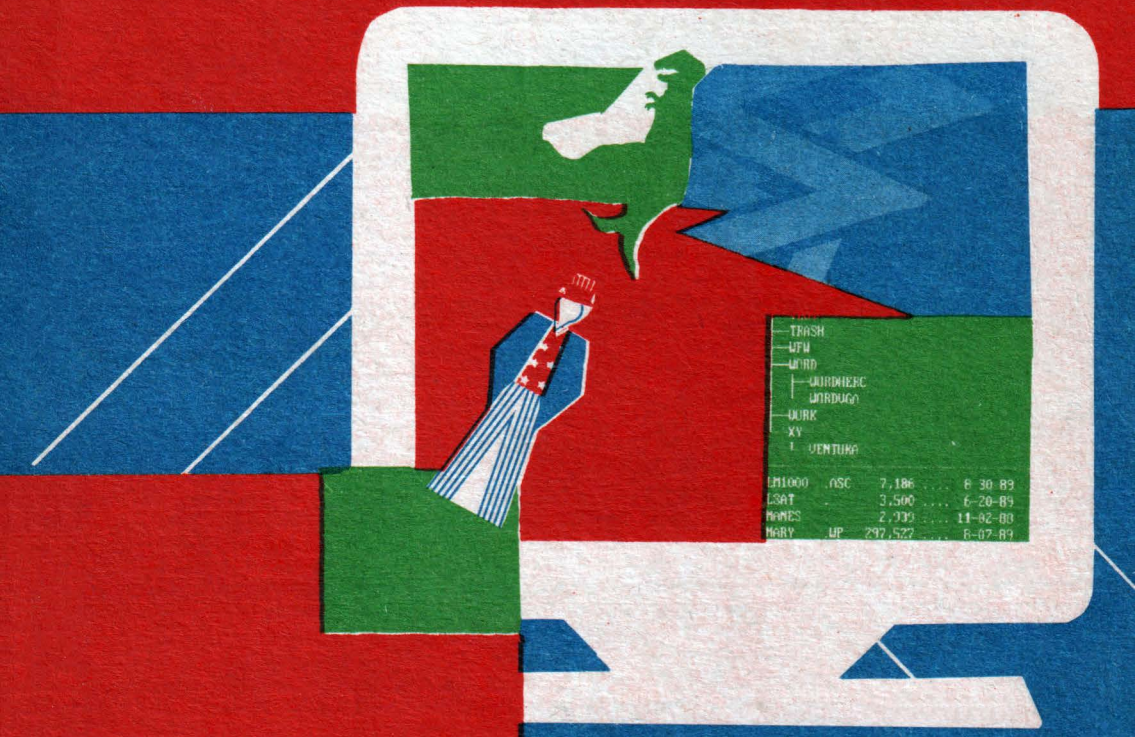
Biblioteca de Informatică

Ion Diamandi

Cum să realizăm JOCURI pe CALCULATOR



Editura Agni

"Lumea jocurilor este imaginea în mic a societății."

(Th. Arnold)

Redactor: **Mihaela IONESCU**
Grafica: **Emil BOJIN**
Coperta: **Iolanda MALAMEN**
Machetare şi tehnoredactare: **Dan ROMANESCU**

ISBN: 973-96347-4-5

Editura AGNI
CP: 30-107, BUCUREŞTI
tel: 615.55.59 fax: 210.93.36

ION DIAMANDI

CUM SĂ REALIZĂM JOCURI PE CALCULATOR

Ediția a II-a

Editura Agni

București 1994

CUVÎNT ÎNAINTE

Impactul calculatoarelor, în general, şi al celor personale, în particular, se face din ce în ce mai simţit în tot mai multe activităţi umane. Mai recent, acest impact s-a înregistrat şi în cadrul şcolii şi al familiei, pe linie educaţională şi de instruire, unde calculatoarele au început să influenţeze decisiv modul de abordare a problemelor specifice. În acest context, a fost lansat un program de lungă durată care prevede atît comercializarea ca bunuri de larg consum a calculatoarelor personale, privite ca instrumente inteligente de joc şi de instruire, cît şi a produselor-program destinate utilizării calculatoarelor: casete magnetice cu jocuri logice, jocuri de aventuri şi de îndemînare, programe care asigură asistenţă din partea calculatorului în însuşirea cunoştinţelor din diferite discipline şcolare (matematică, fizică, chimie, limbi străine etc) şi în dezvoltarea de aptitudini şi deprinderi (conducerea autovehiculelor, circulaţia rutieră, dactilografie etc). De asemenea, sînt avute în vedere produse program livrabile pe suporturi magnetice (casete şi discuri), care reprezintă programe utilitare pentru calculatoare pentru diverse domenii cu care se pot realiza: baze de date, agende personale, desene şi schiţe tehnice, partituri muzicale, prelucrări şi editări de texte, calcule matematice complexe etc.

În cadrul acestei problematici, de o importanţă vitală este sensibilizarea şi obişnuirea consumatorilor cu produsele amintite şi, în general, cu utilizarea calculatoarelor. Potenţialii consumatori şi principalii beneficiari sînt, în primul rînd copiii, ei fiind şi principalii utilizatori de mîine ai acestor instrumente.

Lucrarea de faţă se înscrie pe această linie, propunîndu-se învăţarea folosirii calculatoarelor prin realizarea de jocuri pentru calculatoare.

Este, deci, un **ghid al utilizării calculatoarelor personale**, reprezentînd practic volumul al doilea al lucrării: **"Partenerul meu de joc - calculatorul"**, editată de Jeco Trading S.A. (RECOOP) în 1988 şi reeditată în 1989.

Deşi stă la îndemîna oricărei persoane care doreşte să înveţe utilizarea calculatoarelor, el se adresează în special elevilor de gimnaziu şi de liceu (în primul rînd celor din clasele VI- VIII), multe din problemele rezolvate în lucrare regăsindu-se în programa disciplinelor şcolare pentru aceste clase.

Cîteva precizări pentru profesori

Pentru învățarea utilizării calculatoarelor s-a ales ca mijloc realizarea de jocuri pentru calculator deoarece:

— copiilor le place atît să practice jocuri, cît și să conceapă și să realizeze singuri jocuri după ideile proprii. Astfel, practicînd jocuri pentru calculator și, în special, încercînd să realizeze jocuri pentru calculator, vor învăța, de fapt, să utilizeze calculatoarele, să le programeze. Așa cum ceea ce contează pentru copilul mic, care a făcut un turn din cuburi, nu este atît turnul, cît edificarea lui, tot așa copiii mai mari vor învăța programarea calculatorului prin realizarea de jocuri pentru calculator, procesul de învățare fiind, în acest caz, neimpus;

— jocurile reprezintă un instrument ideal de experimentare a unor idei și a unor instrucțiuni învățate, precum și un mijloc ideal de a utiliza anumite concepte avansate, care, expuse numai teoretic și nelegate de practică, ar putea crea copiilor dificultăți de înțelegere. Este vorba, în primul rînd, de utilizarea **algoritmilor** în rezolvarea problemelor (orice joc prezentînd spre rezolvare o multitudine de probleme) și de realizarea de **modele**. De asemenea, realizarea de jocuri pentru calculator implică practic învățarea utilizării calculatoarelor sub mai multe aspecte: realizarea de desene (grafică), folosirea culorilor, animația, compunerea unor melodii etc. cu ajutorul calculatorului. Toate aceste aspecte conduc la obținerea unor jocuri mai atractive, mai frumoase, mai antrenante și, în același timp, la deprinderea utilizării mai avansate a calculatoarelor de către copii;

— sub formă de joc, se pot prezenta programe care realizează baze de date, prelucrări și editări de texte, editări grafice etc.

Alegerea metodei de învățare a programării calculatoarelor pe baza jocurilor are la bază lucrul practic cu calculatorul, preferîndu-se introducerea directă de subiecte, însoțită eventual de explicarea unor noțiuni teoretice. Exemplele sînt urmate de explicații, care se referă la:

- utilizarea unor mici programe-jocuri care realizează lucruri semnificative și interesante sau care constituie exemple ce evidențiază o anumită structură sau instrucțiune;
- modificarea jocurilor, care astfel vor realiza lucruri noi;
- pregătirea și proiectarea unui nou joc, care va rezolva o nouă problemă.

În această fază, lucrarea și-a propus utilizarea unui număr cît mai mare de instrucțiuni și de comenzi, practic a întregului set de instrucțiuni, comenzi și funcții **BASIC**, punîndu-se accent pe înțelegerea mecanismului de rezolvare a problemelor cu calculatorul. Deoarece jocurile pentru calculator implică rezolvarea unui set de probleme, la începutul lucrării s-a prezentat modul de abordare și de rezolvare a

acestor probleme, folosind descrierea algoritmului în *pseudocod* și *schemele logice*. Fiind astfel puse bazele utilizării calculatoarelor, precum și ale folosirii lor în scopul rezolvării problemelor, se poate trece la proiectarea de jocuri de natură cît mai diversă. Astfel, calculatorul devine pe rînd partener inteligent de joc, profesor de matematică, de istorie și de limbi străine, poet, pictor și compozitor. În sfîrșit, pe măsura deprinderii tehnicilor de programare mai avansate, a îmbogățirii vocabularului cu cuvinte cheie, se trece la o utilizare mai complexă a calculatoarelor. Este momentul în care elevii vor folosi calculatorul ca instrument în realizarea unor modele, a unor experimente, a organizării propriilor informații, în vederea utilizării ulterioare a unor jocuri complexe care vor îngloba multe din tehnicile de programare de acum cunoscute.

"Clasa", adică locul unde are loc deprinderea utilizării calculatoarelor, poate fi acasă, la școală sau oriunde există calculatoare și elevii pot lucra împreună și se pot consulta.

Elevii vor avea un caiet de informatică pe care își vor lua notițe, vor copia programele realizate împreună cu notele explicative privind programele, vor rezolva exercițiile propuse și vor experimenta ideile proprii.

De asemenea, se va păstra cîte o copie (pe casetă magnetică) a jocurilor realizate și, eventual, a rezultatelor (listingul). Pot fi explorate idei (jocuri) noi, care vor fi și ele păstrate pe casetă magnetică.

Cum se utilizează GHIDUL

Pentru a putea utiliza acest GHID, este necesar să se cunoască:

— modul de operare (de introducere a cuvintelor cheie) a calculatorului cu care se lucrează;

— un minim de cuvinte cheie (instrucțiuni, comenzi și funcții);

— cîteva elemente absolut necesare pentru abordarea de mici programe și cîteva cunoștințe privind modul de păstrare (salvare) și de încărcare a acestora.

Toate acestea pot fi foarte lesne cunoscute și asimilate prin parcurgerea lucrării: **"Partenerul meu de joc — calculatorul".**

În plus, pentru a se putea utiliza GHIDUL, mai sînt necesare cunoștințe și noțiuni obișnuite de matematică din clasele de gimnaziu și, în cîteva cazuri, din clasele de liceu.

GHIDUL conține mai multe exemple care se pot constitui în activități, astfel încît orele de informatică să se poată desfășura săptămînal, pe parcursul unui an școlar, sub forma unui cerc de informatică (sau matematică aplicată) și jocuri. Este

recomandabil ca în decursul unei săptămîni, să aibă loc o oră condusă de un profesor și o alta în decursul căreia elevii să-și rezolve temele date și să-și experimenteze propriile idei.

S-a considerat necesară înserarea la început, a unui capitol care să pună bazele teoretice ale proiectării de aplicații cu calculatoare și a modului în care calculatoarele pot fi programate pentru a rezolva probleme. Se insistă asupra abordării problemelor de sus în jos, a programării structurate, a folosirii schemelor logice și a algoritmilor. Apoi, se trece efectiv la proiectarea de jocuri și de aplicații ale calculatoarelor în diverse domenii.

Capitole speciale sînt rezervate tehnicilor referitoare la grafica și muzica cu calculatorul, fără ajutorul acestora neputîndu-se proiecta jocuri mai complicate. Se insistă, în special, asupra utilizării caracterelor grafice și semigrafice, precum și a tehnicilor pentru realizarea animației cu calculatorul. În final, sînt date exemple de proiectare și de realizare a unor jocuri mai complexe care înglobează, fiecare, majoritatea tehnicilor prezentate.

La sfîrșitul lucrării sînt date rezolvările la întrebările și exercițiile propuse, care pe parcursul lucrării sînt însemnate, în ordine, cu cifre. Răspunsurile din caietele elevilor pot fi deseori diferite, dar, acest lucru nu înseamnă, neapărat, că sînt greșite. Adevăratul test al programului este trecut atunci cînd el realizează ceea ce a dorit autorul.

Jocurile sînt concepute într-un **BASIC** standard, astfel încît, în afara calculatoarelor pentru care a fost proiectat GHIDUL **(Sinclair ZX SPECTRUM, CIP, JET, HC, TIM-S, COBRA)** să poată fi executate, cu mici modificări, și pe alte calculatoare. Cei ce doresc să efectueze aceste modificări, trebuie să acorde o atenție sporită modulelor și structurilor din programe care sînt specifice calculatoarelor compatibile **Sinclair ZX SPECTRUM** și anume: lucrul cu variabile de tip șir de caractere, grafică, etc.

Programele sînt prezentate, pentru o înțelegere mai bună, atît cu linii de comentariu (cea de titlu avînd un format special pentru evidențierea acestuia), cît și cu explicații suplimentare efectuate chiar pe listing.

Atenție! O eroare poate ascunde alta. Un mesaj de eroare semnalat de calculator în linia 250 a unui program, de exemplu, nu înseamnă neapărat că eroarea se găsește în această linie.

Este posibil ca linia 120 să fie aceea care necesită a fi corectată sau să existe o incompatibilitate între datele introduse și program.

Cîteva precizări privind modul de ţinere a evidenţei jocurilor şi a informaţiilor referitoare la ele.

O atenţie deosebită se va acorda păstrării evidenţei programelor (jocurilor) salvate pe casete magnetice. Vom denumi o colecţie de programe, memorate pe un suport de memorie externă (casetă sau disc magnetic), **Bibliotecă de programe**. Altă denumire pentru colecţie poate fi **Librărie**.

Exemplu de bibliotecă:

BIBLIOTECA DE JOCURI

Lista numelor programelor din bibliotecă, împreună cu alte cîteva informaţii importante referitoare la ele, o vom numi **Catalog**. Alt nume pentru catalog poate fi **Director** *(Directory)*. De obicei, se realizează cataloage pentru programele de pe un suport magnetic. Toate programele salvate pe casetă magnetică vor trebui notate în caietul de informatică într-un loc special rezervat sau într-un caiet separat, de tip repertoar sau agendă. Acest lucru ar putea părea lipsit de importanţă la începutul activităţii, cînd aveţi realizate doar cîteva programe, dar cînd veţi acumula un număr mare de programe veţi aprecia faptul că aţi lucrat sistematic. Pentru fiecare program tastat, scris sau salvat pe caseta magnetică, va trebui să realizaţi următoarele operaţii:

1. să-i daţi un nume;

2. să-l salvaţi pe caseta magnetică;

3. să-l listaţi la imprimantă, dacă aveţi posibilitatea, şi dacă nu, să-l copiaţi de pe ecran pe caiet;

4. să-l documentaţi (prin notele explicative despre program);

5. să-l treceţi în **catalogul** bibliotecii voastre de programe.

Prin documentarea programului se înţelege completarea unui set de informaţii referitoare la program. Informaţiile vor fi scrise în caiete şi vor trebui să cuprindă:

1. o descriere a ceea ce realizează programul (scopul);

2. cum se realizează acest lucru. Pentru jocurile a căror rezolvare în program necesită algoritmi speciali sau pentru programele care descriu fenomene sau lecţii, trebuie făcută o scurtă prezentare a suportului teoretic (formule folosite, reguli, legi etc.);

3. un listing al programului;

4. o schemă logică a programului (dacă este de dimensiuni mai mari) și lista de variabile folosite, cu o precizare a ceea ce reprezintă;

5. cum să se utilizeze, eventual cu un exemplu de folosire;

6. data la care a fost scris.

Modul de realizare a schemelor logice se va introduce în primul capitol al lucrării.

Iată un exemplu privind o înregistrare (un joc) din secțiunea **Catalog** a caietului:

Nume program:	PICTOR
Casetă:	JOCURI 3
Localizare:	15—18
Lungime program:	28 linii (se poate indica și în octeți)
Data salvării:	16.09.1989
Scop:	realizarea de desene pe ecran

Este indicat ca pe fiecare casetă să se aplice o etichetă (pe fiecare față) pe care să se treacă următoarele informații:

1) Numele casetei (sau codul).

2) Data și numele posesorului.

3) Lista cu denumirea programelor care se găsesc pe casetă.

Observați că, spre deosebire de **Cataog**, în care informațiile referitoare la programe sînt mai bogate (se indică inclusiv locul unde poate fi găsit programul pe casetă), pe casetă se vor trece numai informațiile strict necesare.

I. Fundamente ale realizării jocurilor pe calculator

În lucrarea **„Partenerul meu de joc — calculatorul"** ați învățat cum să introduceți un joc în memoria calculatorului sau, cu alte cuvinte, **modul de operare** a calculatorului. De asemenea, ați învățat cum să realizați jocuri de dimensiuni reduse, în limbaj **BASIC.**

Lucrarea de față își propune să vă învețe cum să realizați jocuri pentru calculator, și, pentru că un joc necesită rezolvarea unei serii de probleme, trebuie în primul rînd, studiat **cum rezolvă calculatorul problemele.** În acest scop, va trebui:

1) să se găsească o metodă pentru rezolvarea problemei;

2) să se realizeze un program.

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

Pînă acum, v-ați și întîlnit, de fapt, cu o serie de algoritmi, cu care ați rezolvat unele probleme de matematică, ca de exemplu: algoritmul aflării celui mai mare divizor comun a două numere naturale, algoritmul găsirii unui număr prim și altele.

Un algoritm este reprezentat, deci, 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.

Dacă această definiție vi se pare complicată, puteți să vă gîndiți că un algoritm este foarte asemănător cu o rețetă dintr-o carte de bucate.

Desigur, poate nu este ușor de aplicat o rețetă pentru a obține cele mai delicioase preparate (sau cele mai bune rezultate) dar, oricum, este mai ușor decît de găsit însăși rețeta. O metodă prin care se pot găsi algoritmi (sau rețete) este **programarea structurată** sau **metoda de abordare de sus în jos a problemelor:** problema este descompusă în mai multe subprobleme și, pe măsură ce aceste subprobleme vor fi mai mici, pașii vor semăna din ce în ce mai mult cu operații 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. Ca ajutor se utilizează scheme, cum sînt de exemplu: **diagramele de structură**. Cea mai simplă dintre diagramele de structură este cea de **tip arbore**.

După realizarea unei astfel de diagrame, se caută să se obțină o descriere a algoritmului într-un limbaj succint, astfel încît regulile (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 să traducem (translatăm) fiecare secțiune a pseudocodului în echivalentul său din limbajul **BASIC**. Uneori, însă, chiar prin intermediul diagramei de structură, fluxul controlului în program poate fi dificil de urmărit. În acest caz, se utilizează o altă tehnică cu ajutorul căreia se determină ordinea în care sînt prelucrate modulele programului și instrucțiunile în cadrul unui modul. Această tehnică utilizează **schemele logice**.

Deci, este bine de reținut că atît algoritmii, cît și reprezentările lor în pseudocod și scheme logice, nu depind nici de calculatorul pe care se dorește să se implementeze programul, nici de limbajul de programare în care se va realiza programul.

REZOLVAREA PROBLEMEI

Producerea algoritmului sau 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. Sarcina de a produce algoritmul va fi mai ușoară dacă se utilizează o metodă de proiectare structurată asociată cu o reprezentare grafică a algoritmului, deci, cu o diagramă de structură sau schemă logică.

Pentru a produce algoritmul de rezolvare a unei probleme vor trebui parcurse următoarele etape:

1. Privire generală asupra problemei

Pentru a rezolva problema, va trebui să știm **ce fel** de problemă este și **ce** trebuie făcut. Mai tîrziu, va trebui să ne gîndim **cum** se va realiza obiectivul propus.

O descriere completă a problemei trebuie să includă:

1.1. ce rezultat urmărim să obținem;

1.2. ce informații (date) ne trebuie, pentru a fi introduse;

1.3. ce operații vom efectua cu datele.

EXEMPLE:

Problemă: să se scrie un program cu ajutorul căruia să se găsească cel mai mare divizor comun al două numere.

Problemă: să se realizeze cu ajutorul calculatorului o agendă telefonică care să conțină maximum 100 de nume (cunoștințe) ale căror caracteristici să poată fi modificate la nevoie.

Pentru prima problemă, introducerea datelor, operațiile și extragerea rezultatelor sînt ușor de realizat: se introduc cele două numere și se scriu multiplii lor pe două coloane: primul număr, care se va găsi în ambele liste, va reprezenta cel mai mare divizor al numerelor (vezi "Partenerul meu de joc — calculatorul").

A doua problemă este, însă, mai complexă și necesită mai multă analiză și mai multe informații.

2. Analiza problemei

Aceasta implică:

2.1. analiza problemei pentru identificarea modului în care poate fi rezolvată de calculator;

2.2. identificarea tuturor formulelor și relațiilor care vor fi folosite;

2.3. identificarea tuturor datelor implicate.

În acest stadiu este important, de fiecare dată, să se scrie:

- ce formule, expresii și funcții vor fi folosite;
- ce fel de date sînt implicate (de tip numeric, șiruri de caractere, etc.);
- care sînt informațiile care vor fi introduse și care este forma în care vor fi introduse; la fel pentru rezultate;
- cîte date vor fi implicate;
- ce operații se vor face și de cîte ori.

Nu uitați! *Toate informațiile obținute din analiza problemei vor trebui scrise; pe baza acestora se va putea trece la proiectarea algoritmului*

3. Proiectarea algoritmului utilizînd o metodă structurată

În acest scop se va proceda la:

3.1. descompunerea problemei în subprobleme;

3.2. utilizarea unei diagrame de structură (de exemplu de tip arbore), ca ajutor;

3.3. identificarea modulelor sau a părților de module ca fiind de:
- introducere
- prelucrare
- extragere

3.4. utilizarea structurilor de control;

3.5. realizarea unui tabel al datelor (variabilelor, constantelor);

3.6. rafinarea algoritmului pînă cînd transpunerea sa în limbaj **BASIC** devine un fapt evident.

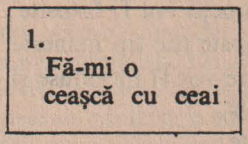
Programarea structurată înseamnă proiectarea algoritmului după modul de abordare de sus în jos prin descompunerea problemei în subprobleme (sau module), fiecare urmînd a fi, apoi, rezolvată separat.

4. Descrierea algoritmului în pseudocod şi realizarea unei scheme logice

EXEMPLU DE REALIZARE A UNEI DIAGRAME DE TIP ARBORE

Diagrama de tip arbore va permite descompunerea problemei în subprobleme şi module, căutîndu-se ca acestea să devină atît de simple, încît să poată fi codificate direct în instrucţiuni BASIC.

Să presupunem că avem un robot înzestrat cu braţe, picioare şi ochi, pe care dorim să-l programăm astfel încît să facă o ceaşcă cu ceai. Deci, sarcina principală pentru robot este:



Deocamdată, robotul nu poate să îndeplinească această sarcină, deoarece el recunoaşte numai obiectele (de exemplu: ceaşcă, ceainic, priză, bucătărie etc.) şi poate îndeplini numai comenzi simple (de exemplu: "mergi la bucătărie", "pune ceainicul pe plită" etc.). Pentru a îndeplini sarcina dată (care este mai complexă), aceasta va trebui descompusă în subsarcini (comenzi) pe care le vom pune în ordine (vezi fig. 1):

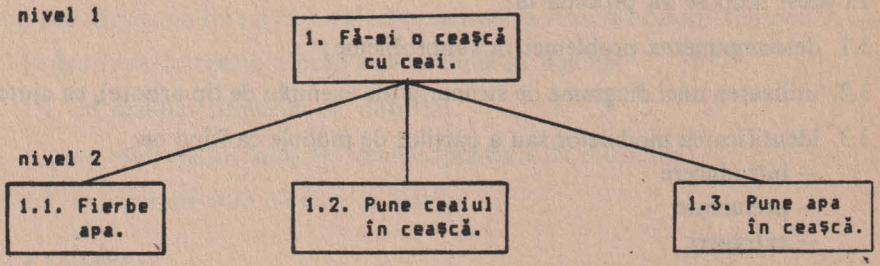


Fig. 1

Fiecare din aceste subsarcini este încă destul de greu de îndeplinit de către robot, astfel încît fiecare trebuie descompusă la rîndul ei (vezi fig. 2).

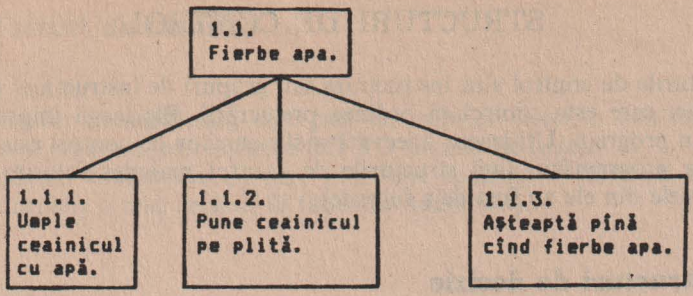


Fig. 2

Robotului trebuie să i se mai spună și cum să umple ceainicul cu apă, astfel încît și această sarcină va trebui descompusă (vezi fig. 3).

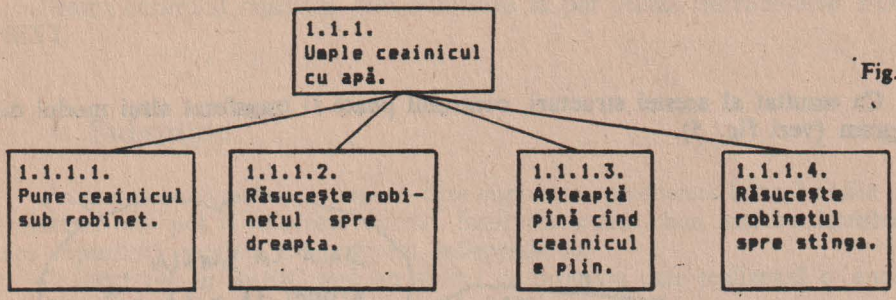


Fig. 3

Majoritatea programelor conțin activități de introducere, prelucrare de date și extragere de rezultate. Proiectînd programul și formînd module, va deveni, în curînd, evident, pentru fiecare modul, ce funcție (sau ce funcții) va îndeplini în introducere, prelucrare sau extragere.

Iată cum arată întreaga diagramă de tip arbore pentru nivelul 3 (vezi fig. 4):

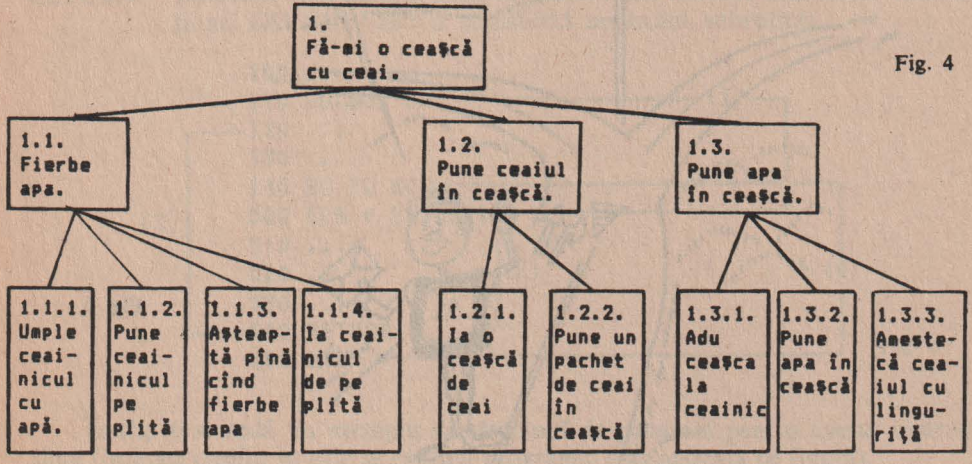


Fig. 4

STRUCTURI DE CONTROL

Structurile de control sînt instrucțiuni sau grupuri de instrucțiuni din program (module) cu care este controlată ordinea prelucrării. Ele leagă împreună diferite module din program. Utilizarea adecvată a structurilor de control este o parte importantă a programării. Iată structurile de control mai des folosite în limbajul **BASIC** (unele din ele au fost deja folosite):

1. Structuri de decizie

Calculatoarele iau decizii comparînd valoarea unei variabile cu altă valoare. De exemplu:
IF A = O THEN (face ceva)

Ca rezultat al acestei structuri, controlul poate fi transferat altui modul din program (vezi fig. 5).

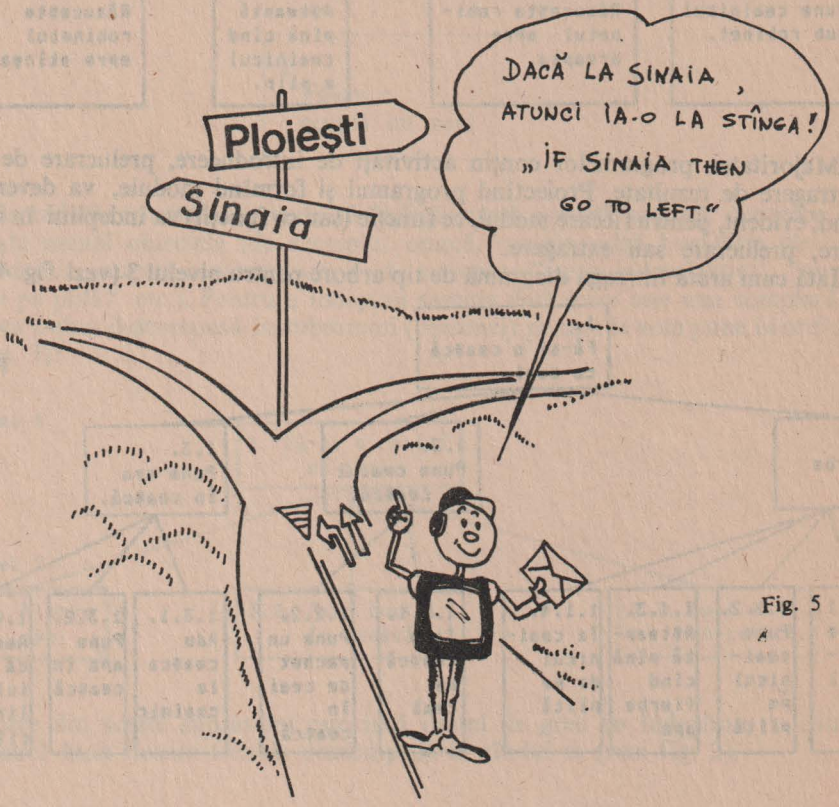


Fig. 5

2. Structuri de transfer

Acestea pot fi:

a) **de transfer necondiționat**, care este un transfer direct al controlului către un alt segment, utilizînd instrucțiunea **GO TO** (număr de linie).

b) **de transfer condiționat**, cu care transferul controlului către un alt segment este făcut ca urmare a unei decizii: **IF** (condiție adevărată).

3. Cicluri

Un ciclu este o secvență de pași repetați într-un program. Repetiția este controlată prin numărare sau prin testarea unei condiții.

Pentru controlul repetiției prin numărare se pot utiliza instrucțiunile **FOR—NEXT.**

4. Subrutine

Programarea structurată implică descompunerea problemei în mai multe subprobleme, care pot fi rezolvate separat. Subrutinele sînt chiar aceste subprobleme, care reprezintă module de program independente.

O subrutină în **BASIC** este un modul de program care realizează o anumită sarcină și este apelat cu o instrucțiune **GO SUB.**

Secțiunea programului (subrutina) este parcursă pînă la instrucțiunea **RETURN,** care va transfera controlul înapoi la linia care urmează celei ce conține instrucțiunea **GO SUB.**

Deci, pentru a crea o subrutină sînt utilizate două instrucțiuni:

GO SUB (număr linie) — transferă controlul la numărul de linie specificat;

RETURN — părăsește subrutina și redă controlul liniei imediat următoare instrucțiunii **GO SUB** care a transferat controlul subrutinei.

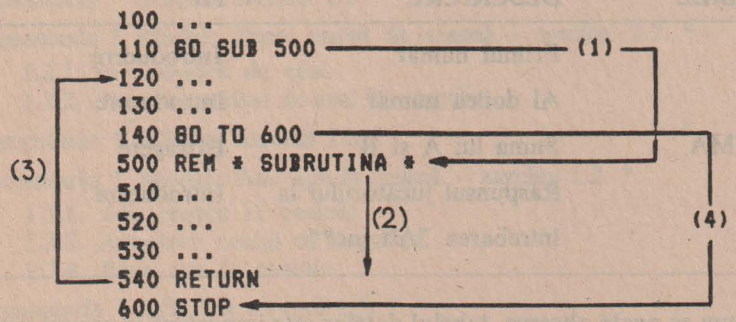


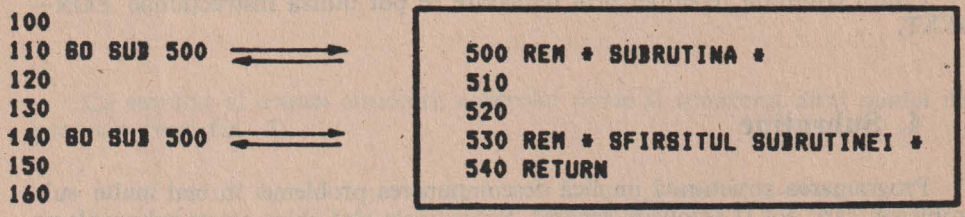
Fig. 6

În fig. 6 se redă un exemplu de structură de program pentru aceste instrucțiuni (ordinea pașilor în care se execută programul este marcată pe figură).

Subrutinele sînt deseori utilizate pentru repetarea unor segmente de instrucţiuni în cadrul aceluiaşi program.

De exemplu, să presupunem că într-un joc calculatorul pune întrebări şi jucătorul răspunde. Fiecare răspuns bun va fi marcat de cîteva secvenţe sonore, de felicitare şi fiecare răspuns greşit de alte secvenţe sonore, de avertizare. În acest caz, este indicat să se realizeze cîte un grup de instrucţiuni de sine stătător (subrutină) pentru fiecare melodie şi să se apeleze cînd este cazul (răspuns bun sau greşit), nemairepetîndu-se în cadrul programului secvenţele de instrucţiuni. Mecanismul este descris în fig. 7.

Program principal **Subrutina** Fig. 7



De remarcat că se poate face un apel al unei subrutine din cadrul altei subrutine şi, de asemenea, este permisă apelarea subrutinei chiar din cadrul ei (subrutine recursive).

TABELUL DATELOR

Iată un exemplu de realizare a unui tabel de date pentru un algoritm:

VARIABILE	DESCRIERE	TIP
A	Primul număr	Introducere
B	Al doilea număr	Introducere
SUMA	Suma lui A şi B	Extragere
A$	Răspunsul jucătorului la întrebarea "Alt joc?"	Introducere

După cum se poate observa, tabelul datelor este important în scopul documentării. Pentru proiectarea unui joc, pot fi trecute în tabelul datelor valorile pe care le pot lua variabilele în anumite puncte ale programului.

Descrierea algoritmului

Pentru descrierea algoritmului se va proceda astfel:

— se scrie metoda de rezolvare în paşi a problemei (algoritmul) în propoziţii simple;

— se realizează schema logică, arătîndu-se funcţionarea programului de la început la sfîrşit;

— se testează dacă algoritmul funcţionează, înainte de codificarea lui în **BASIC.**

În diagrama de tip arbore, fiecărui bloc sau model îi era asociată descrierea unei sarcini. Algoritmul va fi descris printr-o metodă pas cu pas şi va include toate descrierile din blocuri. Primul nivel al descrierii va fi titlul programului sau algoritmului. Următorul nivel va fi reprezentat de titlurile secţiunilor din program. Fiecare din aceste secţiuni va cuprinde un grup de module. Ultimul nivel al diagramei de tip arbore va fi reprezentat de instrucţiunile specifice pe care calculatorul le poate rezolva.

Utilizînd o diagramă de tip arbore, se poate scrie, de exemplu, algoritmul prin care comandăm robotului o ceaşcă de ceai ca o secvenţă de comezi pe care el le înţelege, sau algoritmul pentru realizarea pe calculator a unei agende telefonice ca o secvenţă de instrucţiuni care vor fi codificate în limbaj BASIC. Pentru pseudocod, vom folosi scurte propoziţii în limba română şi fiecare modul se va identifica prin comentarii.

Iată o descriere în pseudocod (după diagrama de tip arbore din fig. 4) pentru ca robotul să facă o ceaşcă de ceai:

Comentariu ** Algoritm pentru ca robotul să facă o ceaşcă de ceai **

Comentariu * Fierberea apei — sarcina 1.1. *

1.1.1. Pune ceainicul pe plită.
1.1.2. Umple ceainicul cu apă.
1.1.3. Aşteaptă pînă cînd fierbe apa.

Comentariu * Sfîrşitul sarcinii 1.1. *

Comentariu * Modul: Pune ceaiul în ceaşcă — sarcina 1.2. *

1.2.1. Ia o ceaşcă de ceai.
1.2.2. Pune un pachet de ceai în ceaşcă.

Comentariu * Sfîrşitul sarcinii 1.2. *

Comentariu * Modul: Pune apa în ceaşcă — sarcina 1.3. *

1.3.1. Adu ceaşca la ceainic.
1.3.3. Amestecă ceaiul cu linguriţa.
1.3.4. Pune apa în ceainic.

Comentariu * Sfîrşitul sarcinii 1.3. *

Comentariu ** Sfîrşitul algoritmului — ceaşca de ceai **

EXERCIȚII ȘI PROBLEME

1. *Descrierea în pseudocod a algoritmului pentru ca robotul să facă o ceașcă de ceai conține cîteva greșeli:*

a) *unele comenzi sînt altfel formulate față de cele din diagramă și robotul nu va fi capabil să le recunoască;*

b) unele comezi sînt înscrise într-o ordine greșită;

c) unele comezi lipsesc din descrierea algoritmului.

Corectați descrierea algoritmului, astfel încît robotul să poată să facă o ceașcă de ceai.

2. *Extindeți diagrama de tip arbore și algoritmul cu subsarcini pe nivel* 4.

3. *Realizați o diagramă de tip arbore și scrieți algoritmul în pseudocod pentru ca robotul să realizeze conectarea calculatorului la rețea.*

Răspunsurile la paginile 135—136

SCHEME LOGICE

Schema logică reprezintă o altă metodă utilizată în realizarea de programe. O schemă logică constă din blocuri logice legate între ele. Blocurile sînt reprezentate prin diverse forme geometrice, fiecare avînd o anumită semnificație și, la fel ca la diagrama de tip arbore, fiecare conține o descriere succintă a ceea ce programul trebuie să realizeze în acel punct.

Motivul utilizării schemelor logice este acela că ele fac vizibil și descriu fluxul controlului în algoritm și, deci, în program, prin evidențierea celor mai importante structuri de control, ușurînd astfel codificarea algoritmului în instrucțiuni **BASIC**. De asemenea, ele se constituie ca o parte importantă a documentației programului, putînd fi astfel de mare ajutor atunci cînd se va proceda, de exemplu, la modificarea programului sau la înțelegerea lui de către alte persoane.

Blocurile folosite în realizarea schemelor logice au (prin convenție) anumite semnificații care trebuie respectate, altfel schema logică nu va putea fi înțeleasă de către alte persoane. Fluxul logic din program va fi ilustrat printr-o serie de blocuri logice (sub formă de dreptunghiuri, romburi, paralelograme, etc.), legate între ele prin săgeți.

BLOCURILE UTILIZATE PENTRU SCHEME LOGICE

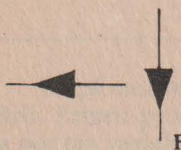


Fig. 8

liniile de legătură (de flux) sînt reprezentate prin **săgeți**. Ele conectează blocurile între ele. Săgețile arată direcția fluxului (fig. 8).

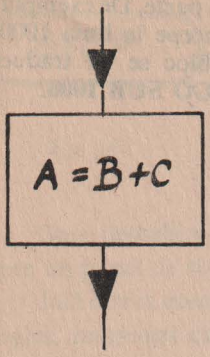


Fig. 9

blocul (funcţia) de prelucrare este reprezentat printr-un dreptunghi în interiorul căruia se indică o instrucţiune (sau sarcină) programabilă (fig. 9). De exemplu: "Cumpără un pachet de ceai" sau A = B + C. Ultima se va traduce în program prin instrucţiunea **LET A = B + C.**

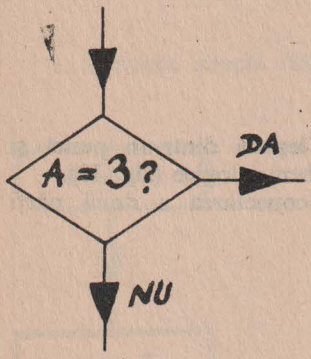


Fig. 10

blocul de decizie cu test de condiţie este reprezentat printr-un romb în interiorul căruia se va indica o întrebare, ca: A = 3? sau "Mai este ceai în cămară?" (fig. 10).
În programe acest lucru se va traduce cu o instrucţiune de tip **IF A = 3 THEN . . .**
Testul are două ramuri de ieşire, corespunzătoare pentru DA şi pentru NU, după cum condiţia este adevărată sau falsă, ceea ce determină fluxul în program.

Fig. 11

blocul de introducere date sau **de extragere** (prin afişarea rezultatelor pe ecran sau listarea la imprimantă) este reprezentat printr-un paralelogram în interiorul căruia se specifică tipul (introducere sau extragere) şi datele sau rezultatele (fig. 11). De exemplu: CITEŞTE B (vezi figura) sau AFISEAZA A. Acestea se vor traduce în program prin instrucţiuni, ca: **READ B** şi **PRINT A.**

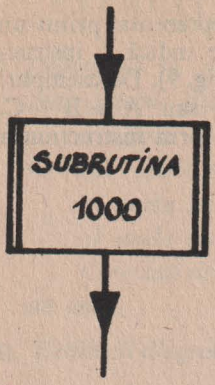


Fig. 12

blocul unui proces specificat în altă parte. De exemplu: pentru apelarea subrutinei care începe la linia 1000: Subrutina 1000 (fig. 12). Acest bloc se va traduce în program printr-o instrucțiune **GO SUB 1000**.

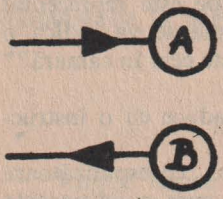


Fig. 13

simboluri folosite pentru **ieșirea** dintr-un punct și intrarea în alt punct al schemei logice (fig. 13). Aceste simboluri permit conectarea a două părți din schemă.

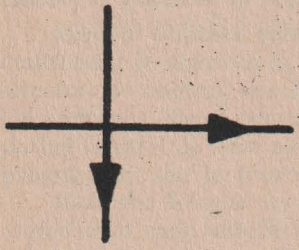
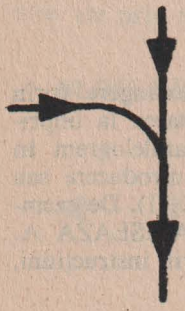


Fig. 14

simboluri folosite pentru **traversarea liniilor de legătură**. Se observă că liniile de legătură nu se conectează (fig. 14).



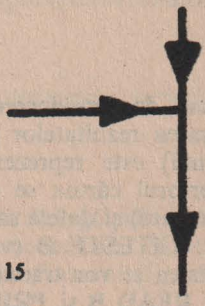


Fig. 15

simboluri folosite pentru **joncțiunea liniilor de legătură**. Se observă că cele două linii se conectează (fig. 15).

punct terminal al schemei logice, de exemplu: de început a unui program (**START**) sau de sfîrşit (**STOP**) (fig. 16).

Fig. 16

Spre deosebire de diagrama de tip arbore, schema logică converge totdeauna spre un punct de stop şi este într-o legătură directă cu programul pe care îl descrie.

Iată cîteva exemple (care reprezintă în acelaşi timp şi tipuri principale) de scheme logice, împreună cu programele pe care le reprezintă:

1. **Secvenţă simplă (fig. 17).**


```
Program

10 LET x = 5
20 INPUT y
30 PRINT x,y
```

Fig. 17

2. Decizie (fig. 18).

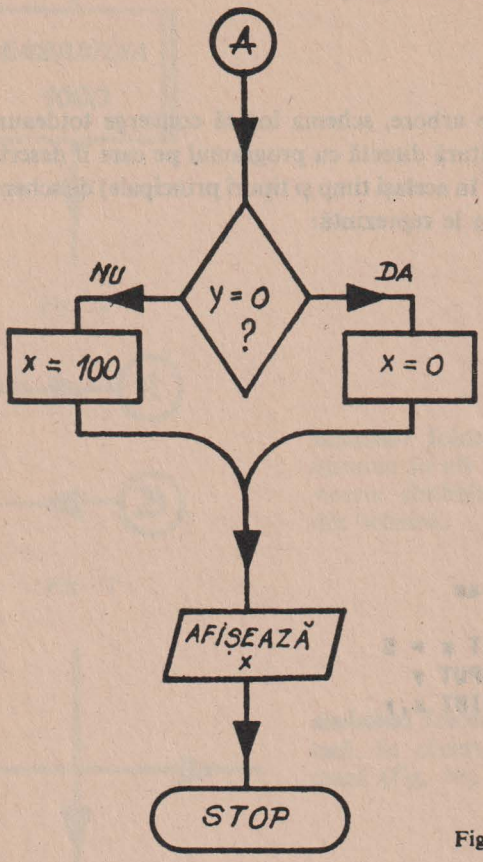


Fig. 18

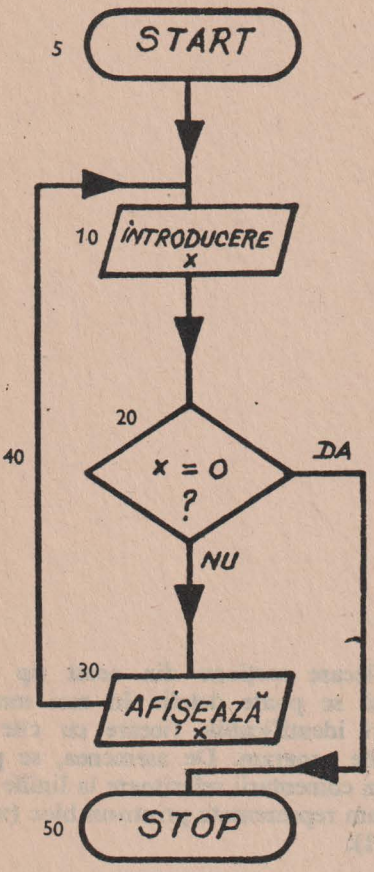
```
Program

40 IF y = 0 THEN GO TO 70
50 LET x = 100
60 GO TO 80
70 LET x = 0
80 PRINT x
90 STOP
```

Observați că în ultima schemă logică s-a omis simbolul pentru linia 60. Acest lucru s-a întîmplat deoarece instrucțiunea respectivă de program (GO TO) este indicată de către linia de legătură. Același lucru se întîmplă și cu GO TO din instrucțiunea de condiție din linia 40.

Uneori, pentru ușurarea înțelegerii legăturii dintre schema logică și program, se pot trece lîngă blocurile și săgețile schemei logice numerele de linie corespunzătoare din program, așa cum s-a realizat în următorul exemplu:

3. Cicluri (fig. 19).




```
Program

 5 REM CICLU
10 INPUT x
20 IF x=0 THEN GO TO 50
30 PRINT x
40 GO TO 10
50 REM SFIRSIT
```

Fig. 19

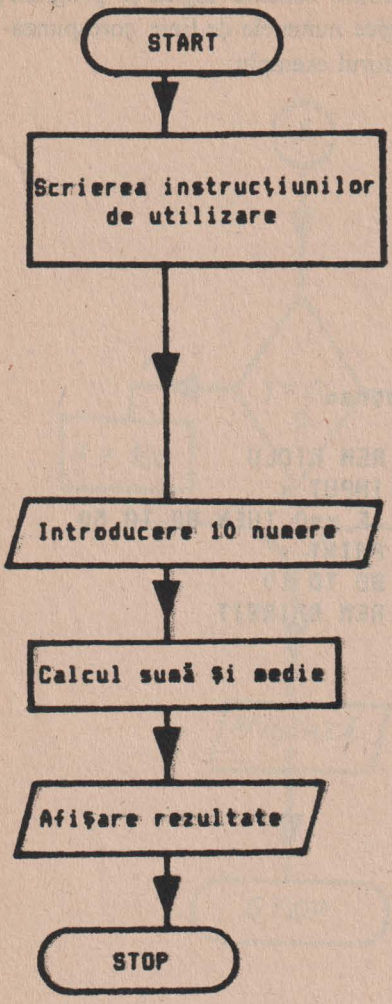


Fig. 20

Observați că schemele logice pe care le-am prezentat reprezintă programele, linie cu linie. Se pot însă, realiza, scheme logice mai puțin detaliate, ale căror blocuri să reprezinte secvențe de mai multe instrucțiuni din program sau chiar module. Astfel, se pot realiza și scheme logice care să arate ca în fig. 20.

Fig. 21

Fiecare secțiune din acest tip de schemă se poate detalia în mai multe blocuri identificabile fiecare cu cîte o linie de program. De asemenea, se pot adăuga comentarii referitoare la liniile de program reprezentate printr-un bloc (vezi fig. 21).

```
... ciclu de introducere
    liniile 40-60
```

Pentru programe cu un număr mare de linii, se recomandă concentrarea schemei logice (la o mărime ușor de utilizat), dar numai pe porțiunile pentru care secvențele de instrucțiuni sînt ușor de urmărit și în program. Orice secvență mai complicată trebuie, însă, inclusă în detaliu. **Iată o schemă logică de acest tip, pe baza căreia se poate programa robotul să cumpere un pachet de ceai** (vezi fig. 22):

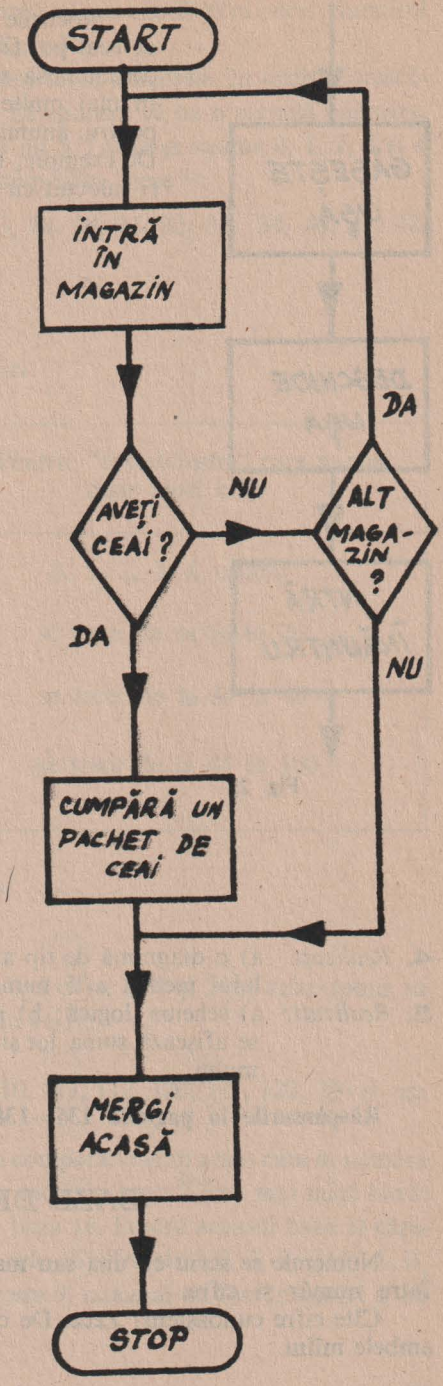


Fig. 22

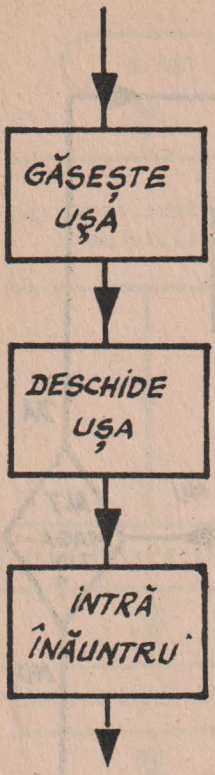


Fig. 23

Blocurile din schema logică a cumpărării unui pachet de ceai pot fi și ele descompuse în același mod în care era descompusă sarcina robotului de a face o ceașcă de ceai în mai multe subsarcini, spre a putea fi executată. Deci, pentru anumite puncte, schema logică va trebui extinsă. De exemplu, blocul logic "INTRA ÎN MAGAZIN" poate fi înlocuit cu (vezi fig. 23):

4. *Realizați:* a) o diagramă de tip arbore; b) schema logică și c) programul calculului mediei a 5 numere.
5. *Realizați:* a) schema logică; b) programul cu care se introduc două numere, se afișează suma lor și se întreabă dacă se dorește reluarea programului.

Răspunsurile la paginile 136—138

BAZE DE NUMERAȚIE

Numerele se scriu cu una sau mai multe cifre. De aici apare uneori confuzia între număr și cifre.

Cîte cifre cunoaștem? Zece. De ce zece? Poate pentru că avem zece degete la ambele mîini.

Cifrele arabe sînt 0, 1, 2, 3, ..., 9. Remarcați că nu există cifra zece. Numărul zece este compus din cifrele 1 și 0 scrise în ordinea 10.

Să ne jucăm puțin. Ne propunem să scriem toate numerele în ordine crescătoare, formate numai cu cifrele 0, 1, 2, 3 și 4. Să spunem că pe o planetă oarecare, ființele de acolo nu cunosc decît cinci (atenție nu 5!) cifre și anume 0, 1, 2, 3 și 4 (deci, fără cifra "5" — așa cum noi nu cunoaștem cifra "zece"):

0, 1, 2, 3, **4**, **10**, 11, 12, 13, **14**, **20**, 21, 22, 23, **24**, **30**, 31, 32, 33, **34**, **40**, 41, 42, 43, **44**, **100**, 101, 102, etc.

Ce observăm?
Urmăriți tabelul de asemănări și deosebiri:

Pentru "pămînteni" care cunosc zece cifre	Pentru "extratereștri" care cunosc doar cinci cifre
0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ultima	0, 1, 2, 3, **4** ultima
trecem de la 19 la 20	se trece de la 14 la 20
trecem de la 39 la 40	se trece de la 34 la 40
trecem de la 99 la 100	se trece de la 44 la 100

Într-un cuvînt, totul se poate rezuma la:
"cifra 4 joacă rol de cifra 9"

Dacă, de exemplu, vrem să folosim doar trei cifre, 0, 1 și 2 (și vom spune că numărăm în baza TREI), atunci numărătoarea va fi:

0, 1, 2, 10, 11, 12, 20, 21, 22, 100, 101, 102, 110, 111, 112, 120, 121, 122, 200 și așa mai departe.

Să urmărim mai bine un tabel în care, prin comparație vom arăta cum se numără în diverse baze de numerație. Observați că am inclus și baza 2 (cea mai mică bază) și, totodată, o bază mai mare ca 10, și anume baza 16. Pentru această bază al cărei sistem de numerație se numește hexazecimal, cele 16 cifre sînt: 0, 1, 2, ... 9, A, B, C, D, E și F. Observați că nemaiavînd cifre peste 9, oamenii au fost inventivi și au continuat cu literele alfabetului.

BAZA 10 totodată nr.curent	BAZA 2	BAZA 3	BAZA 4	BAZA 7	BAZA 8	BAZA 10	BAZA 16
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	10	2	2	2	2	2	2
3	11	10	3	3	3	3	3
4	100	11	10	4	4	4	4
5	101	12	11	5	5	5	5
6	110	20	12	6	6	6	6
7	111	21	13	10	7	7	7
8	1000	22	20	11	10	8	8
9	1001	100	21	12	11	9	9
10	1010	101	22	13	12	10	A
11	1011	102	23	14	13	11	B
12	1100	110	30	15	14	12	C
13	1101	111	31	16	15	13	D
14	1110	112	32	20	16	14	E
15	1111	120	33	21	17	15	F
16	10000	121	100	22	20	16	10
17	10001	122	101	23	21	17	11
18	10010	200	102	24	22	18	12
19	10011	201	103	25	23	19	13
20	10100	202	110	26	24	20	14
21	10101	210	111	30	25	21	15
22	10110	211	112	31	26	22	16
23	10111	212	113	32	27	23	17
24	11000	220	120	33	30	24	18
25	11001	221	121	34	31	25	19
26	11010	222	122	35	32	26	1A
27	11011	1000	123	36	33	27	1B
28	11100	1001	130	40	34	28	1C
29	11101	1002	131	41	35	29	1D
30	11110	1010	132	42	36	30	1E
31	11111	1011	133	43	37	31	1F
32	100000	1012	200	44	40	32	20
33	100001	1020	201	45	41	33	21
34	100010	1021	202	46	42	34	22
35	100011	1022	203	50	43	35	23
36 ...etc.	100100	1100	210	51	44	36	24
BAZA 10 cifre folosite	BAZA 2 0...1	BAZA 3 0...2	BAZA 4 0..3	BAZA 7 0...6	BAZA 8 0...7	BAZA 10 0...9	BAZA 16 0...A

După analizarea acestui tabel să concluzionăm cîteva lucruri:

. ● Mai întîi, ca să știm din ce coloană am luat un număr vom scrie jos, ca indice, numărul bazei (de exemplu, numărul 7 se scrie în baza 3 ca 21_3). Dacă nu se scrie nici un indice se va înțelege baza 10 (cu care sîntem obișnuiți).

deci $7(= 7_{10}) = 111_2 = 21_3 = 13_4 = 10_7 = 7_8 = 7_{16}$

sau $29 = 11101_2 = 1002_3 = 131_4 = 41_7 = 35_8 = 1D_{16}$

● Știm că în modul nostru curent de numerație adăugarea la sfîrșitul numărului a unu, doi sau trei de zero este echivalentă cu înmulțirea cu 10, 100 și, respectiv, 1 000, adică 10^1, 10^2 și 10^3. În altă bază (fie ea b), adăugarea a 1, 2 sau 3 de 0 produce înmulțirea nu cu 10, 100 și, respectiv, 1 000, ci cu b, b^2 și, respectiv b^3. Într-adevăr:

$\begin{cases} 11_3 = 4 \\ 1100_3 = 36 \end{cases}$ de 3^2 ori mai mare $(9 \times 4 = 36)$

$\begin{cases} 5_7 = 5 \\ 50_7 = 35 \end{cases}$ de 7^1 ori mai mare $(7 \times 5 = 35)$

$\begin{cases} 101_2 = 5 \\ 101000_2 = 40 \end{cases}$ de 2^3 ori mai mare $(8 \times 5 = 40)$

Priviți următorul tabel:

limbaj natural	mii	sute	zeci	unități
puteri ale bazei	10^3	10^2	10^1	10^0
valoarea puterii bazei	1000	100	10	1
scrierea numărului	1	9	8	7

valoare 1987

Construcții de forma **una mie nouă sute optzeci și șapte** (unități) se întîlnesc în toate limbile și prin ele se identifică numărul (valoarea) care se obține prin:

de 1 ori 1 000 plus
de 9 ori 100 plus
de 8 ori 10 plus
de 7 ori 1

Să analizăm ce diferențe apar dacă se lucrează cu altă bază de numerație (7):

limbaj natural	—	—	—	—	—	UNITĂȚI
puteri ale bazei	7^5	7^4	7^3	7^2	7^1	7^0
valoarea puterii bazei	16807	2401	343	49	7	1
scrierea numărului	0	0	5	5	3	6

valoare 1987

După cum se vede din tabel, valoarea se obține astfel:

de 5 ori 7^3 plus	(un fel de "mii")
de 5 ori 7^2 plus	(un fel de "sute")
de 3 ori 7^1 plus	(un fel de "zeci")
de 6 ori 7^0	
TOTAL 1987	

Deci $5536_7 = 1987$

Ca temă verificați și următoarele egalități: $1987 = 133003_4 = 11111000011_2 = 7C3_{16} = 7C3H = 7C3h$.

Deoarece baza 16 este des folosită, rețineți că această bază se mai "anunță" și prin literele H sau h (această literă nefăcînd parte din suita cifrelor folosite în sistemul hexazecimal — 0, 1, 2, ..., F — nu pot apărea confuzii).

După aceste indicații (reguli), ne va fi foarte ușor să găsim echivalentul zecimal al unui număr în orice bază.

6. *Găsiți echivalentul zecimal al următoarelor numere:*

a) $(101101)_2$

b) (3A7)H

Răspunsul la pagina 139.

Baza doi și octetul

Să presupunem că între două calculatoare sau între două părți componente ale aceluiași calculator trebuie să se transmită numărul 1987.

Conform figurii 24, pe cele 4 fire vor trebui transmise semnele (impulsuri) electrice care să simbolizeze cifrele din care este format numărul (1, 9, 8 și 7). Re-

marcați că scrierea 1987 sau pronunțarea acestui număr nu au un suport fizic adecvat mașinilor electronice (în primul caz simbolurile grafice de pe o foaie de hîrtie sau sunetele, pentru al doilea caz, nu pot fi transmise direct unui calculator). Trebuie să găsim un suport fizic. În fig. 24, s-a ales ca suport fizic patru fire conducătoare

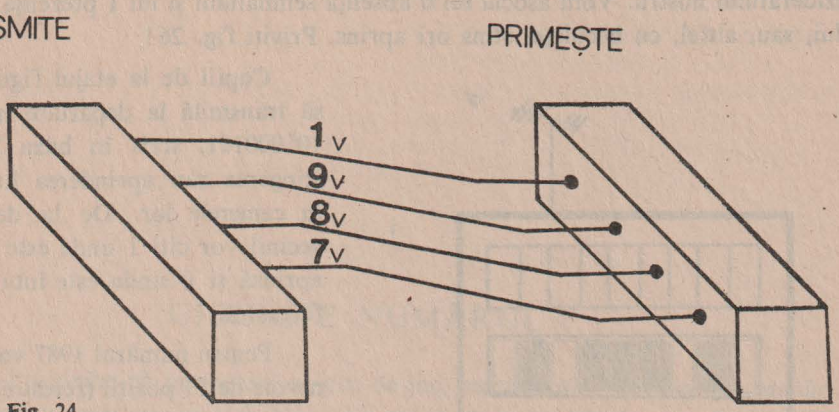


Fig. 24

prin care se trimit semnele de 0, 1, 2, ..., 9 volți, după cum indică cifra pe care vrem să o transmitem. Am putea păstra sau transmite numărul 1987 și prin alt suport fizic, de exemplu, o suită de bețe de chibrituri de diferite dimensiuni, ca în fig. 25.

Este dificil ca un circuit electronic să poată analiza exact dacă semnalul pe care l-a primit este mai aproape de 7 volți sau mai aproape de 8 volți (pot să apară perturbații pe linia de transmisie care să producă la recepție 7,9 volți în loc de 8 volți sau chiar mai puțin).

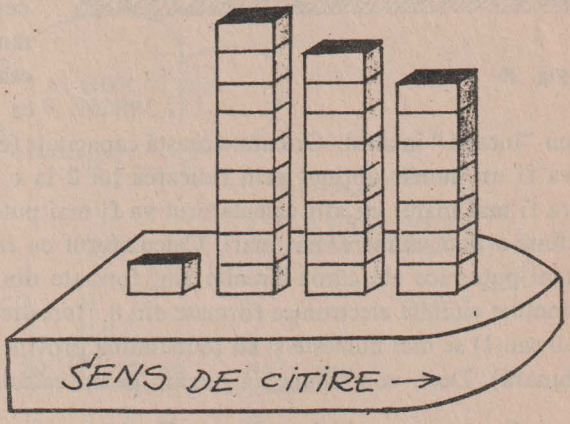


Fig. 25

Ceva asemănător s-ar putea produce dacă de la depărtare cineva ar vrea să vă transmită un număr folosind, să zicem, sticle de diferite înălțimi, puse la fiecare dintre ferestrele unui bloc. Ei bine, vă va fi greu să apreciați exact relațiile de proporționalitate între înălțimile diferitelor "cifre de sticlă". Ar fi mult mai ușor dacă nu ați avea de observat decît existența sau nonexistența unei sticle la fiecare fereastră (DA sau NU).

Deci, mult mai simplu este să realizăm un circuit care să sesizeze doar dacă a apărut sau nu un semnal.

Acum ați înțeles de ce baza doi este pe primul plan în informatică? În sistemul binar de numerație (baza 2), cele două cifre cunoscute sînt 0 și 1, ceea ce corespunde dezideratului nostru. Vom asocia lui 0 absența semnalului și lui 1 prezența semnalului, sau, altfel, cu beculețe, stins ori aprins. Priviți fig. 26!

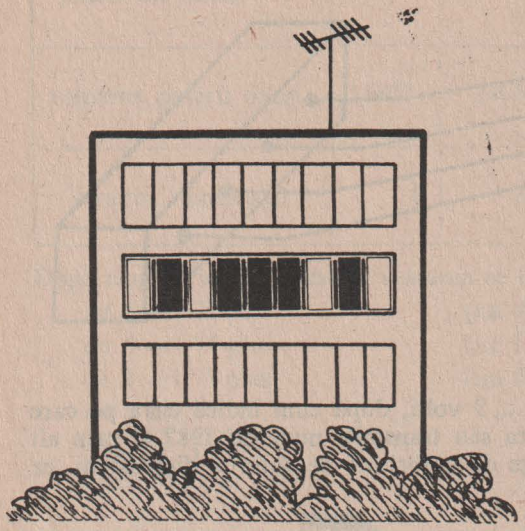


Fig. 26

Copiii de la etajul figurat vor să transmită la depărtare numărul 101000101, scris în baza 2, prin stingerea sau aprinderea luminilor în camerele lor. De la depărtare vecinii vor citi 1 unde este lumina aprinsă și 0 unde este întuneric la fereastră.

Pentru numărul 1987 vom avea nevoie de 11 poziții (ferestre), deoarece cum am mai arătat $1987 = 11111000011_2$.

În calculator, circuitele între care pot avea loc transmisii de date (numere) trebuie să aibe aceeași capacitate (sau altfel spus același număr de "ferestre"), deoarece în caz contrar s-ar putea întîmpla ca ce "pleacă" dintr-un circuit să nu "încapă" în altul. Oricum, această capacitate (exprimată în număr de "ferestre") va fi un număr obținut prin ridicarea lui 2 la o anumită putere. Cu cît numărul va fi mai mare, cu atît calculatorul va fi mai puternic, el putînd, astfel, manipula dintr-o dată numere mai mari. Calculatorul cu care lucrați (există și calculatoare mai puternice ale căror circuite sînt formate din 16, 32 sau chiar 64 "ferestre") conține circuite electronice formate din 8 "ferestre". În informatică, o cifră binară (0 sau 1) se mai numește și **bit** (denumirea provine din engleză, **binary digit** = cifră binară). Deci, vom spune că lucrăm pe un calculator de 8 biți.

O succesiune de 8 astfel de cifre binare se mai numește **octet**.

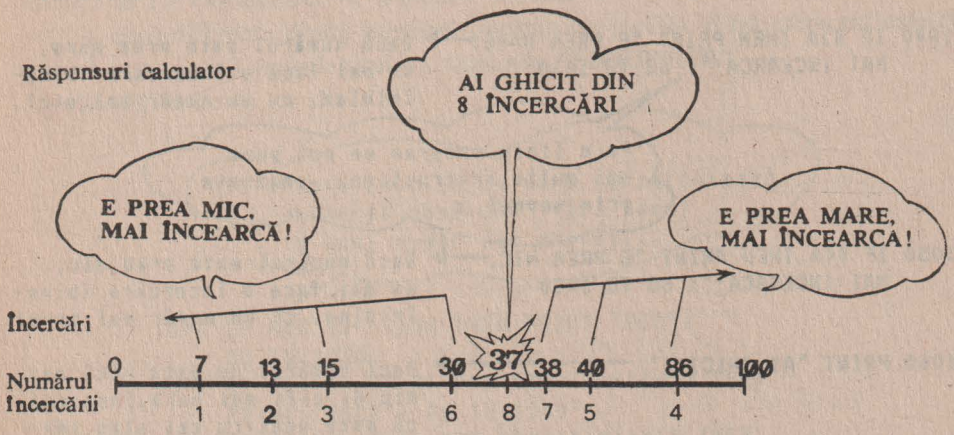
7. *Care este cel mai mare număr care se poate înscrie într-un octet? Dar cel mai mic?*

Răspunsul la pagina 139.

II. Începem să realizăm jocuri

GHICEȘTE NUMĂRUL

Să ne imaginăm că partenerul nostru de joc, calculatorul, își alege un număr întreg și pozitiv, mai mic ca 100. Vom încerca să găsim numărul respectiv, putînd efectua mai multe încercări (introduceri de numere). Dacă numărul introdus este mai mic decît cel ales, calculatorul va da răspunsul "E PREA MIC. MAI ÎNCEARCA!". Dacă numărul introdus este mai mare, calculatorul va da răspunsul "E PREA MARE. MAI INCEARCA!".



Calculatorul își alege numărul 37.

Fig. 27 GHICEȘTE NUMĂRUL

Restrîngînd intervalul în care se găsește numărul, vom reuși să-l găsim după un anumit număr de încercări (vezi fig. 27). Haideți să încercăm să dăm viață acestui joc, pe care o să-l numim „**Ghicește numărul**", cu ajutorul unui program.

Tastați programul exact cum apare mai jos, iar, apoi, pentru a vă juca, nu uitați să introduceți comanda **RUN**:

```
PROGRAM JOC: GHICESTE NUMARUL

Atenție ! La salvarea jocului pe caseta magnetică
          va trebui să puneți un nume care să nu
          depășească 10 caractere, altfel calcula-
          torul nu va accepta numele programului.

                    Numerele de linie pot fi
                    de la 1 la 9999.

1000 REM ** JOC GHICESTE NUMARUL **
         ***************************
1010 PRINT "GHICESTE NUMARUL"
1020 LET A=INT (RND*100+1)  ---> Calculatorul își alege un nu-
                                 măr oarecare, întreg și pozi-
                                 tiv mai mic sau egal cu 100,
                                 pe care îl va memora în vari-
                                 abila A.

1030 INPUT "TASTEAZA NUMARUL ";B ---> Numărul introdus de jucător
                                      este memorat în variabila B.

1040 IF B>A THEN PRINT "E PREA MARE:  ---> Dacă numărul este prea mare,
     MAI INCEARCA!": GO TO 1030            se mai face o încercare (bine-
                                           înțeles, cu un număr mai mic).

                    Pe o linie program se pot pune
                    mai multe instrucțiuni, separate
                    prin semnul :

1050 IF B<A THEN PRINT "E PREA MIC.  ---> Dacă numărul este prea mic,
     MAI INCEARCA!": GO TO 1030            se mai face o încercare (bine-
                                           înțeles, cu un număr mai mare).

1060 PRINT "AI GHICIT!"  ---------------> Dacă numărul nu este nici mai
                                          mic și nici mai mare, înseamnă
                                          că este egal cu cel ales ini-
                                          țial de calculator.

1061 PRINT
1070 STOP
```

Jocul se va derula așa cum apare în fig. 28.

```
GHICESTE NUMARUL

TASTEAZA NUMARUL: 7
E PREA MIC. MAI INCEARCA
TASTEAZA NUMARUL: 13
E PREA MIC. MAI INCEARCA
TASTEAZA NUMARUL: 15
E PREA MIC. MAI INCEARCA
TASTEAZA NUMARUL: 86
E PREA MARE. MAI INCEARCA
```

Fig. 28

Puteți realiza și o altă variantă de joc, care vă va aduce aminte de jocul cu obiectul ascuns pe care, sigur, l-ați mai jucat, dar fără ajutorul calculatorului. Acum, pentru găsirea numărului, calculatorul vă va da următoarele indicații:

— dacă diferența dintre numărul introdus de jucător și cel memorat este mai mare decît 50, calculatorul va răspunde "GER";

— dacă diferența dintre numărul introdus de jucător și cel memorat este cuprinsă între 15 și 50, calculatorul va răspunde "RECE";

— dacă diferența dintre numărul introdus de jucător și cel memorat este cuprinsă între 3 și 15, calculatorul va răspunde "CALD";

— dacă diferența dintre numărul introdus de jucător și cel memorat este cuprinsă între 1 și 3, calculatorul va răspunde "FIERBINTE".

Modificările de joc sînt:

ABS este o funcție cu care se calculează valoarea absolută (fără semn) a unui număr sau expresii.

```
1040 IF ABS (B-A)>50 THEN PRINT "GER":
     GO TO 1030
1045 IF 15<ABS (B-A) AND ABS (B-A)<=50 THEN
     PRINT "RECE": GO TO 1030
1050 IF 3<ABS (B-A) AND ABS (B-A)<=15 THEN
     PRINT "CALD": GO TO 1030
1055 IF 1<=ABS (B-A) AND ABS (B-A)<=3 THEN
     PRINT "FIERBINTE": GO TO 1030
```

Observați cum se testează, într-un program, dacă un număr aparține unui interval: trebuie ca ambele condiții (numărul să fie mai mare ca marginea din stînga și mai

mic ca marginea din dreapta) să fie adevărate. Pentru aceasta se folosește particula **AND** (ȘI).

Sigur, veți dori să mai jucați o dată sau de mai multe ori jocul.

Cum se poate realiza acest lucru fără să mai introduceți comanda **RUN**? Simplu!

Introduceți următoarele linii:

```
1063 REM CONTINUAREA JOCULUI
1065 PRINT "ALT JOC ? (D/N)"
1067 INPUT C$
1069 IF C$="D" OR C$="d" THEN ⟶ Dacă la întrebarea calculato-
     GO TO 1000                 rului se acționează tasta D
                                (literă mare sau literă mică)
                                atunci se va mai putea juca
                                o dată.
```

Acum puteți juca foarte bine avînd ca partener calculatorul. Haideți să vedem cum putem face ca la joc să participe mai mulți copii, care să se întreacă între ei!

Aveți vreo idee?

Sigur că da! V-ați adus aminte de numărarea reușitelor! Putem calcula de cîte încercări a fost nevoie pentru ghicirea numărului. Vom păstra acest număr într-o variabilă **I**, care la începerea jocului (de către fiecare participant) va avea valoarea inițială **O**. Deci, introducem linia:

```
1025 LET I=0
```

Cînd vrem să calculăm o sumă cu ajutorul calculatorului, folosim o variabilă căreia îi dăm la început valoarea 0.
Cînd vrem să calculăm un produs cu ajutorul calculatorului, folosim o variabilă căreia îi dăm la început valoarea 1.
Exemplu pentru calculul sumei și al produsului primelor 10 numere naturale:

```
Calculul sumei (S)          Calculul produsului (P)

10 LET S=0                  10 LET P=1
20 FOR I=1 TO 10            20 FOR I=1 TO 10
30 LET S=S+I                30 LET P=P*I
40 NEXT I                   40 NEXT I
50 PRINT "SUMA=";S          50 PRINT "PRODUSUL=";P
```

și după fiecare încercare adăugăm o unitate valorii pe care o are variabila **I**:

```
1035 LET I=I+1
```

Nu uităm să modificăm și linia 1060, pentru a indica, la fiecare joc, cîte încercări au fost necesare pentru ghicirea numărului:

```
1060 PRINT "AI GHICIT DIN ";I;" INCERCARI!"
```

Mai aveți vreo idee? Da! Calculatorul poate cere la începutul fiecărui joc numele jucătorului, iar, în final, să afișeze numele campionului sau un clasament al jucătorilor cu rezultatele obținute.

8. *De ce s-a scris* **+1** *în linia* **1020** *pentru a se genera un număr întreg aleator mai mic sau egal cu* **100**?

Răspunsul la pagina 139.

Cine este campionul ?

Să presupunem că vom construi jocul "**Ghicește numărul**", astfel încît să se întrecă între ei 5 jucători: Ștefan, Daniela, Oana, Petrică și Știe-Tot.

Cu rezultatele obținute de fiecare (din cîte încercări a reușit să găsească numărul) putem forma o listă de 5 numere. Ca să facem un clasament, va trebui să memorăm fiecare din aceste numere în cîte o variabilă. Dar ce ne facem atunci cînd va trebui să memorăm 100 sau chiar 1000 de valori? Vom defini cîte o variabilă pentru fiecare din ele? Lucrul nu este imposibil, dar foarte greu de realizat; va fi însă și mai greu să găsim aceste valori și să ținem evidența acestor variabile.

Există totuși un mecanism cu ajutorul căruia putem memora o listă de numere de același tip și, anume, se poate defini o *variabilă de tip tablou*. Elementele ei reprezintă un set de variabile, toate avînd același nume și distingîndu-se numai printr-un număr (index) scris între paranteze după nume. De aceea, variabila de tip tablou **A** se mai numește și *indexată*. *A(3)* va reprezenta, de exemplu, valoarea variabilei care ocupă a treia poziție în cadrul variabilei tablou **A**.

Înainte de a se utiliza o variabilă tablou, **trebuie rezervat pentru ea un spațiu de memorie suficient**; acesta se realizează cu instrucțiunea **DIM**. Denumirea provine de la DIMENSION (DIMENSIUNE) și arată că trebuie precizată dimensiunea

maximă a spațiului de memorie care va trebui rezervat (vezi fig. 29). Pentru definirea unei variabile de tip tablou, este indicată folosirea instrucțiunii DIM la începutul programului, ea avînd următorul efect:

— rezervă spațiul de memorie necesar tabloului definit:

— inițializează toate elementele tabloului cu O.

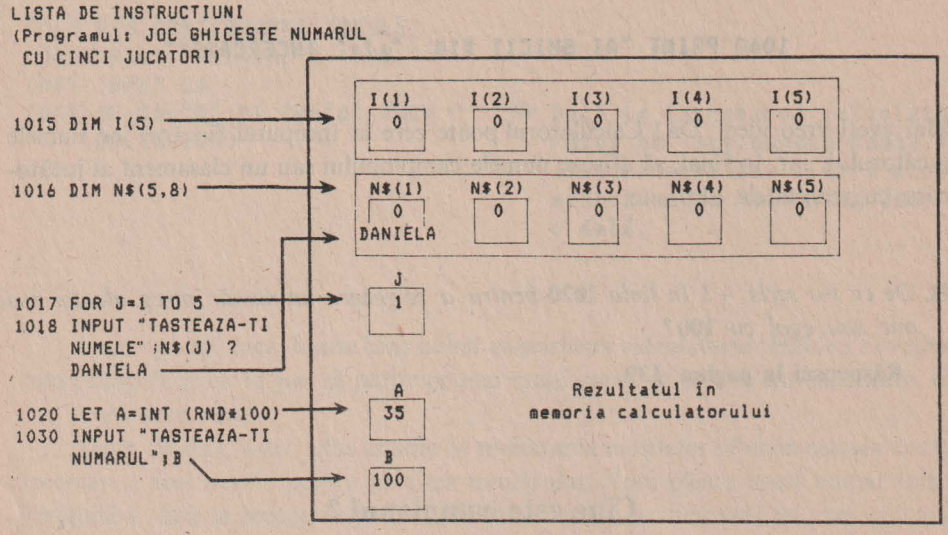


Fig. 29

Nota **1** Numele variabilei de tip tablou trebuie reprezentat printr-o singură literă, spre deosebire de variabilele obișnuite, care pot fi reprezentate printr-un șir de mai multe litere.

Nota **2** Nu este obligatorie atribuirea de valori tuturor elementelor lui **A** într-un program, dar, dacă rămîn mai multe elemente neatribuite, spunem că spațiul de memorie nu a fost eficient folosit.

Nota **3** Într-un program pot exista împreună o variabilă de tip tablou și o variabilă simplă cu același nume, fără ca acest lucru să provoace confuzii.

Nota **4** Se pot defini și variabile de tip tablou cu mai multe dimensiuni. Pentru a se indica poziția unui caracter pe ecranul unui televizor este nevoie să se specifice linia și coloana caracterului respectiv. În mod similar, pentru a se specifica un element al unei variabile cu două dimensiuni, sînt necesare două numere. Astfel, putem să ne imaginăm variabila de tip tablou cu 2 dimensiuni ca avînd forma unui ecran (tablă). De exemplu, o varia-

bilă (C) de tip tablou, cu 2 dimensiuni (3 și 6), stabilită printr-o instrucțiune **DIM C(3,6)**, va avea următoarele 18 elemente:

C(1,1), C(1,2) ... C(1,6)
C(2,1), C(2,2) ... C(2,6)
C(3,1), C(3,2) ... C(3,6)

Aceleași principii se aplică pentru oricîte dimensiuni.

Nota **5** Se pot defini și variabile de tip tablou de șiruri de caractere. Ele se deosebesc prin semnul **$**, pus după numele variabilei.

Șirurile de caractere din variabilele de tip tablou diferă de șirurile de caractere obișnuite prin aceea că elej sînt de **lungime fixă**. Astfel **DIM a (5,10)** definește o variabilă de tip tablou ale cărei elemente sînt 5 șiruri de cîte 10 caractere:

a (1) = a (1,1) a (1,2) ... a (1,10)

.

.

.

a (2) = q (2,1) a (2,2) ... a (2,10)

a (5) = a (5,1) a (5,2) ... a (5,10)

Să vedem cum va arăta jocul „**Ghicește numărul**" jucat de 5 jucători. Înainte de a începe să tastați jocul, nu uitați să introduceți comanda **NEW** pentru a șterge vechiul program din memoria calculatorului, deoarece noua variantă diferă în mai

```
1000 REM ** JOC GHICESTE NUMARUL **
         *  CU 5 JUCATORI          *
         ****************************
1010 PRINT "GHICESTE NUMARUL"
1015 DIM I(5)  ------>  Se rezervă un spațiu suficient de mare
                        în memoria calculatorului pentru a se
                        păstra cele 5 numere care reprezintă
                        scorul. I(3) va arăta din cîte încer-
                        cări a reușit cel de-al treilea jucă-
                        tor să găsească numărul.

1016 DIM N$(5,8)  ------>  Se rezervă un spațiu suficient de mare
                           pentru a memora numele celor 5 jucă-
                           tori care participă. Cel mai lung nume
                           va fi de maxim 8 caractere (STIE-TOT).
                           N$(3) va reprezenta numele celui de al
                           treilea jucător.
```

```
1017 FOR J=1 TO 5 ──────► Jocul se joacă de 5 ori (de către
                          fiecare participant).

1018 INPUT "TASTEAZA-TI NUMELE ";N$(J)
1020 LET A=INT (RND*(100+1))
1025 LET I(J)=0 ──────► Pentru a calcula din cîte încercari s-a
                        găsit numărul (suma), se inițializează cu
                        zero variabila care reprezintă numărul de
                        încercări, înainte de prima încercare.

1030 INPUT "TASTEAZA NUMARUL ";B
1035 LET I(J)=I(J)+1 ──────► După fiecare încercare, numă-
                        rul de încercări crește cu o unitate.
1036 REM MOTORUL JOCULUI
1040 IF B>A THEN PRINT "E PREA MARE.
     MAI INCEARCA!": GO TO 1030
1050 IF B<A THEN PRINT "E PREA MIC.
     MAI INCEARCA!": GO TO 1030
```

Apostroful (') într-o linie PRINT va determina continuarea afișării pe rîndul următor (salt pe rîndul următor).

```
1060 PRINT N$(J)'"AI GHICIT DIN
     ";I(J);" INCERCARI"
1061 PRINT
1063 NEXT J
1065 REM CINE E CAMPIONUL ?
```

multe puncte de cea precedentă. Totuși, puteți încerca și pe vechea variantă să modificați liniile care diferă, neuitînd să le ștergeți pe acelea care nu mai există în noul program.

Campion va fi acela care a ghicit numărul din cele mai puține încercări, deci acel (J) pentru care numărul de încercări (I) va fi cel mai mic. Așadar, se pune problema să găsim dintr-o listă de numere I(1), I(2) ... I(5) care este cel mai mic (minimul) și să memorăm cărui indice J îi corespunde acesta.

Vom folosi următorul raționament, pe care o să-l numim **algoritmul pentru determinarea minimului dintr-o listă de numere**: considerăm la început că minimul este chiar primul număr din listă. Apoi, îl comparăm cu următorul număr (al doilea din listă). Dacă acesta este mai mic îl vom alege pe el minimul, dacă nu, vom compara minimul cu următorul număr de listă. Vom repeta procedeul pînă cînd lista de numere se va epuiza. În final, minimul va fi chiar cel mai mic număr din listă.

Pentru ca programul nostru de joc să ne indice care este campionul, vom completa programul anterior cu următoarele linii:

```
1070 LET MIN=I(1)
```
Consideră ca fiind minim primul element din listă.

```
1080 FOR J=2 TO 5
1090 IF MIN<=I(J) THEN GO TO 1120
```
Compară minimul cu următorul element din listă. Dacă acesta este mai mare, se trece la următoarea comparare.

```
1100 LET MIN=I(J)
```
Dacă este mai mic, atunci îl alege pe el ca minim.

```
1110 LET C=J
```
Nu uită să salvăm indicele minimului (J) în altă variabilă (C), pentru a şti cărui nume îi corespunde minimul.

```
1120 NEXT J
```
Comparările se fac până când lista se epuizează.

```
1130 PRINT N$(C);" ESTE CAMPION"'
     '"A GHICIT NUMARUL DIN";MIN;
     " INCERCARI"
```
După ce lista s-a terminat, se afişează rezultatul.

Calculatorul joacă numai cu prieteni

Să încercăm să facem calculatorul să se joace numai cu cine vrem noi. Pentru aceasta avem două posibilităţi:

1. cerem calculatorului să solicite introducerea unei parole (să zicem "SESAM") şi dacă jucătorul nu o va cunoaşte, nu se va putea juca. Bineînţeles, vom avea grijă să aducem la cunoştinţă prietenilor care este parola. Vom adăuga programului "**Ghiceşte numărul**" următoarele linii:

```
100 REM ** JOC CU PAROLA **
      ********************
110 INPUT "INTRODUCETI PAROLA ";P$
120 IF P$="SESAM" THEN GO TO 1000
130 PRINT "ALTA DATA ! LA REVEDERE !"
140 STOP
```

2. să facem calculatorul să joace cu prietenii pe care îi vom da într-o listă de nume.

Adăugăm jocului **"Ghicește numărul"** următoarele linii:

> **Cînd într-un program se vor folosi mai multe date care nu necesită modificări ulterioare, acestea se pot introduce cu ajutorul instrucțiunilor READ - DATA**

```
100 REM ** CALCULATORUL JOACA NUMAI **
      *       CU PRIETENI           *
      *********************************
110 DATA "STEFAN","DANIELA","OANA",
    "PETRICA","STIE-TOT"
120 DIM N$(5,8)
130 FOR I=1 TO 5
140 READ N$(I)
150 NEXT I
160 LET K=0
170 PRINT "TASTEAZA, TE ROG, NUMELE TAU"
180 INPUT X$
190 LET L=LEN (X$)
200 FOR I=1 TO 5
210 IF X$=N$(I)(1 TO L) THEN
    LET K=1
220 NEXT I
230 IF K=0 THEN GO TO 260
```

Lista de nume pe care calculatorul le va recunoaște.

Se rezervă în memoria calculatorului un spațiu N$, suficient de mare pentru ca să se memoreze cele 5 nume.

Linia 140 se repetă de 5 ori și de fiecare dată unul din nume se va pune în N$.

Răspunsul va fi memorat în variabila X$. Pentru a introduce un nume format din mai mult de 8 litere, se poate modifica linia 120. De exemplu: DIM N$(5,20).

LEN (X$) arată cîte caractere sînt în variabila X$ (lungimea).

De fiecare dată cînd se execută linia 210, calculatorul compară X$ cu unul din numele memorate în variabila N$. Dacă X$ este identic cu unul din aceste nume, variabila K va deveni 1.

Dacă nici unul din nume nu este identic cu X$, atunci K va rămîne 0 și calculatorul va afișa mesajul din linia 260.

```
240 PRINT "TE CUNOSC ";X$ ──────► Dacă K=1, recunoaşte persoana
250 GO TO 1000                    şi jocul va putea începe.

260 PRINT "NU MA JOC CU ────────► Dacă A=0, calculatorul ignoră
    PERSOANE PE CARE NU LE        liniile 240 şi 250 şi afişează
    CUNOSC"                       mesajul din linia 260.
270 STOP
```

CALCULATORUL PROFESOR

Iată trei jocuri cu ajutorul cărora puteţi să testaţi cunoştinţele voastre sau ale prietenilor la geografie, istorie, limbi străine şi matematică.

Care este capitala?

> Tastaţi toate datele şi răspunsurile cu litere mari, deoarece calculatorul tratează literele mici şi cele mari ca litere diferite.

```
10 REM ** CARE ESTE CAPITALA ? **
       **************************
20 PRINT "CARE ESTE CAPITALA ?"
30 PRINT
40 LET N=5 ──────────────────► Variabila N memorează numărul de
                               întrebări care vor fi puse.

50 LET C=0 ──────────────────► Variabila C memorează numărul de
                               răspunsuri corecte.

60 LET G=0 ──────────────────► Variabila G memorează numărul de
                               răspunsuri greşite.

70 DATA "FRANTA","PARIS"   ┐
80 DATA "BULGARIA","SOFIA" ├► Perechi de întrebări şi
90 DATA "CHINA","BEIJING"  │  răspunsuri.
100 DATA "CANADA","OTTAWA" │
110 DATA "ITALIA","ROMA"   ┘

120 FOR I=1 TO N ────────────► Începutul ciclului care repetă
                               liniile 120-190.

130 READ X$,Y$ ──────────────► De fiecare dată cînd bucla este
140 PRINT "CARE ESTE CAPITALA  repetată, calculatorul ia una din
    TARII ";X$;" ?"            perechile de nume şi le memorează
                               în variabilele X$ şi Y$.

150 INPUT Z$ ────────────────► Răspunsul este memorat în varia-
                               bila Z$.
```

```
160 IF Y$=Z$ THEN LET C=C+1:
    PRINT "CORECT"

170 IF Y$<>Z$ THEN LET G=G+1:
    PRINT "GRESIT"
180 PRINT

190 NEXT I

200 PRINT
210 PRINT N;" INTREBARI"
220 PRINT "-----------"
230 PRINT "CORECTE: ";C
240 PRINT "GRESITE: ";G
```

Compararea răspunsului dat (Z$) cu răspunsul corect (Y$).

Păstrarea scorului (C - răspunsuri corecte, G - răspunsuri greşite) şi afişarea mesajului CORECT sau GRESIT.

Calculatorul se întoarce la începutul ciclului (în linia 120), ia următoarea pereche de nume şi pune întrebarea.

Afişarea scorului final.

Programul testează cunoştinţele despre capitalele ţărilor. Calculatorul pune o întrebare şi aşteaptă introducerea răspunsului, după care anunţă dacă răspunsul a fost corect sau greşit (vezi fig. 30).

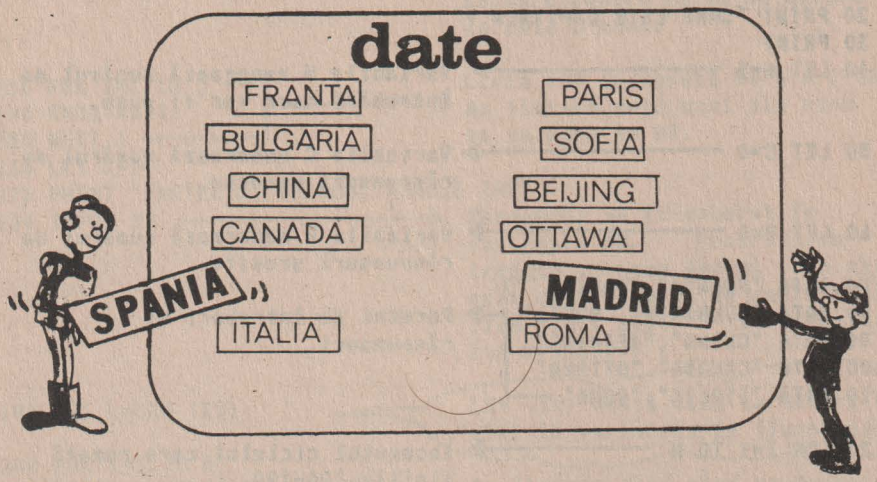


Fig. 30

CALCULATORUL PROFESOR DE LIMBI STRĂINE

Calculatorul se poate transforma lesne din profesor de geografie în profesor de limbi străine. Pentru aceasta se vor efectua următoarele modificări (sau adăugiri) asupra programului **"Care este capitala?"**:

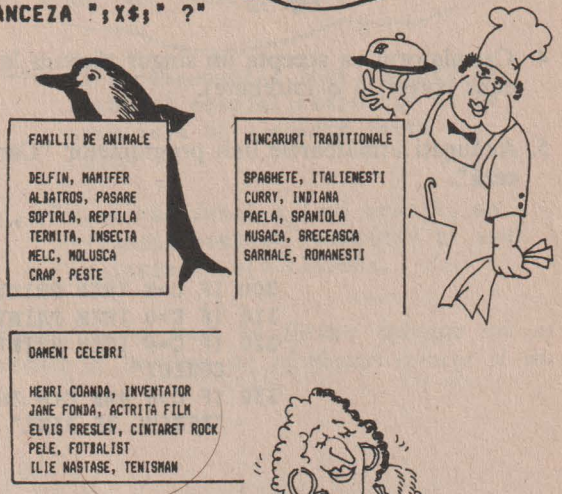
```
10 REM ** TEST DE FRANCEZA **
     **********************
20 PRINT "TEST DE FRANCEZA"
40 LET N=7
 .
 .
 .
70 DATA "UMBRELA","LE PARAPLUIE"
80 DATA "SCAUN","LA CHAISE"
90 DATA "MARINAR","LE MARIN"
100 DATA "PIPER","LE POIVRE"
110 DATA "ROCHIE","LA ROBE"
111 DATA "NAS","LE NEZ"
112 DATA "GRADINA","LE JARDIN"
 .
 .
 .
 .
 .
140 PRINT "CUM SE SPUNE IN FRANCEZA ";X$;" ?"
 .
 .
 .
```

Vor fi şapte perechi de cuvinte în test, astfel încît vom modifica pe N în 7.

Perechi de întrebări şi răspunsuri.

Dacă mai adăugaţi perechi de întrebări şi răspunsuri pînă la linia 140, nu uitaţi să modificaţi valoarea lui N din linia 40.

Atenţie! Valoarea lui **N** *din linia* **40** *a programului trebuie să fie egală cu numărul de linii DATA.*



ALTE IDEI ŞI SUGESTII

Iată unele sugestii şi idei noi pentru jocuri cu întrebări şi răspunsuri:

1. Puteţi face testul oricît de lung doriţi (cu mai multe întrebări) adăugînd mai multe linii DATA.
2. Puteţi alcătui teste pentru orice domeniu: biologie, istorie, cunoştinţe generale (vezi figurile 31 şi 32).

Fig. 31

3. Puteți să folosiți în test atît cuvinte, cît și numere, de exemplu, în testul cu date istorice.

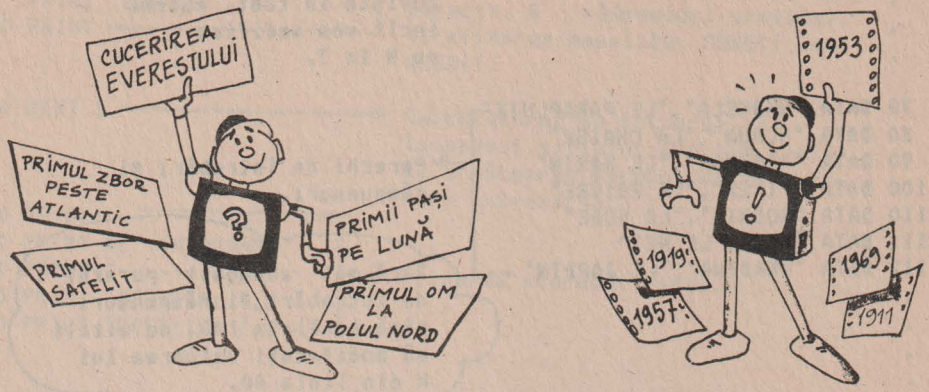


Fig. 32

4. Calculatorul va accepta un singur răspuns la o întrebare (există un singur răspuns corect la o întrebare).

5. Adăugați următoarele linii programelor **"Care este capitala?"** și **"Test de franceză"**.

```
300 IF C=N THEN PRINT "PERFECT"
310 IF C=0 THEN PRINT "SLAB"
320 IF C=6 THEN PRINT "JUMATATE
    CORECT"
330 IF C>6 AND C<N THEN PRINT
    "DESTUL DE BINE"
```

Aceste noi linii vor da jucătorului o imagine asupra scorului general obținut. Puteți modifica cum doriți mesajele dintre ghilimele. De exemplu: la linia 310 se poate pune mesajul "MERGI LA PLIMBARE" sau "MAI INVAȚA".

9. *De ce calculatorul nu va scrie niciodată ce i-ați indicat la linia* **320**?

10. *Puteți adăuga programelor* **„Care este capitala?"** *și* **„Test de franceză"** *o linie, astfel încît la un răspuns greșit înainte de o nouă întrebare, să se indice care ar fi fost răspunsul corect?*

Răspunsurile la pagina 139.

TABLA ÎNMULȚIRII

Iată un joc care testează tabla înmulțirii, așteptînd, de fiecare dată, să se introducă rezultatul pentru o operație de înmulțire dintre două numere întregi aleatoare cuprinse între 0 și 10:

```
 5 REM ** TABLA INMULTIRII **
   ************************
10 LET A=INT (RND*(10+1))
20 LET B=INT (RND*(10+1))
30 INPUT ("CIT FACE ";A;"X";B;"?");C
40 IF C=A*B THEN PRINT "AI
   RASPUNS BINE": GO TO 10
50 PRINT "RAU, MAI INCEARCA":
   GO TO 30
```

→ Se generează un număr oarecare cuprins între 0 și 10.

→ Se generează alt număr oarecare între 0 și 10.

Parantezele dintr-un INPUT determină afișarea atît a mesajelor cît și a variabilelor din interiorul lor !

→ Dacă răspunsul a fost corect, se afișează mesajul respectiv și se generează alte numere (altă operație).

→ Dacă răspunsul a fost greșit, se afișează mesajul respectiv și apoi se încearcă alt răspuns.

11. *Modificați jocul* „**Tabla înmulțirii**“, *astfel încît la fiecare operație să se afișeze numărul de operații, precum și numărul de răspunsuri corecte și incorecte.*

Răspunsul la pagina 139.

CALCULATORUL POET

Puteți programa calculatorul să aleagă cuvinte dintr-o listă memorată în instrucțiuni **DATA** și, apoi, să le pună împreună în linii (versuri), astfel încît să formeze o poezie. Calculatorul va alege cuvintele la întîmplare, astfel încît, de obicei, cîteva linii vor reprezenta versuri bune, printre altele care nu au nici un înțeles. Aceasta depinde însă și de modul în care sînt alese cuvintele inițiale, fiind bine ca acestea să fie grupate după mai multe funcții gramaticale ca: substantive, adjective, verbe, adverbe, prepoziții, etc. Puteți să vă distrați modificînd cuvintele din instrucțiunile **DATA** și să încercați astfel să realizați poezii mai interesante sau poezii care să trateze anumite subiecte.

```
10 REM ** POEZIE CU CALCULATORUL **
       ****************************

20 DATA "LACUL","PADUREA","IAZUL",
   "CRINGUL","COPACUL"
30 DATA "ALB","GALBEN","NEGRU",
   "INVERZIT","RUGINIU"
40 DATA "SI","AL","ALE","IL","CU",
   "DE"
50 DATA "CRENGI","FLORI","NUFERI",
   "ZAPADA"
60 DATA "SCUTURA","INCARCA","ADIE",
   "PLUTESTE","TRESARE"
65 DATA "ACOPERA","CODRILOR","MUNTILOR",
   "COVORUL","CIMPUL"

70 DIM C$(30,8)
80 REM CITIREA DATELOR

90 FOR I=1 TO 30
100 READ C$(I)

110 NEXT I
120 REM CREAREA POEZIEI
130 FOR I=1 TO 3

140 FOR J=1 TO 4

150 FOR K=1 TO INT (RND*3+2)

160 LET R=INT (RND*30+1)

170 PRINT C$(R);"  ";
180 NEXT K
190 PRINT
200 NEXT J
210 PRINT
220 PRINT
230 NEXT I
240 STOP
```

Puteți modifica cuvintele din program și veți realiza alte poezii!

30 de cuvinte care vor fi utilizate de calculator în poezie.

Rezervarea unui spațiu de memorie suficient pentru 30 de cuvinte de cîte maxim 8 litere.

Cuvintele vor fi memorate în variabila C$.

Ciclu pentru numărul de strofe.

Ciclu pentru numărul de versuri în strofă.

K reprezintă din cîte cuvinte va fi format versul (între 2 și 4).

Calculatorul alege un număr R între 1 și 30 și scrie cuvîntul corespunzător numărului R în această poziție în C$.

În programul prezentat, cuvintele din liniile **DATA** au fost alese pornindu-se de la lirica eminesciană:

Lacul codrilor albastru
Nuferi galbeni îl încarcă . . .

În fig. 33, se poate observa cum a reuşit calculatorul să realizeze, în replică, un frumos tablou de iarnă.

Fig. 33

CALCULUL MEDIILOR ŞI SORTAREA DATELOR

Calculatorul este de mare ajutor în realizarea rapidă a calculelor ca şi în organizarea datelor şi a altor informaţii. Programul următor va calcula media, precum şi cel mai mare şi cel mai mic dintre numerele date într-o listă de numere.

```
10 REM ** CALCUL MEDIE **
       ********************
20 PRINT "CALCULUL MEDIEI"

30 DIM X(50)  ------------>  Puteţi introduce pînă la 50 de
                             numere care vor fi memorate în
                             variabila de tip tablou X.
```

```
 40 REM INTRODUCEREA DATELOR
 50 LET N=0 ──────────► Variabila N ține evidența
                        a cîte numere s-au introdus.
 60 PRINT "SE POT INTRODUCE"
 70 PRINT "PINA LA 50 DE NUMERE"
 80 PRINT
 90 PRINT "TASTEAZA 999 LA SFIRSITUL" ──────► 999 este un cod
100 PRINT "LISTEI DE NUMERE"          care atunci cînd se tastează, va
                                      indica faptul că ați terminat
                                      introducerea numerelor.Utilizați
                                      un cod de număr care nu face
                                      parte din lista de numere,altfel
                                      calculatorul va crede că ați
                                      terminat lista.

120 LET N=N+1
130 IF N>50 THEN GO TO 180 ──► Calculatorul nu va accepta mai
                               mult de 50 de numere.

140 PRINT "TASTEAZA NUMARUL AL ";N;"-LEA"
150 INPUT X(N)
160 IF X(N)=999 THEN GO TO 180 ──► Dacă tastați 999, calcu-
                               latorul merge în linia 180 și
                               începe calculele.

170 GO TO 120
180 REM CALCULE PENTRU MEDIE
190 LET N=N-1
200 PRINT "AI INTRODUS ";N;" NUMERE"
210 PRINT
220 LET T=0 ──────┐ ──► Ciclu pentru calculul totalului
230 FOR I=1 TO N  │     (sumei) numerelor introduse.
240 LET T=T+X(I)  │
250 NEXT I ───────┘
260 PRINT "TOTALUL ESTE ";T ──► Pentru a calcula media (A), calcu-
270 LET A=T/N               latorul va împărți totalul T la
                            numărul N

280 PRINT "MEDIA ESTE ";A
290 REM CALCUL MAX SI MIN
300 LET MAX=X(1) ──────┐ ──► La început primul număr este consi-
310 LET MIN=X(1)       │     derat, în același timp, și cel mai
320 FOR I=2 TO N       │     mare și cel mai mic număr din listă.
330 IF X(I)>MAX THEN   │     Apoi, calculatorul compară toate
    LET MAX=X(I)       │     numerele pentru a-l găsi pe cel mai
340 IF X(I)<MIN THEN   │     mic și pe cel mai mare.
    LET MIN=X(I)       │
350 NEXT I ────────────┘
360 PRINT "CEL MAI MARE NUMAR ESTE ";MAX
370 PRINT "CEL MAI MIC NUMAR ESTE ";MIN
380 PRINT
390 GO TO 30
```

12. *Pentru utilizarea programului este bine, uneori, ca rezultatele să se exprime numai cu două zecimale. De exemplu: pentru calcularea mediilor şcolare. Aveţi vreo idee cum trebuie modificat programul pentru a realiza acest lucru?*

Răspunsul la pagina 139.

Încercaţi să calculaţi cu ajutorul programului „Calcul medie" următoarele medii:

- **media obţinută la o materie pe un trimestru;**
- **media generală pe an;**
- **media notelor tuturor elevilor din clasă (media clasei);**
- **vîrsta medie din familie;**
- **masa medie (în kg.) din familie:**
- **înălţimea medie a băieţilor din clasă;**
- **înălţimea medie a fetelor din clasă;**
- **înălţimea medie a elevilor din clasă;**
- **media timpului petrecut în faţa ecranului TV pe zi.**

Dacă aveţi la bicicletă un indicator pentru kilometraj, puteţi calcula distanţa parcursă în interval de o săptămînă (sau o lună) şi împărţind această distanţă la perioada de timp, adică la numărul de zile (sau săptămîni), puteţi afla ce distanţă aţi parcurs în medie pe zi sau pe săptămînă (vezi fig. 34).

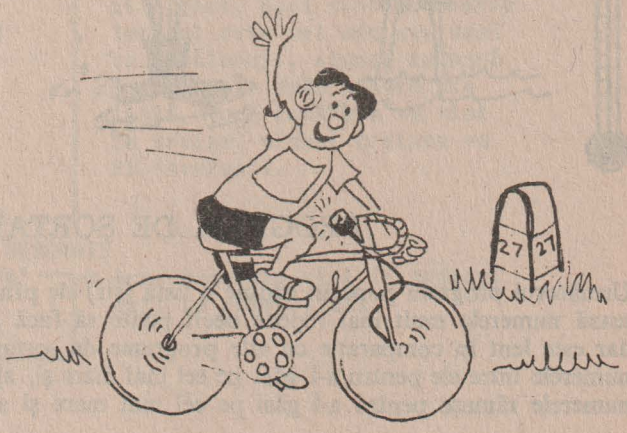


Fig. 34

METEOROLOGIE

Dacă sînteţi interesaţi în meteorologie, puteţi afla cu ajutorul calculatorului temperatura medie a aerului şi căderile medii de apă pe o perioadă (lună, săptămînă, an) (vezi fig. 35). Pentru aceasta aveţi nevoie de un termometru şi un cilindru gradat cu ajutorul cărora veţi înregistra zilnic datele respective.

La sfîrşitul perioadei, folosind programul de medie, veţi obţine valorile medii, precum şi valorile minime şi maxime ale aerului şi căderile de apă din perioada respectivă.

De asemenea, folosind un program de sortare, ca acela care va fi descris în continuare, veţi putea obţine o listă a valorilor, în care acestea vor fi afişate în ordine crescătoare.

Dacă aveţi o rudă sau un prieten cu preocupări asemănătoare, dar care locuieşte într-o localitate la o distanţă mare, îi puteţi cere să facă înregistrări pe care le le puteţi compara cu ale voastre.

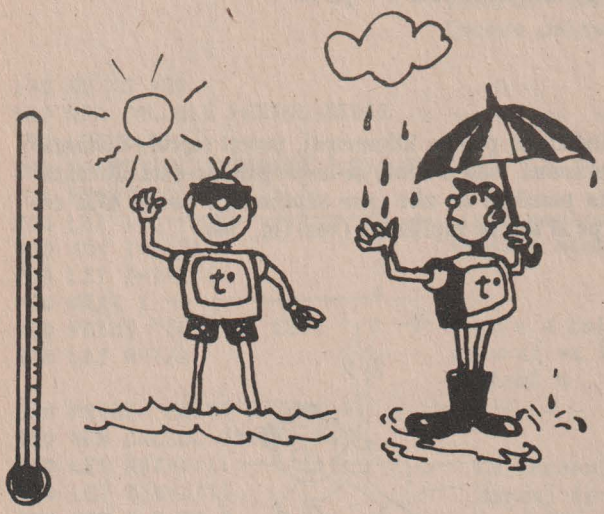


Fig. 35

PROGRAM DE SORTARE

Următorul program pune în ordine o listă (şir) de pînă la 20 de numere. El sortează numerele mult mai repede decît poate să facă acest lucru orice persoană, dar este lent în comparaţie cu alte programe de sortare, deoarece compară toate numerele între ele pentru a-l găsi pe cel mai mare şi, apoi, compară din nou toate numerele rămase pentru a-l găsi pe cel mai mare şi aşa mai departe.

```
10 REM ** PROGRAM DE SORTARE **
       ************************
20 DIM A(20)
30 PRINT "PROGRAMUL PUNE IN ORDINE"
40 PRINT "PINA LA 20 DE NUMERE"
50 PRINT
60 REM INTRODUCERE NUMERE
70 GO TO 90
80 PRINT "TREBUIE SA FIE
   INTRE 2 SI 20"
90 PRINT "CITE NUMERE DORITI
   SA PUNETI IN ORDINE?"
100 INPUT N
110 IF N<2 OR N>20 THEN GO TO 80
120 PRINT "INTRODUCETI CITE
   UN NUMAR ODATA"
130 FOR I=1 TO N
140 PRINT "CARE ESTE NUMARUL?
150 INPUT A(I)
160 NEXT I
170 REM AFISAREA LISTEI
   NEORDONATA
175 PRINT "LISTA NEORDONATA"
180 FOR I=1 TO N
190 PRINT A(I);" ";
200 NEXT I
210 PRINT
220 REM ORDONAREA NUMERELOR
230 FOR I=1 TO N-1
240 FOR J=1 TO N-I
250 IF A(J)<=A(J+1) THEN
   GO TO 300
260 LET B=A(J)
270 LET C=A(J+1)
280 LET A(J)=C
290 LET A(J+1)=B
300 NEXT J
310 NEXT I
320 REM AFISAREA LISTEI ORDONATE
325 PRINT "LISTA ORDONATA"
330 FOR I=1 TO N
340 PRINT A(I);" ";
350 NEXT I
360 PRINT
370 STOP
```

Puteți modifica numărul 20 dacă doriți să sortați mai multe numere.

Calculatorul întreabă cîte numere sînt în listă și memorează răspunsul în variabila N.

Numerele sînt puse în variabila de tip tablou A în ordinea în care le testați.

Numerele sînt afișate în ordinea în care au fost introduse, unul după altul (cu un spațiu între ele).

Calculatorul compară numerele din listă între ele, două cîte două. Dacă sînt în ordine (primul este mai mic sau egal cu următorul), atunci compară următoarele două numere din listă. Dacă numerele nu sînt în ordine, atunci ordinea va fi inversată.

Acum numerele sînt în ordine, deci le putem lista.

Nu ați înțeles care este mecanismul cu care se realizează sortarea numerelor?

Să luăm, în acest caz, pentru exemplificare o listă formată din următoarele elemente:

2 6 4 1 5 3

Să comparăm numerele între ele pe perechi: primul cu al 2-lea; al 2-lea cu al 3-lea etc. și să le schimbăm ordinea atunci cînd al doilea este mai mic ca primul. Vom obține următoarea listă:

2 4 1 5 3 6

Ce observați?

In primul rînd că lista nu este încă ordonată, deci va trebui să reluăm procedeul de la început, iar în al doilea rînd, numărul cel mai mare (MAX = 6) se găsește la coada listei. Deci, la reluarea procedeului va fi suficient să comparăm elementele listei fără să mai luăm în evidență și ultimul număr din listă. La a 2-a trecere vom obține următoarea listă:

2 1 4 3 5 6

Observați că nici acum lista nu este încă ordonată, dar la reluarea procedeului, va fi suficient să comparăm elementele listei fără ultimele două.

Acum se clarifică de ce indicele J din ciclul din linia 240 ia valoarea pînă la (N-I)? Da! deoarece la fiecare I-a trecere prin șir vom lua în considerație, pentru comparare, termenii din listă pînă la al (N-I)-lea (după acesta termenii sînt în ordine). După încă o trecere prin listă, vom obține lista ordonată:

1 2 3 4 5 6

Acum devine foarte clar că primul ciclu (**230—310**) este necesar pentru a realiza N — 1 treceri prin listă, iar al 2-lea ciclu (**240—300**) este necesar pentru a realiza comparările dintre elementele listei în cadrul unei treceri. Iată de ce și algoritmul este destul de lent: am avut nevoie doar de 3 treceri prin listă pentru a ordona cele 6 elemente, pe cînd calculatorul va mai efectua încă două treceri (N — 1 = 5) pentru a afișa lista ordonată, cu toate că, în aceste două treceri, nu va mai efectua nici o modificare.

13. *Aveți vreo idee cum puteți modifica* „**Programul de sortare**“, *astfel încît calculatorul să nu mai efectueze treceri prin listă dacă lista este deja ordonată?*

Răspunsul la pagina 139.

PROGRAM DE CĂUTARE

În multe jocuri se pune problema **găsirii** unui element dintr-o listă.

Vă mai aduceți aminte de jocul "**Ghicește numărul**"?

Acesta este, de fapt, un joc în care trebuie căutat și găsit un element dintr-o listă de elemente. Probabil, ați jucat jocul de mai multe ori și, pînă la urmă, de fiecare dată ați ghicit numărul. Dar, v-ați pus problema aflării unei metode (algoritm) de găsire rapidă și sigură a numărului?

Să presupunem că, acum, calculatorul este cel care trebuie să ghicească un număr. Va trebui să-l "învățăm" cum să-l găsească printr-o metodă rapidă. Sigur, el are la îndemînă și metoda primitivă, bazată pe faptul că poate să realizeze foarte rapid operațiile de comparare, adică să ia numerele la rînd și să le compare cu cel care trebuie ghicit. În acest mod, el va identifica pînă la urmă numărul căutat, dar numărul de operații pe care le va efectua va fi, de obicei, mare și va depinde de întîmplare. Acest mod de căutare se mai numește *lineară* și se aplică, de obicei, la listele nesortate (neordonate).

Iată o altă metodă de căutare, care se numește *binară*, și care, atenție!, nu se poate aplica decît la listele care, în prealabil, au fost *sortate*: ideea de bază este de a se compara numărul care trebuie găsit cu elementul din mijlocul listei ordonate. Elementul care trebuie găsit va putea fi ori mai mic, ori mai mare decît elementul din mijloc, dacă nu este chiar egal cu el, în care caz căutarea este terminată. În primul caz, vom ști că elementul se găsește în prima jumătate a listei, iar în al doilea, în a doua jumătate a listei. Procesul se repetă, de fiecare dată, lista împărțindu-se în două.

Iată un exemplu: să considerăm căutarea elementului 30 din următoarea listă ordonată:

1 2 4 6 8 10 12 14 16 18 20 24 28 30 36

Mai întîi alegem 14 (al 8-lea element este mijlocul listei avînd în fiecare parte cîte 7 elemente). Deoarece elementul căutat este mai mare, vom ști că el se află în lista: 16 18 20 24 28 30 36 și alegem, deci 24 (mijlocul listei). Acum lista va fi 28 30 36 și alegem 30 (mijlocul), aflînd astfel din 3 încercări elementul căutat.

Iată și programul cu ajutorul căruia se poate găsi un element dintr-o listă ordonată de 50 de elemente:

```
 10 REM ** PROGRAM DE CAUTARE **
        ************************
 20 DIM A(50)
 30 PRINT "CITE NUMERE IN LISTA?"
 40 INPUT N
 50 REM GENERAREA ELEMENTELOR DIN LISTA
 60 FOR I=1 TO N ------------------------> Se generează maximum 50 de
 70 LET A(I)=INT (100*RND)+100             numere aleatoare, cuprinse
 80 FOR R=1 TO I-1                         între 100 și 199.

 90 IF A(I)=A(R) THEN GO TO 70 ---------> Numerele din listă nu trebuie
100 NEXT R                                să se repete.
110 NEXT I ------------------------------>
```

```
120 REM AFISARE LISTA NEORDONATA
125 PRINT "LISTA NEORDONATA"
130 FOR I=1 TO N
140 PRINT A(I);" ";
150 NEXT I

160 PRINT
170 REM SORTARE
180 FOR I=1 TO N-1
190 FOR J=1 TO N-I
200 IF A(J)<=A(J+1) THEN GO TO 250
210 LET B=A(J)
220 LET C=A(J+1)
230 LET A(J)=C
240 LET A(J+1)=B
250 NEXT J
260 NEXT I
270 REM AFISARE LISTA SORTATA
275 PRINT "LISTA ORDONATA"
280 FOR I=1 TO N
290 PRINT A(I);" ";
300 NEXT I
310 PRINT

320 PRINT "TASTATI NUMARUL DE CAUTAT"
330 PRINT "INTRE 100 SI 199"
340 INPUT X
350 PRINT
360 REM CAUTARE
370 LET L=1

380 LET H=N

390 LET C=0

400 LET M=INT ((H+L)/2)

410 LET C=C+1
420 IF X=A(M) THEN GO TO 490
430 IF L>=H THEN GO TO 520
440 IF X>A(M) THEN GO TO 470

450 LET H=M-1
460 GO TO 400
470 LET L=M+1
480 GO TO 400
```

Afișare elemente lista nesortată.

Sortare elemente.

Afișare elemente lista sortată.

Numărul care va fi căutat este memorat în variabila X.

Primul element din listă.

Ultimul element din listă.

Numărul de încercari (inițial 0)

O metodă de a se împărți lista în două părți și găsirea elementului din mijloc (M).

Numărul de încercări crește cu o unitate.

Dacă numărul căutat este mai mare decît cel din mijloc, atunci se va cerceta lista din partea dreaptă (L=M+1); dacă nu, se va cerceta lista din partea stîngă (H=M-1).

```
490 PRINT "NUMAR GASIT ";X
500 PRINT "DIN ";C;" INCERCARI"
510 GO TO 320
520 PRINT "NUMARUL NU A FOST GASIT"
530 PRINT "DUPA ";C;" INCERCARI"
540 GO TO 320
```

CIFRAREA MESAJELOR ŞI DEZLEGAREA ENIGMELOR

În timpul celui de-al II-lea război mondial, un calculator electronic a fost folosit pentru descifrarea mesajelor. Cu ajutorul calculatorului se poate lesne realiza codificarea (cifrarea) mesajelor, deoarece în interiorul calculatorului toate caracterele de pe tastatură (litere, cifre, semne) sînt convertite (transformate)în numere. Astfel, fiecărui caracter îi corespunde un cod numeric conform unei codificări adoptate de multe ţări, numit ASCII *. În zilele noastre, majoritatea calculatoarelor utilizează codul ASCII pentru codificarea caracterelor. Puteţi obţine pe ecran codurile ASCII rulînd programul "Codul ASCII". Veţi observa că sînt codificate chiar şi caracterele grafice şi cuvintele pe care le folosiţi în programe (instrucţiunile, comenzile şi funcţiile).

Însă, veţi observa că nu sînt codificate literele Ş, Ţ, Ă, Î, ele neregăsindu-se nici pe tastatură. Se vor utiliza în scopul codificării numai literele din alfabetul englez (26 de litere). Puteţi să obţineţi caracterul corespunzător unui anumit număr (cod) folosind într-un program funcţia CHR$ (prescurtarea de la CHARACTER). Iată programul care, ca rezultat, va afişa în dreptul fiecărui număr caracterul pe care îl codifică.

Pînă la numărul 32 codurile reprezintă caractere neafişabile. Se începe cu numărul 32, care reprezintă codificarea spaţiului gol (SPACE) şi se termină cu 143, care reprezintă codificarea unui caracter grafic aflat pe tastatură (■):

```
10 REM ** CODUL ASCII **
       ******************
20 FOR I=32 TO 143
30 PRINT I;" ";CHR$ I
40 NEXT I
```

Cînd ecranul se umple cu caractere şi coduri şi apare mesajul „scroll?", va trebui să acţionaţi o tastă, oricare doriţi (în afară de N sau BREAK) pentru a obţine şi restul de coduri.

Cuvîntul CODE pus înaintea unui caracter realizează funcţia inversă, rezultînd codul ASCII al caracterului.

Astfel, puteţi realiza mesaje codificate în numere şi puteţi trimite unui prieten un astfel de mesaj printr-un program pe o casetă magnetică.

---

* American Standard Code for Information Interchange (Codul standard american pentru transmiterea de informaţii).

Următorul program convertește (codifică) un mesaj literă cu literă în numere **ASCII** și, apoi, adună un număr de cod, **C**, pentru a face codul mai greu de descifrat:

```
10 REM ** CODIFICATOR MESAJE **
      ************************
20 LET C=27
30 PRINT "TASTEAZA PRIMA LITERA"
40 INPUT X$ ------------------------> Se introduce fiecare
                                      literă a mesajului.
50 LET X=CODE (X$)+C
60 PRINT X ------------------------> Pe ecran se va afișa
                                      codul acestei litere.
70 PRINT "TASTEAZA URMATOAREA LITERA"
80 GO TO 30
```

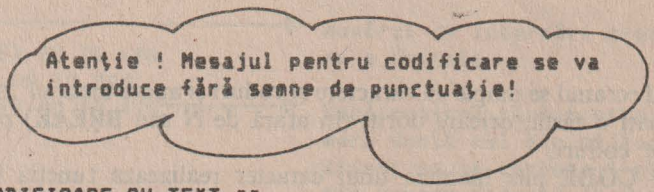
14. *Puteți să scrieți programul pe care trebuie să îl dați prietenului căruia îi transmiteți mesaje cifrate pentru a decodifica mesajele pe care i le trimiteți?*

Răspunsul la pagina 140

Mesajele în care literele sînt codificate cu alte litere sînt mai interesante și mai dificil de decodificat.

Personajul din fig. 36 cifrează un mesaj înlocuind fiecare literă cu a cincea literă din alfabet, plasată după ea. Următorul program cifrează mesajele în același mod, găsind codul **ASCII** pentru fiecare literă și, apoi, adăugînd un număr și afișînd litera corespunzătoare codului **ASCII** astfel determinat:

Fig. 36




```
10 REM ** CODIFICARE CU TEXT **
      ************************
20 LET Z=CODE ("Z")
30 PRINT "ALEGETI UN NUMAR PENTRU COD"
40 PRINT "INTRE 1 SI 25"
```

```
 50 INPUT S
 60 PRINT "TASTATI MESAJUL DE CODIFICAT"
 70 INPUT X$
 80 PRINT
 90 FOR I=1 TO LEN (X$)
100 LET Y$=X$(I TO I)
110 IF Y$<"A" OR Y$>"Z" THEN GO TO 170
120 LET X=CODE (Y$)
130 IF X+S<Z+1 THEN PRINT CHR$ (X+S);
140 IF X+S>Z THEN PRINT CHR$ (X+S-26);
150 NEXT I
160 STOP
170 PRINT Y$
180 GO TO 150
```

DESCIFRAREA TEXTELOR CODIFICATE

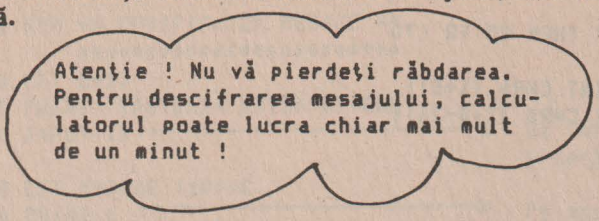
Persoana care primește mesajul pe care l-ați codificat poate folosi următorul program, fără a cunoaște codul numeric pe care l-ați ales. Sînteți curioși cum realizează acest lucru? Programul afișează toate cele 26 de versiuni posibile ale mesajului, iar voi veți fi capabili (sîntem siguri) să o recunoașteți pe cea corectă.

```
 10 .REM ** DECODIFICATOR MESAJE **
     **************************
 20 DIM Z$(26)
 30 PRINT "TASTATI MESAJUL CODIFICAT"
 40 INPUT X$
 50 LET L=LEN (X$)
 60 FOR K=1 TO 26
 70 FOR I=1 TO 26
 80 IF I<K THEN LET Z$(I)=CHR$ (CODE ("A")+I-K+26)
 90 IF I>=K THEN LET Z$(I)=CHR$ (CODE ("A")+I-K)
100 NEXT I
110 FOR J=1 TO L
120 LET A$=X$(J TO J)
130 IF A$=" " THEN PRINT A$;
140 IF A$=" " THEN GO TO 160
150 PRINT Z$(CODE (A$)-CODE ("A")+1);
160 NEXT J
170 PRINT
180 NEXT K
```

15. *Încercați să descifrați singuri următorul mesaj și, apoi, observați ce rapid realizează acest lucru calculatorul, folosind programul* **„Decodificator de mesaje"**: *TYS JSEK VES CDOPKX CDYZ KPVK NKMK KBO EX TYM NO CKR CS EX MKVMEVKDYB CDYZ*
Ce rezultat ați obținut la descifrarea mesajului?

Răspunsul la pagina 140

Iată cîteva linii suplimentare şi cîteva modificări pe care le puteţi adăuga programului **"Decodificator de mesaje"**, prin care acesta va deveni „mai deştept": acum programul poate recunoaşte anumite cuvinte trecînd prin cele 26 de versiuni posibile ale mesajului, iar dacă acest lucru se va întîmpla (calculatorul recunoaşte un cuvînt), atunci se va afişa versiunea curentă a mesajului, care va fi şi cea corect decodificată.




```
15 DIM C$(100)
```

Mesajul codificat nu va fi mai mare de 100 de caractere.

```
130 IF A$=" " THEN LET C$(J)=A$
150 LET C$(J)=Z$(CODE (A$)-CODE ("A")+1)
170 FOR J=1 TO L-2
180 LET R$=C$(J)+C$(J+1)
```

Dacă vă hotărîţi să alegeţi pentru decodificare cuvinte cheie de 3 litere (de exemplu LUI, DAR, ARE), atunci linia 180 va arăta astfel: LET R$=C$(J)+ C$(J+1)+ C$(J+2)

```
190 IF R$="UN" THEN GO TO 280
200 IF R$="DE" THEN TO TO 280
210 IF R$="SI" THEN GO TO 280
220 IF R$="LA" THEN GO TO 280
```

Puteţi pune pe liniile 190-220 propriile cuvinte formate din două litere care credeţi că au şanse mari să apară într-un mesaj codificat. Toate aceste linii se pot înlocui cu una singură:
IF R$="UN" OR R$="DE" OR R$="SI" OR R$="LA" ...
THEN GO TO 280.

```
230 NEXT J
240 NEXT K
250 PRINT "NU POATE FI DESCIFRAT"
260 PRINT
270 GO TO 30
280 PRINT "MESAJ DESCIFRAT"
290 FOR J=1 TO L
300 PRINT C$(J);
310 NEXT J
320 PRINT
330 PRINT "GATA PENTRU ALT MESAJ"
340 PRINT
350 GO TO 30
```

16. *Eroii din romanul „Mathias Sandorf", scris de Jules Verne, trimiteau mesaje cifrate prin intermediul porumbeilor voiajori. Mesajele erau scrise pe bilețele sub formă de grupe de litere, care formau un pătrat format din 36 de pătrățele (deci, 6 pe o latură), în fiecare pătrat putînd intra o literă. Ele nu puteau fi descifrate decît cu ajutorul unui grătar special, de dimensiunea pătratului respectiv, decupat în interior, în așa fel încît, punîndu-se deasupra unei grupe de litere, unele litere să fie acoperite, iar altele nu (vezi fig. 37)*

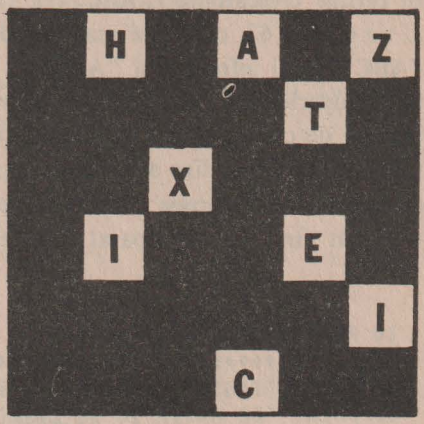


Fig. 37

Cu toate că reușiseră să fure grătarul și să-l pună peste text, banditul Sarcany și bancherul Silas Toronthal (personajele negative ale cărții), n-au putut să descifreze mesajul, deoarece eroii mai folosiseră un ultim truc la cifrarea mesajului.

Știți care era acel truc și cum trebuie modificat programul **„Decodificator de mesaje"** *pentru ca textul să fie descifrat?*

Răspunsul la pagina 140.

CLASAMENT ALFABETIC

Am văzut că memorarea cuvintelor de către calculator se face prin codificarea literelor în codul **ASCII**. Dacă literele se codifică numeric și cunoaștem și modul în care se pot pune în ordine numerele (algoritmul de sortare), înseamnă că putem pune în ordine și cuvintele, adică să facem un clasament alfabetic. Acesta se realizează în felul următor: **compararea** a două șiruri de caractere, de exemplu: două cuvinte, este realizată caracter cu caracter. Atunci cînd caracterul cu care începe primul cuvînt este o literă situată în alfabet înaintea litere i cu care începe

al doilea cuvînt, primul cuvînt va fi considerat inferior (mai mic) decît cel de-al doilea Atunci cînd amîndouă cuvintele încep cu aceeaşi literă, compararea se continuă asupra următoarelor două caractere, şi aşa mai departe, pînă este întîlnită o diferenţă. Dacă diferenţa constă numai în faptul că un cuvînt este mai scurt decît celălalt, atunci primul se va considera inferior.

Astfel, va fi posibil de realizat un clasament a două (sau mai multe) cuvinte sau grupe de cuvinte (spaţiul gol contează) în sens alfabetic. Clasamentul se va face în ordine crescătoare: "**ABC**" va fi inferior lui "**ABD**", deoarece codul **ASCII** al lui **C** este 67, care este mai mic decît 68, codul **ASCII** al lui **D**. De asemenea "**ABC**" va fi inferior lui "**ABCD**", deoarece este mai scurt. Bineînţeles, "**CAB**" va fi superior lui "**BACD**" din cauza primei litere. Dacă sînt mai mult de două cuvinte de clasat, o metodă simplă de aplicat constă în căutarea, mai întîi, a primului cuvînt în ordine alfabetică. Apoi, se reia procedeul fără să se mai ţină seama de primul cuvînt, care acum este în ordine corectă. Clasamentul alfabetic va fi terminat cînd s-au comparat între ele ultimele două cuvinte.

```
10 REM ** CLASAMENT ALFABETIC **
       ************************
20 PRINT "TASTATI NUMARUL DE CUVINTE"
30 PRINT "DE MAXIM 20 DE CARACTERE"
40 INPUT NC ------------------------> Numărul de cuvinte.

50 DIM N$(NC,20) -------------------> Rezervarea spaţiului de
                                      memorie pentru fişierul
                                      de cuvinte.
60 REM INTRODUCERE CUVINTE
70 PRINT "INTRODUCETI CUVINTUL" ----> Introducere cuvinte.
80 FOR I=1 TO NC
90 INPUT N$(I)
100 NEXT I
110 REM AFISARE FISIER NEORDONAT ---> Afişarea cuvintelor în
    DE CUVINTE                        ordinea în care au fost
120 FOR I=1 TO NC                     introduse.
130 PRINT N$(I)
140 NEXT I
150 PRINT
160 REM SORTARE SI AFISARE FISIER SORTAT
170 PRINT "FISIER CLASAT"
180 FOR I=1 TO NC-1
190 LET M$=N$(I) -------------------> Considerăm primul cuvînt
                                      ca cel mai mic.

200 LET POZ=I ----------------------> Indică poziţia celui mai
                                      mic cuvînt.
```

```
210 FOR J=I+1 TO NC
220 IF N$(J)>=M$ THEN GO TO 250
230 LET M$=N$(J)
240 LET POZ=J
250 NEXT J
260 PRINT N$(POZ)
270 LET N$(POZ)=N$(I)
280 LET N$(I)=M$
290 NEXT I
300 PRINT N$(NC)
310 STOP
```

Ciclul de căutare al celui mai mic cuvînt.

Afișarea cuvîntului bine clasat.

Schimbarea ordinii cuvintelor.

Afișarea ultimului cuvînt.

La rularea programului puteți obține, de exemplu, ceva asemănător cu fig. 38

```
pruna
gutuie
portocala
para
mar
strugure
cireasa
caisa
capsuna
fraga

FISIER CLASAT:
caisa
capsuna
cireasa
fraga
gutuie
mar
para
portocala
pruna
strugure
```

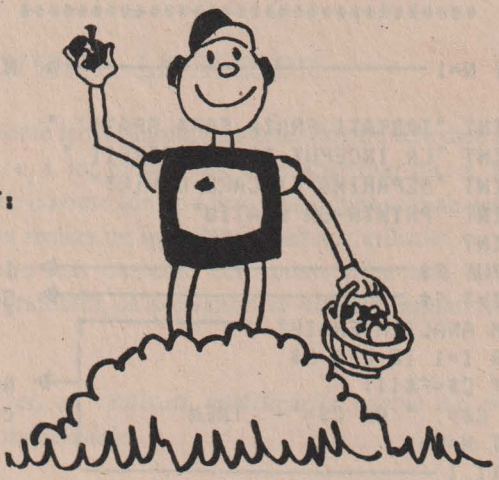


Fig. 38

La ce puteți folosi programul „**Clasament alfabetic**"?

Pentru o **agendă** cu nume de prieteni și de cunoștințe, cu adresele și numerele lor de telefon, programul „**Clasament alfabetic**" va aranja în ordine alfabetică numele (ca în cartea de telefon). Încercați să realizați o astfel de agendă și veți vedea că

fără ajutorul calculatorului, dacă aveţi peste 30 de prieteni şi cunoştinţe, aranjarea în ordine alfabetică vă va lua foarte mult timp. În acelaşi fel, puteţi să vă puneţi în ordine cărţile din bibliotecă (se va putea face o ordonare alfabetică atît după numele autorului, cît şi după titlurile cărţilor), precum şi să vă ţineţi în ordine casetele cu programe, realizînd un catalog al programelor de pe casetele magnetice.

STUDIUL FRAZEI. LINGVISTICĂ CU CALCULATORUL

Unul din exemplele de studiu al unui text este descompunerea fiecărei fraze în cuvinte. Programul următor permite **numărarea cuvintelor** dintr-o frază, cu condiţia separării fiecărui cuvînt printr-un spaţiu (şi numai unul), aşa cum se şi face de obicei în practică. O excepţie este luată în consideraţie: liniuţa de unire, deoarece aceasta separă practic două cuvinte. La introducerea textului nu trebuie plasat spaţiu decît după semnul de punctuaţie: dacă acesta ar avea de o parte şi de alta spaţiu, atunci ar fi considerat un cuvînt.

```
10 REM ** NUMARARE CUVINTE DIN FRAZA **
       **********************************

20 LET N=1 ------------------------------> N reprezintă numărul de
                                            cuvinte din frază.
30 PRINT "TASTATI FRAZA FARA SPATII "
40 PRINT "LA INCEPUT SI LA SFIRSIT "
50 PRINT "SEPARIND FIECARE CUVINT"
60 PRINT "PRINTR-UN SPATIU"
70 PRINT
80 INPUT F$ ------------------------------> Introducerea frazei.
90 PRINT F$ ------------------------------> Scrierea frazei.
100 REM ANALIZA FRAZEI ---------------------
110 FOR I=1 TO LEN F$                      |
120 LET C$=F$(I)                           |-> Analiza frazei caracter
130 IF C$=" " OR C$="-" THEN               |   cu caracter.
    LET N=N+1                              |
140 NEXT I ---------------------------------
150 PRINT
160 PRINT "FRAZA ARE ";N;" CUVINTE"
170 PRINT
180 GO TO 20
```

Exemplu de utilizare (vezi fig. 39):

```
TASTATI FARA SPATII
LA INCEPUT SI LA SFIRSIT
SEPARIND FIECARE CUVINT
PRINTR-UN SPATIU
Nu stiu altii cum sint, dar eu,
cind ma gindesc la locul nasterii
mele, la casa parinteasca din
Humulesti, la stilpul hornului unde
lega mama o sfoara cu motocei
la capat, de crapau mitele jucindu-se
cu ei, la prichiciul vetrei cel
humuit, de care ma tineam,
cind incepusem a merge copacel,
la cuptiorul pe care ma ascundeam,
cind ne jucam noi, baietii,
de-a mijoarca, si la alte jocuri
si jucarii pline de hazul si farmecul
copilaresc, parca-mi salta si
acum inima de bucurie !

FRAZA ARE 86 CUVINTE
```

Fig. 39

FRECVENȚA LITERELOR

Literele alfabetului nu sînt toate la fel de mult folosite. Printre cele care apar mai des în texte sînt, de exemplu: **a**, **e**, **i**, iar printre cele care apar mai rar: **z**, **h**, **x**.

Cei care participă la jocuri de cuvinte (de ex.: Scrabble), cunosc bine acest lucru.

În programul următor se va realiza un inventar al literelor utilizate, cu numărul de apariții al acestora, pentru un text oarecare care poate avea mai multe linii. Pentru a simplifica scrierea programului, să convenim să utilizăm numai litere mici (minuscule).

17. *Rămîne ca exercițiu pentru voi, să realizați modificările pentru ca programul să trateze în același fel și majusculele.*

Răspunsul la pagina 140

```
10 REM ** FRECVENTA LITERELOR **
       ************************
20 DIM N(26) ──────────────────►  Rezervarea spațiului de
                                 memorie pentru variabila
                                 care reprezintă numărul de
                                 apariții a fiecărei litere.
```

```
 30 PRINT "INTRODUCETI TEXTUL CU MINUSCULE"
 40 INPUT F$ ------------------------> Introducerea textului.
 50 PRINT
 60 PRINT F$ ------------------------> Afişarea textului.
 70 PRINT
 80 REM NUMARAREA APARITIEI LITERELOR
 90 FOR I=1 TO 26
100 LET N(I)=0
110 LET M$=CHR$ (I+96) -------------> M$ contine cea de a I-a
                                      minusculă.

120 FOR J=1 TO LEN F$ ---------------> Ciclu de explorare a textu-
130 IF F$(J)=M$ THEN                  lui caracter cu caracter.
    LET N(I)=N(I)+1
140 NEXT J ---------------------------
150 PRINT M$;" ";N(I) ---------------> Afişarea rezultatului
160 NEXT I                            pentru fiecare literă.
170 STOP
```

Iată un exemplu de utilizare pentru un text în care cîteva litere (t, n) apar de mai multe ori (vezi fig. 40).

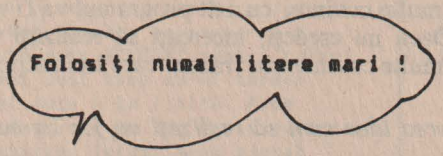
```
INTRODUCETI TEXTUL CU MINUSCULE
te rog tanti, fii atenta si prudenta
un moment!
a 4          b 0
c 0          d 1
e 4          f 1
g 1          h 0
i 4          j 0
k 0          j 0
m 2          n 5
o 2          p 1
q 0          r 2
s 1          t 7
u 2          v 0
w 0          x 0
y 0          z 0
```

Fig. 40

Încercaţi cu texte mai lungi pentru a obţine cele mai reprezentative rezultate pentru frecvenţa literelor din limba română. Ştiaţi că după ce s-au realizat mai multe teste de acest fel, pe baza rezultatelor obţinute (frecvenţele de apariţie ale literelor) s-a proiectat aranjarea clapelor (tastelor) la maşinile de scris şi la tastaturi, astfel încît literele cu cele mai mari frecvenţe de apariţie să fie cît mai apropiate de degetele cu care se tastează mai uşor?

CONVERSAŢII CU CALCULATORUL

În jocul următor, calculatorul va părea a fi dotat cu inteligenţă proprie, putînd face conversaţii (uneori inteligente) cu oricare din voi. Totuşi, din program se poate observa că el nu recunoaşte decît cuvintele DA şi NU, iar pentru restul de răspunsuri la mesajele tastate de interlocutor, el alege la întîmplare cuvintele dintr-o listă de date. Din această cauză, uneori, răspunsurile nu se potrivesc, părînd stranii.




```
10 REM ** CONVERSATIE **
       ******************
20 DATA "DE ACORD","DA","NU","SIGUR"
30 DATA "BINE","OH!","DACA SPUI TU!"
40 DATA "ESTE OARE ADEVARAT?",
   "MAI SPUNE O DATA"
50 DATA "TE DERANJEAZA?","AH!","HM!"
60 DATA "INTERESANT!","ITI DAI SEAMA?"
70 DIM R$(14,20)
80 DIM R(14)
90 LET C=0
100 REM MEMORARE REPLICI
110 FOR I=1 TO 14
120 READ R$(I)
130 LET R(I)=0
140 NEXT I
150 REM INCEPE CONVERSATIA
160 PRINT "SALUT!"
170 PRINT "CUM TE CHEAMA?"
180 INPUT A$
190 PRINT "IMI PARE BINE, ";A$
200 PRINT "VREI SA-MI SPUI CEVA?"
210 INPUT X$
220 IF X$="NU" THEN GO TO 370
230 FI X$="DA" THEN GO TO 390
240 LET R=INT (RND*14+1)
250 IF C=14 THEN GO TO 310
260 IF R(R)=1 THEN GO TO 240
270 LET R(R)=1
280 PRINT R$(R)
290 LET C=C+1
300 GO TO 210
310 FOR I=1 TO 14
320 LET R(I)=0
330 NEXT I
```

Cuvinte sau propoziţii care vor fi alese la întîmplare drept răspuns.

Rezervarea spaţiului de memorie pentru 14 cuvinte de maxim 20 caractere.

Memorarea şirurilor de caractere (a cuvintelor).

Inceput politicos de conversaţie.

Se introduce replica de către interlocutor.

Se alege drept răspuns unul din cele 14 şiruri de caractere.

```
340 PRINT "DE CE ZICI ";X$;"?"
350 LET C=0
360 GO TO 210
370 PRINT "DE CE NU?"
380 GO TO 210
390 PRINT "SIGUR?"
400 GO TO 210
```

Observați că programele care realizează conversații nu sînt ușor de făcut. Cel prezentat recunoaște doar două din numeroasele răspunsuri posibile. Cu cît va recunoaște mai multe cuvinte, cu atît programul va fi mai lung și va consuma mai multă memorie. Dacă nu credeți, încercați să realizați un astfel de program, care recunoaște mai multe cuvinte sau fraze și care în funcție de ele să dea anumite răspunsuri.

18. *Aveți vreo idee cum să realizați un joc cu care calculatorul să recunoască dacă persoana cu care conversează este de sex masculin sau feminin?*

Răspunsul la pagina 140.

Există un întreg domeniu al aplicațiilor calculatoarelor care se ocupă cu probleme de acest fel. Recunoașterea de cuvinte, fraze și forme, dicționare și traducerea automată, limbajul natural (posibilitatea de a ne putea adresa calculatorului în limbaj obișnuit), sinteza vocală (calculatorul care vorbește) sînt cîteva din aceste probleme. Rezolvarea lor ar face, într-adevăr, calculatoarele mult mai inteligente. De aceea, acest domeniu se mai numește al **inteligenței artificiale**.

CALCULATORUL VĂ AJUTĂ SĂ SCRIEȚI SCRISORI ȘI SĂ TRIMITEȚI FELICITĂRI

Programele cu care se pot realiza prelucrarea și editarea de texte se numesc **editoare de texte**. Ele vă permit să introduceți orice text în memoria calculatorului (prin tastare) și să-l corectați sau să îl modificați înainte de a-l tipări. Pentru tipărire aveți nevoie de o **imprimantă** (un echipament care se cuplează la calculator și care este înzestrat cu un dispozitiv de imprimare a caracterelor pe hîrtie, asemănător cu cel de la o mașină de scris. Vezi fig. 41), iar pentru memorarea pe un suport magnetic a textului veți folosi, bineînțeles, un casetofon.

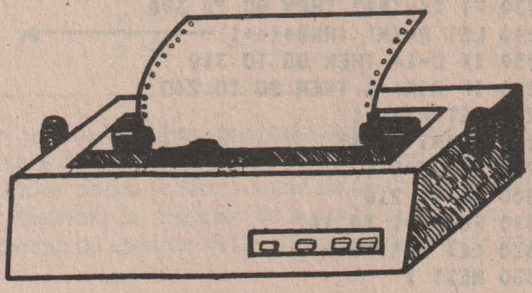


Fig. 41

Cu ajutorul programelor editoare de texte, puteți scrie ușor compuneri, scrisori, lecții sau articole. Permanent veți avea afișat pe ecran întreg textul (sau o pagină a lui, dacă este mai mare) și nu va trebui să vă concentrați la despărțirea cuvintelor în silabe în timpul introducerii textului (tastării), la sfîrșitul rîndului. Programul realizează automat o reordonare a cuvintelor, astfel încît acestea să nu fie despăr-

```
          Draga RECOOP
Sint un prieten al jocurilor pe care ni le pui la
dispozitie si pe care le practic cu placere. Mi-a
placut foarte mult jocul Scrabble (pe care il iau
cu mine si la scoala) precum si caseta cu jocuri
logce pentru calculator pe care am cumparat-o.
Intr-o am jucat foarte mult REVERSI contra calcu-
latorului si nu am mai avut timp sa-mi termin
lectiile luind astfel nota 6 la fizica. Acum
parintii nu mai imi dau voie sa joc decit dupa ce
imi termin toate lectiile. Totusi m-au ajutat
mult la matematica REBEC si GRAFICE iar in urma
aflarii raspunsurilor la unele intrebari cu
chichite din jocul MINITEHNICUS LA SCOALA, mi-am
indrptat notat la fizica luind un 10.
Pe cind o noua caseta cu jocuri ?
Vreau sa te anunt ca am facut si eu un joc pentru
calculator care te-ar putea interesa, mai ales
ca toti colegii mei se inghesuie sa-l joace. Se
numeste HOTUL. Acesta, adica hotul, fura ceva de
pe o strada si si apoi fuge. Jocul se desfasoara
pe o tabla caroiata. Politia incearca sa-l prinda
la cite o intersectie. De obicei il prinde dar
uneori mai si scapa. Jucatorul face parte din
echipajul Politiei. Ce zici?
Daca te intereseaza HOTUL si daca vrei sa imi
raspunzi gasesti adresa mea pe plic.
                    Sorin
```

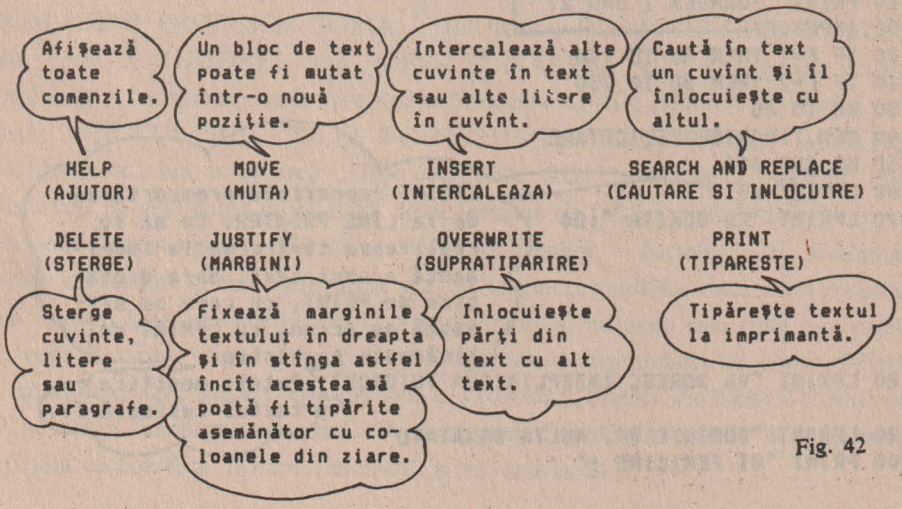


Fig. 42

ţite la sfîrşitul rîndului. Cursorul îl puteaţi aşeza oriunde în text, mişcîndu-l la dreapta, stînga, sus sau jos cu ajutorul săgeţilor.

Puteţi indica în orice moment calculatorului cum să aranjeze sau să pună textul în pagină, sau cît de lungi trebuie să fie rîndurile. De obicei, puteţi face toate aceste lucruri selectînd scurte comenzi, care apar pe ecran împreună cu textul, de obicei, în partea de jos a ecranului, sub forma unei liste.
Alte lucruri pe care le poate realiza un editor de texte le puteţi observa din figura nr. 42, care redă ecranul TV după introducerea unui text pentru o scrisoare. Pentru comenzi nu apar (de obicei) explicaţii pe ecran; pentru ele se poate consulta un manual de utilizare.

19. *V-au necăjit cele cîteva greşeli pe care le-aţi observat în text? Ei bine, nu este nici o problemă. Ele pot fi îndreptate repede înainte de a se tipări scrisoarea pentru a se expedia. Sînteţi ajutaţi (de data aceasta nu de către calculator) prin marcarea cu semnul * rîndurilor în care se găsesc greşeli. Puteţi indica, la fiecare, ce opţiuni (comenzi) trebuie acţionate?*

Răspunsul la pagina 140.

Iată un program care vă va ajuta să scrieţi felicitări în diverse ocazii, deşi, fiind destul de simplu, nu poate fi încă numit un editor de texte:

```
 10 REM ** FELICITARE **
      ****************
 20 PRINT "PE CINE FELICITI?"
 30 INPUT N$
 40 PRINT "CU CE OCAZIE?"
 50 INPUT O$
 60 PRINT "DIN PARTEA CUI?"
 70 INPUT P$
 80 PRINT "FORMULA 1 SAU 2?"
 90 INPUT F
100 IF F=1 THEN GO TO 140
110 IF F=2 THEN GO TO 250
130 GO TO 80
140 REM TIPARIRE FELICITARE
150 GO SUB 400
160 LPRINT N$
170 LPRINT "CU OCAZIA ";O$
180 LPRINT "VA DORESC INDEPLINIREA TUTUROR"
190 LPRINT "DORINTELOR, MULTA SANATATE"
200 PRINT "SI FERICIRE !"
```

Introducere date iniţiale.

LPRINT reprezintă prescurtarea de la LINE PRINTER. Cu ea se realizează tipărirea la imprimantă a unui text. Spre deosebire de PRINT, cu care se afişează pe ecran, cu LPRINT se tipăreşte pe hîrtie.

Puteţi modifica textul felicitării.

```
210 LPRINT ──────────►  Pentru a putea vedea felicitarea
                       pe ecran, puteţi să folosiţi la
                       început PRINT în loc de LPRINT.
220 LPRINT "                                ";P$
230 LPRINT " ****************************** "
240 GO TO 10
250 GO SUB 310
260 LPRINT "LUI ";N$                  ( Puteţi modifica textul
270 LPRINT "CU OCAZIA ";O$              felicitării. )
280 LPRINT "ITI DORESC NUMAI NOTE BUNE,"
290 LPRINT "SANATATE SI VOIE BUNA !"
300 GO TO 10
310 LPRINT
320 LPRINT "                                          ";P$
330 RETURN
400 REM SUBRUTINA GRAFICA
410 LPRINT "**********************************************"
420 LPRINT
430 LPRINT "*** *** *   *   ***  * *****  *    ***   ***"
440 LPRINT "*   *   *   *  *     *   *   * *  *  *  *  "
450 LPRINT "*** *** *   *  *     *   *   ****  ***   ***"
460 LPRINT "*   *   *   *  *     *   *   *  *  *  *  *  "
470 LPRINT "*   *** *** *   ***  *   *   *  *  *   * ***"
480 LPRINT
490 LPRINT "**********************************************"
550 RETURN
```

La cererea calculatorului de a introduce datele referitoare la persoana pe care o felicitaţi, veţi putea tasta:

DOMNULUI PROFESOR TOMA TUDOR sau DOAMNEI CHIRCA ELENA sau PASCU TUDOREL sau POPESCU MARINA, etc. La cererea calculatorului de a introduce datele referitoare la ocazia cu care trimiteţi felicitarea puteţi tasta: ANULUI NOU, ZILEI DE NAŞTERE, ZILEI FEMEII, MĂRŢIŞORULUI, etc., iar, la cererea „DIN PARTEA CUI?" veţi tasta propriul nume. Observaţi că veţi putea alege între două formule de felicitare (nu vă veţi adresa la fel domnilor profesori — formula 1 — şi prietenilor — formula 2). Bineînţeles, textul felicitărilor şi chiar subrutina grafică le puteţi modifica dacă aveţi o formulă mai reuşită. Dacă nu aveţi acces la o imprimantă şi nu aveţi nici rude nici prieteni pe care îi puteţi ruga să vă extragă (tipărească) la imprimantă cîteva felicitări în anumite ocazii, atunci veţi modifica în program **LPRINT** cu **PRINT** şi veţi copia de pe ecran felicitarea obţinută. Dacă obţineţi felicitările tipărite la imprimantă, veţi tăia cu foarfeca fiecare felicitare şi vă veţi iscăli.

Veți putea adresa multe felicitări, unele arătînd ca cele din figura 43:

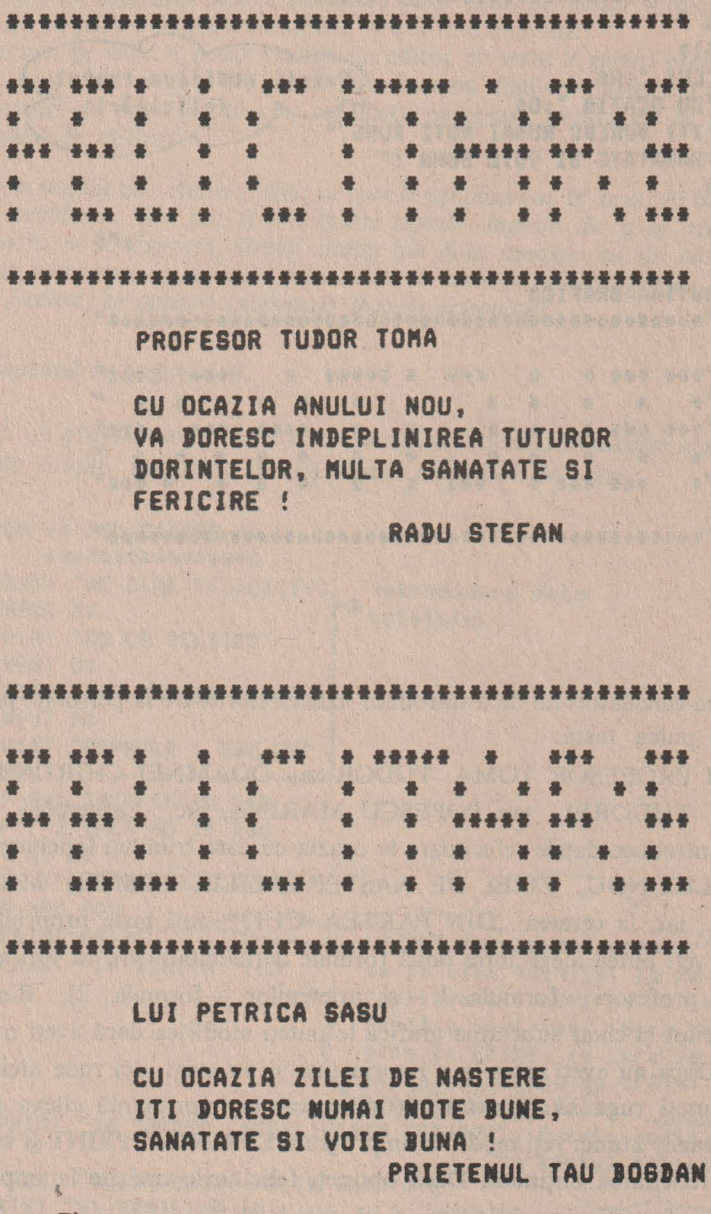


Fig. 43

În ce scopuri mai puteți realiza programe de acest fel? Sigur! Pentru a trimite scrisori, invitații etc. cu diverse ocazii.

Iată două modele în fig. 44:

```
**************************************
        Draga unchiule Paul,
Iti multumesc mult pentru cadoul
pe care mi l-ai facut de ziua
mea. Trenuletul este foarte frumos
si ma joc deseori cu el.

        Nepotelul tau,
              Dorel
**************************************


   *************************
   *  I N V I T A T I E  *
   *************************

        Draga Doina,
Pe  data de 23 mai implinesc
virsta de 14 ani. Cu aceasta
ocazie  organizez o mica pe-
trecere  duminica  28 mai la
ora 17, la  care  mi-ar face
placere sa vii.
      Ne vom amuza cu citeva
jocuri  noi  si  sper ca vom
dansa.  Adresa  mea   este:
Bd. A.I.Cuza  nr.53,  et.2,
ap.3.
        Prietena ta,
        Maria
```

Fig. 44

EXPERIMENTE ŞI MODELE CU AJUTORUL CALCULATORULUI

Se pot realiza mai multe programe cu care se pot face experimente şi se pot analiza informaţii. Aceste programe reprezintă simple exemple de utilizare a calculatoarelor de către matematicieni, economişti şi alţi cercetători. Unele din aceste programe sînt ele însele experimente. Se pot modifica diverşi factori în program şi, apoi, se poate urmări ce se întîmplă. Aceste programe sînt numite **modele**.

Programele utilizate de oamenii de ştiinţă sau alţi cercetători sînt deseori foarte complexe şi funcţionează pe calculatoare foarte puternice. Calculatoarele sînt ideale pentru realizarea de modele care utilizează ecuaţii matematice, deoarece ele pot face calcule foarte rapid. Iată cîteva exemple în care se arată cum sînt utilizate aceste programe:

MODELAREA APARIŢIEI UNIVERSULUI

Calculatoarele sînt folosite pentru a modela evenimente care sînt prea vaste sau complexe pentru a fi studiate în realitate. De exemplu, fizicienii utilizează unul dintre cele mai puternice calculatoare din lume, Cray-1, pentru a modela naşterea Universului. Modelul respectiv se bazează pe teoria conform căreia Universul s-a născut printr-o explozie uriaşă numită big bang. **Programul foloseşte ecuaţii matematice pentru a reprezenta ce s-a întîmplat începînd cu prima secundă după explozie, trasînd, apoi, structura Universului de atunci şi pînă acum.**

Prin compararea rezultatelor modelului privitoare la structura Universului cu măsurătorile şi observaţiile actuale, oamenii de ştiinţă pot desprinde concluzii referitoare la veridicitatea teoriei.

PROIECTAREA MAŞINILOR

Modelele pentru calculatoare sînt utilizate pentru proiectarea şi testarea noilor produse, ca de exemplu autoturismele. Astfel, pentru proiectarea autoturismelor OLTCIT se folosesc programe pentru a testa comportarea diverselor componente ale maşinii (motor, direcţie, caroserie etc.) la diferite viteze, la curbe în cazul unui impact etc.

De asemenea, forma autoturismului este proiectată în funcţie de rezultatele simulărilor pe calculator a rezistenţei pe care autoturismul o opune cînd este în mişcare, la diferite viteze, în condiţii de vînt din diferite direcţii, etc. Sînt proiectate piese noi sau componente din alte materiale (de obicei mai uşoare), în funcţie de rezultatele modelelor.

Iată un program simplu care realizează modelul aruncării unei monede. Puteţi experimenta cu ajutorul său şi urmări rezultatele care se obţin atunci cînd se va arunca moneda de un anumit număr de ori.

```
10 REM ** MODEL **
       ***********
20 CLS
30 LET S=0
40 LET B=0
50 PRINT "CITE ARUNCARI?"
60 INPUT N
70 FOR K=1 TO N
80 LET X=RND
90 IF X<=0.5 THEN LET S=S+1
100 IF X>0.5 THEN LET B=B+1
110 NEXT K
120 PRINT "STEMA S-A OBTINUT DE ";S;" ORI"
130 PRINT "BANUL S-A OBTINUT DE ";B;" ORI"
```

Se iniţializează variabilele care reprezintă de cîte ori s-a obţinut stema şi respectiv banul.

Calculatorul generează un număr cuprins între 0 şi 1.

Dacă acest număr este mai mic sau egal cu 0,5 convenim că am obţinut stema, dacă este mai mare, convenim că am obţinut banul.

Atunci cînd rulaţi programul, veţi introduce numărul de aruncări ale monedei. Programul va realiza modelul aruncării, rezultînd numărul de apariţii a stemei şi a banului.

De ce am comparat numărul aleator generat cu 0,5? Deoarece atunci cînd se aruncă o monedă se poate obţine în egală măsură stema sau banul, şansa (probabilitatea) de apariţie a fiecărei feţe fiind, deci, de 1/2.

20. *Realizaţi următorul experiment: încercaţi modelul (programul) pentru* **10, 100, 1000** *de aruncări ale monedei şi înregistraţi datele obţinute într-un tabel, ca cel de mai jos:*

Număr de aruncări	Rezultate			
	Obţinerea stemei		Obţinerea banului	
	Numărul (frecvenţa de apariţie)	Procentul din numărul de aruncări (%)	Numărul (frecvenţa de apariţie)	Procentul din numărul de aruncări (%)
10				
100				
1000				

Ce observaţi în legătură cu datele obţinute?

Răspunsul la pagina 141.

Deşi programul prezentat este foarte simplu, el arată cum un model realizat pe calculator poate economisi timp şi resurse materiale, comparativ cu un experiment real.

Încercaţi să aruncaţi moneda de 100 de ori şi să scrieţi rezultatele obţinute.

Cît timp v-a luat acest experiment comparat cu cel în care calculatorul v-a dat rezultatul?

III. Cum se pot realiza desene cu calculatorul

UTILIZAREA CULORILOR

Calculatorul poate produce opt culori diferite, fiecare dintre ele fiind reprezentată printr-un număr (cod). Fiecare culoare poate fi folosită în trei scopuri: pentru colorarea marginii (chenarul) ecranului, pentru colorarea fondului pe care este înscris un caracter şi pentru colorarea caracterului propriu-zis.

Următorul tabel arată culorile şi codurile utilizate de calculator în acest scop. Nu este necesar să memoraţi aceste coduri; cifra înscrisă pe tasta deasupra căreia se află culoarea reprezintă chiar codul culorii respective

Număr (sau tastă)	Culoare	Denumire în engleză (cum apare pe tastatură)
0	negru	BLACK
1	albastru	BLUE
2	roşu	RED
3	purpuriu	MAGENTA
4	verde	GREEN
5	bleu	CYAN
6	galben	YELLOW
7	alb	WHITE

Dacă nu aveţi la dispoziţie un TV sau un monitor color, culorile vor fi reprezentate prin diverse nuanţe de gri (ordonate de la închis spre deschis, corespunzător codurilor 0 — 7).

Cînd porniți calculatorul, culoarea caracterelor (literele, numerele, semnele) și a punctelor sau liniilor grafice este neagră, iar cunoarea chenarului și a fondului este albă.

Pentru a modifica aceste culori, aveți la dispoziție cele trei posibilități reprezentate prin următoarele cuvinte:

BORDER — se obține acționînd tasta B în modul **K**, după care se introduce un număr (între 0—7) care reprezintă culoarea chenarului

controlează culoarea chenarului (vezi fig. 45) din jurul ecranului grafic.

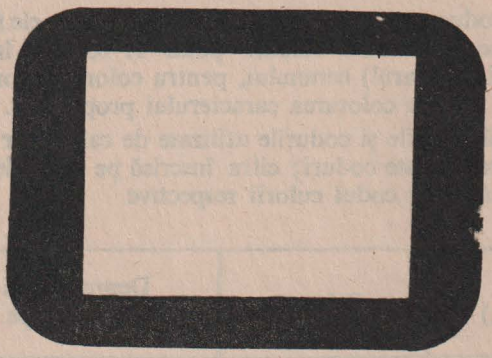


Fig. 45

INK — se obține acționînd tastele X și SS în modul **E**, după care se introduce un număr (între 0—7), care va reprezenta culoarea unui caracter, a punctelor grafice și a liniilor.

controlează culoarea caracterelor (literele, numerele, semnele), punctelor și liniilor grafice.

PAPER — se obține acționînd tastele C și SS în modul **E**, după care se introduce un număr (între 0—7) care va reprezenta culoarea fie a fondului întregului ecran grafic, fie a fondului unui caracter.

controlează culoarea fondului întregului ecran grafic sau a fondului pe care este înscris un caracter.

TESTAREA CULORILOR

Introduceţi următorul program:

```
10 REM ** TESTARE CULORI **
       **********************
20 PRINT "*";
30 GO TO 20
```

Ecranul se va umple cu un model de steluţe negre pe un fond alb, ca în fig. 46.

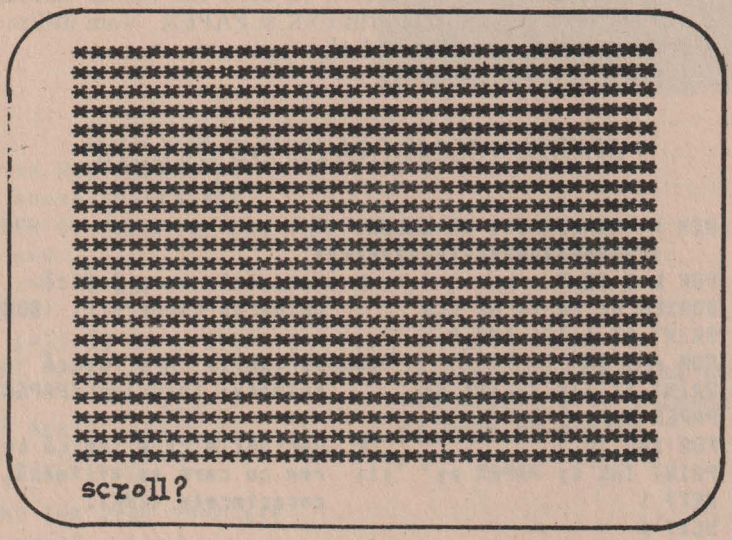


Fig. 46

Opriţi programul cu BREAK şi introduceţi cîteva comenzi pentru culori. De exemplu:

```
BORDER 4          Nu uitaţi să introduceţi
PAPER 2           CR după fiecare comandă.
INK 7
```

Rulaţi din nou programul **"Testarea culorilor"** Observaţi acum culorile (dacă folosiţi un TV color): verde pentru chenar, albă pentru steluţe şi roşie pentru fond?

Opriţi din nou programul şi rulaţi-l încă o dată după ce introduceţi alte comenzi pentru culori, de exemplu: **BORDER 3, PAPER 5** şi **INK 1**

CUM SE REALIZEAZĂ PROGRAME CU CULORI

În jocuri se pot utiliza **BORDER, PAPER** și **INK** pentru a realiza texte, tabele, modele și desene cu diverse culori. Utilizarea lui **BORDER** într-o linie de program va produce modificarea culorii chenarului de îndată ce, rulînd programul, calculatorul ajunge la linia respectivă. **INK** într-o linie de program va produce o nouă culoare pentru culorile caracterelor sau liniilor care vor apărea în continuare pe ecran. **PAPER** într-o linie va produce modificarea culorii fondului pe suprafața poziției caracterelor (deci din jurul caracterelor, inclusiv a punctelor și liniilor grafice). Dacă doriți ca întreg ecranul grafic să aibă o anumită culoare, atunci comanda **PAPER** va trebui urmată de **CLS**.

Se poate, de asemenea, folosi INK și PAPER după PRINT (sau PRINT AT) fiind separate prin intermediul semnului ;. În acest caz numai caracterele afișate cu PRINT vor avea culorile indicate prin INK și PAPER. Vom utiliza BORDER, PAPER și INK în cîteva jocuri cu culori.

JOCUL „COMBINAȚII DE CULORI"

```
 5 REM ** COMBINATII DE CULORI **
       **************************
10 FOR b=0 TO 7 ----------------> Variabila care indică
20 BORDER b: PAPER b: CLS         culoarea chenarului (BORDER).
30 PRINT AT 6,12; INK 9;b
40 FOR p=0 TO 7 ----------------> Variabila care indică
50 PRINT AT p+8,8; INK p;         culoarea fondului (PAPER).
   PAPER 9;p;
60 FOR i=0 TO 7 ----------------> Variabila care indică culoa-
70 PRINT INK i; PAPER p;" ";i;    rea cu care se afișează
80 NEXT i                         caracterele (INK).
90 NEXT p
100 NEXT b
```

Cînd rulați acest program, veți vedea toate combinațiile de BORDER, PAPER și INK. Liniile care încep cu FOR și cele cu NEXT marchează începutul și sfîrșitul ciclului care modifică succesiv codul tuturor culorilor de la 0 la 7.

Notă: Atît INK, cît și PAPER, pot lua și valoarea 9. Aceasta va face culoarea caracterelor sau a fondului fie neagră, fie albă.

PROGRAM PENTRU REALIZAREA DE HISTOGRAME

Următorul program utilizează culorile calculatorului pentru a produce histograme. Histograma este un mod de reprezentare grafică prin care un fenomen sau evoluția unui fenomen în timp sînt evidențiate sub formă de coloane verticale.

Programul de mai jos evidențiază temperatura maximă a aerului în timpul unei zile de primăvară în 12 țări ale Europei (vezi fig. 47).

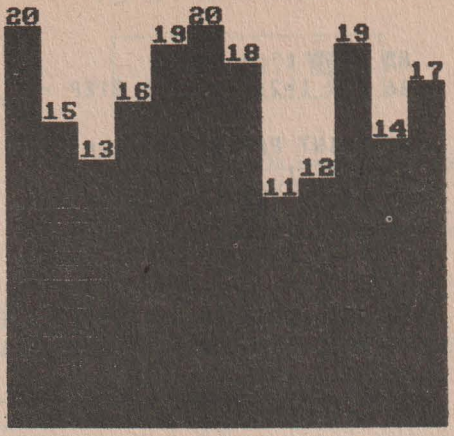


Fig. 47

```
 5 REM ** HISTOGRAME **
       ****************
10 BORDER 0: PAPER 1: CLS
20 LET c=4
30 FOR x=1 TO 12
40 READ t
50 FOR l=21 TO 21-t STEP -1
60 PRINT PAPER 6;AT l,c;"  "
70 NEXT l
80 PRINT INK 2;AT 20-t,c;t
90 LET c=c+2
100 NEXT x
110 DATA 20,15,13,16,19,20,18,
    11,12,19,14,17
```

Două spații goale

Temperaturile maxime se vor reprezenta prin coloane galbene cu numere.

La ce se mai pot folosi histogramele?

Cu ajutorul lor puteți reprezenta grafic:

— cîți elevi bruneți, blonzi, șateni și roșcați sînt într-o clasă;

— din cîte încercări au reușit să termine un joc (să îndeplinească obiectivul) mai mulți jucători;

— cîți elevi dintr-o clasă au media generală între 9 și 10, cîți între 7 și 9, cîți între 5 și 7 și cîți între 3 și 5.

Adăugați următoarele linii programului și modificați linia 110 astfel:

```
85 READ t
86 FOR i=21 TO 21-t STEP -1
87 PRINT PAPER 3;AT i,c;"  "
88 NEXT i
89 PRINT INK 1; PAPER 5;
   AT 20-t,c;t

110 DATA 20,6,15,4,13,5,16,6,19,10,
    20,8,18,6,11,4,14,6,19,8,14,
    9,17,7
```

Două spații goale

Temperaturile minime se vor reprezenta pe aceleași coordonate orizontale X prin coloane de culoare purpurie.

Acum programul va reprezenta grafic histograme duble: pe aceeași coloană care va reprezenta o anumită țară, se va evidenția atît temperatura maximă, cît și cea minimă dintr-o zi.

Puteți folosi histograme duble, de exemplu pentru a reprezenta prin coloane cîte încercări a efectuat fiecare jucător într-un joc și cîte din aceste încercări au fost reușite.

DESENE PE ECRAN

Cu calculatorul puteți realiza desene (grafică) prin două moduri de bază:

A. **utilizînd caractere grafice**. Poziționarea acestora pe ecran se face întocmai ca și a caracterelor pentru text, iar prin combinarea lor se pot realiza diverse desene.

B. **utilizînd ecranul grafic** de 256 × 176 de puncte, deci **puncte, linii, cercuri**, etc.

Amîndouă modurile de realizare a desenelor pot fi utilizate în programe și pot produce desene care să apară în același timp pe ecran.

Deoarece prima modalitate folosește ecranul avînd ca unitate de reprezentare caracterul (se pot afișa 32 de caractere pe rînd și 22 de linii de caractere) desenele rezultate în urma aplicării ei se mai numesc **grafică de rezoluție scăzută**, în timp ce pentru a doua modalitate, care folosește ecranul, avînd ca unitate de reprezentare punctul (256 × 176 puncte), desenele rezultate se mai numesc **grafică de rezoluție înaltă** (vezi grila din fig. 48).

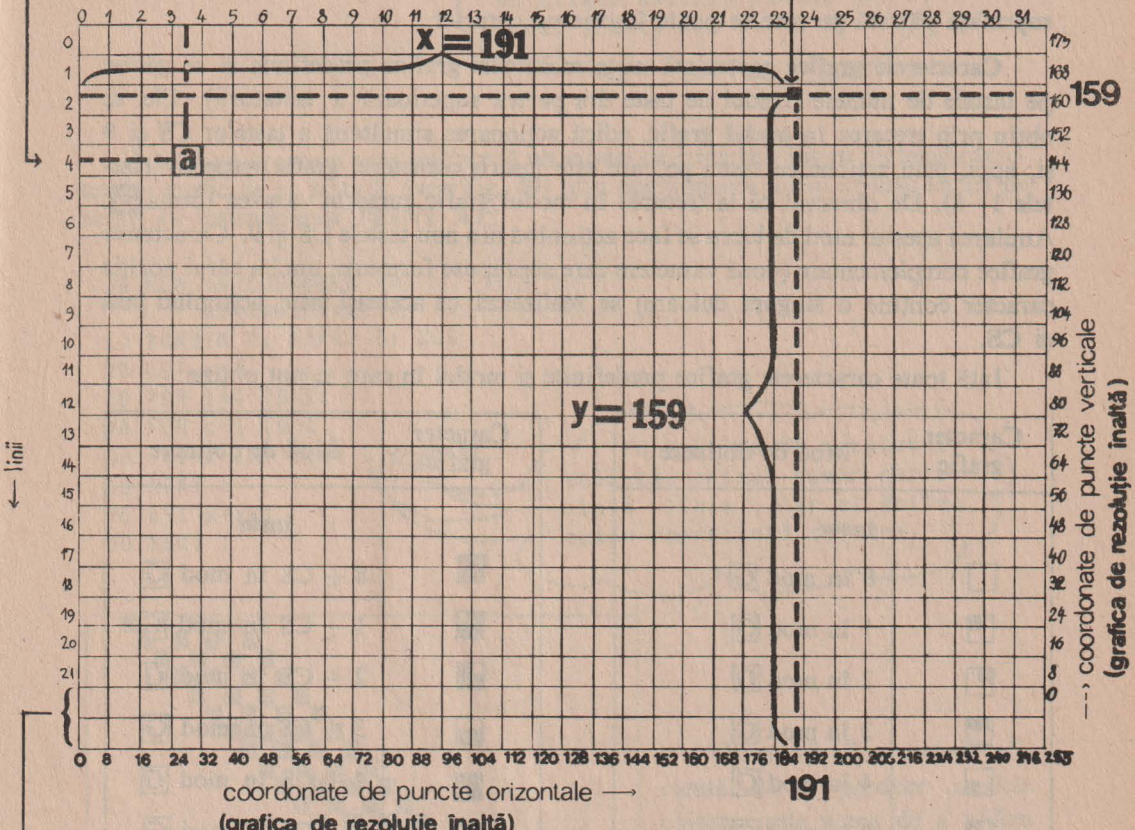
Fig. 48 Grila de afişare a ecranului.

Un exemplu de utilizare pentru grafica de rezoluţie scăzută: un caracter pe linia 4 şi coloana 3.

```
PRINT AT 4,3; "a"
```

```
PLOT 191,159
```

Un exemplu de utilizare pentru grafica de rezoluţie înaltă: un punct de coordonate 191, 159.



cele două linii din partea de jos a ecranului nu se pot utiliza în mod normal cu PRINT şi PLOT.

A. GRAFICA DE REZOLUȚIE SCĂZUTĂ

Utilizarea caracterelor grafice pentru desene

Pe ecranul de rezoluție scăzută fiecare caracter poate fi plasat pe ecran într-un loc determinat prin două numere: primul reprezintă linia sau rîndul (de la 0 la 21. inia 0 fiind cea din partea superioară a ecranului), al doilea reprezintă coloana (de la 0 la 31, coloana 0 fiind cea mai din stînga) (vezi fig. 48).

Imaginați-vă liniile (rîndurile) și coloanele pentru caractere ca niște dungi mai groase; orizontale și verticale, unele lîngă altele.

Intersecția unei linii cu o coloană va determina o poziție caracter, adică o mică suprafață pătrată pe care se poate înscrie un caracter.

Caracterele grafice reprezintă niște mozaicuri grafice predefinite și se găsesc pe tastele de numere (rîndul de taste din partea superioară a tastaturii). Ele se obțin prin trecerea în **modul grafic**, adică acționarea simultană a tastelor **CS** și **9** și, apoi, prin acționarea tastei pe care este înscris caracterul grafic respectiv (tastele 1—8). De observat că la trecerea în modul grafic cursorul va avea forma **G**. Anularea acestui mod de lucru se face acționînd din nou tastele CS și 9. Caracterele grafice complementare (două caractere care suprapuse formează unul a cărui poziție caracter conține o singură culoare) se realizează cu aceleași taste, acționînd însă și CS.

Iată toate caracterele grafice predefinite și modul în care se pot obține:

Caracter grafic	Mod de obținere
	tasta
□	8 în mod G
▗	1 în mod G
▖	2 în mod G
▄	3 în mod G
▟	4 în mod G
▙	5 în mod G
▚	6 în mod G
▛	7 în mod G

Caracter grafic	Mod de obținere
	taste
■	8 + CS în mod G
▝	1 + CS în mod G
▘	2 + CS în mod G
▀	3 + CS în mod G
▜	4 + CS în mod G
▟	5 + CS în mod G
▞	6 + CS în mod G
▜	7 + CS în mod G

Notă: **Observați cum caracterele grafice din cele două tabele alăturate sînt complementare.**

Iată un program care va realiza umplerea tuturor poziţiilor de caractere de pe ecran cu culori alese la întîmplare, folosindu-se în acest scop caracterul grafic corespunzător tastei 8

```
10 REM ** CARACTER GRAFIC **
       **********************
20 BORDER 1: INK RND*8
30 PRINT "■";
40 GO TO 20
```

Pentru a obţine acest caracter, se acţionează tasta CS şi 9 şi, apoi, se apasă tasta 8 (tot cu CS). Nu uitaţi, apoi, să ieşiţi din modul grafic tot cu CS şi 9.

Iată şi un program care realizează un model de curcubeu tot cu ajutorul caracterului grafic de pe tasta 8, (vezi fig. 49) prin poziţionarea lui în diverse locuri pe ecran cu instrucţiunea **PRINT AT**:

```
 5 REM ** CURCUBEU **
       ***************
10 BORDER 0: PAPER 5: CLS
20 LET x=1
30 FOR l=0 TO 21
40 FOR c=1 TO 6
50 PRINT INK c;AT l,c+x;"■"
60 NEXT c
70 LET x=x+1
80 NEXT l
```

In grafica de rezoluţie scăzută, primul număr reprezintă linia , iar al doilea număr reprezintă coloana.

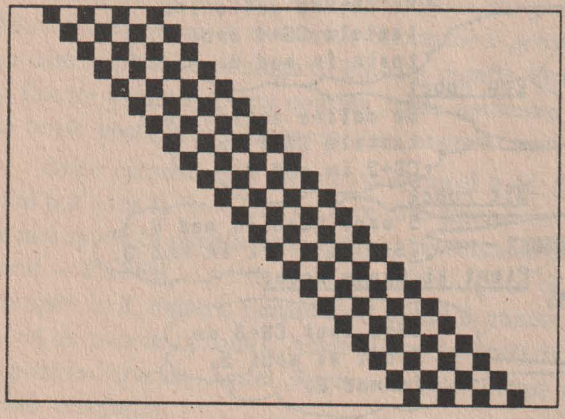


Fig. 49

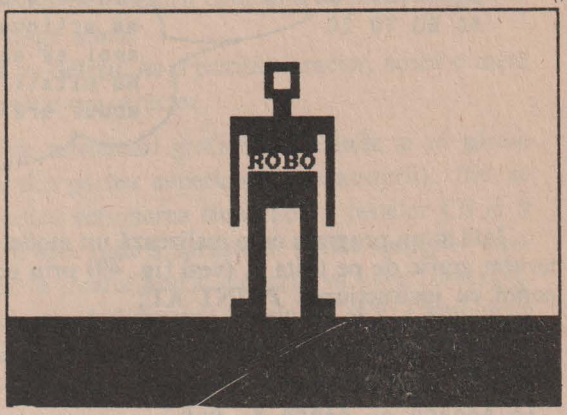
Observaţi că o metodă eficientă de a produce modele colorate este aceea de a utiliza cicluri **FOR-NEXT** în programe. În exemplul nostru, ciclurile **FOR-NEXT** au fost utilizate pentru a modifica culorile şi poziţiile produse de **INK** şi, respectiv, **AT**.

Programarea desenelor cu caractere grafice

Pe ecranul de rezoluție scăzută puteți realiza "picturi" cu ajutorul unor schițe (proiecte) prealabile întocmite pe grila care reprezintă ecranul de 32 coloane și de 22 rînduri. Apoi, se vor selecta caracterele grafice necesare și se va introduce programul, linie cu linie, construindu-se astfel întregul desen.

Următorul program demonstrează o posibilitate de utilizare a caracterelor grafice pentru desenarea pe ecran a unui robot (vezi fig. 50).

Fig. 50



Toate formele cu care este desenat acest robot pot fi găsite pe tastele 1—8. Puteți să așteptați pînă introduceți toate liniile și, apoi, să rulați programul, dar dacă veți rula programul după ce introduceți fiecare linie, veți vedea cum diferite părți ale robotului se vor "lega" între ele.

Notă: Dacă veți introduce greșit unul sau mai multe caractere grafice (care nu se potrivesc cu schița), puteți edita linia respectivă și modifica acele caractere grafice întocmai cum realizați acest lucru cu literele sau numerele incorecte dintr-o linie.

```
 5 REM ** DESEN ROBOT **
     ******************
10 BORDER 2: PAPER 1: CLS
20 PRINT INK 2;AT 3,15;"▀▀"
25 PRINT INK 2;AT 4,15;"▐▌"
30 PRINT INK 2;AT 5,15;"▐ "
40 PRINT INK 5;AT 6,13;"▐▀▀▀▀▀▀▀▌"
50 FOR I=7 TO 10: PRINT INK 5;
   AT I,13;"▌       ▐":NEXT I
```

Se obține acționînd tastele CS+4 apoi 7, toate în mod 6.

Cap robot

Se obține acționînd tastele CS+1 apoi CS+2 în mod 6.

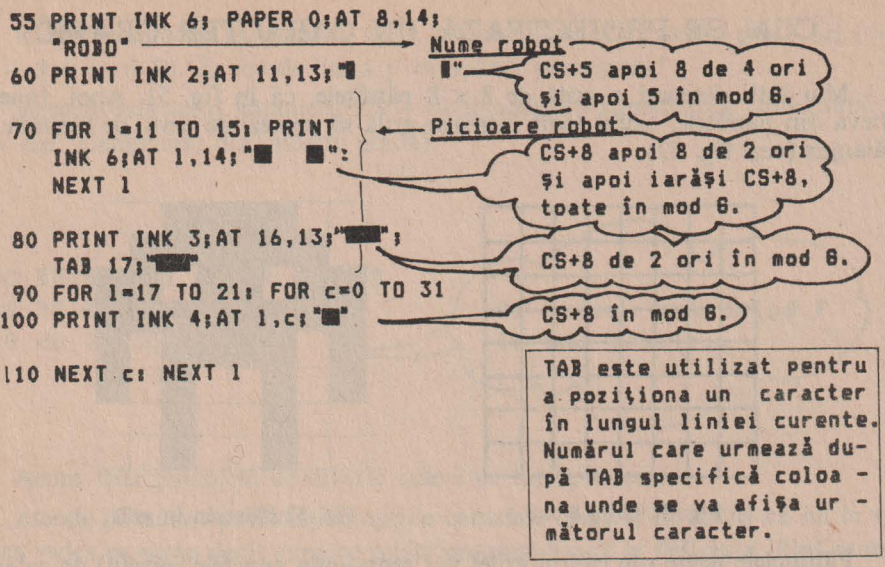
Șold robot

5 apoi CS+5 în mod 6.

CS+8 de 6 ori în mod 6.

Piept și brațe robot

CS+5 apoi CS+8 de 4 ori și apoi 5, în mod 6.



21. *Scrieți un program care să realizeze cu ajutorul caracterelor grafice afișarea pe ecran a cuvîntului ROBOT (scris mare).*

Răspunsul la pagina 141.

CUM SE POT CREA NOI CARACTERE GRAFICE

Calculatorul nu este limitat la caracterele grafice pe care le găsiți pe tastatură. Într-un loc special al memoriei, calculatorul poate memora și alte caractere grafice pe care le puteți crea singuri. Ele sînt numite **caractere grafice definite de utilizator** și fiecare program poate conține 21 de asemenea caractere. Fiecare caracter creat se poate apela prin acționarea unei anumite taste de literă în modul grafic.

Orice caracter este definit cu ajutorul a 64 de puncte (se mai numesc pixeli), care pot avea fie culoarea cu care este desenat caracterul — **INK** — (și în acest caz se mai spune că punctul este aprins), fie culoarea fondului pe care este desenat caracterul — **PAPER** — (și în acest caz se mai spune că punctul este stins). Punctele sînt aranjate în 8 rînduri, fiecare rînd avînd 8 puncte (8 rînduri × 8 puncte fiecare = = 64 de puncte în total). Acest tablou (pătrat) de 8 × 8 puncte formează pe ecran o poziție caracter. Astfel, fiecare caracter ocupă o poziție caracter pe ecranul de joasă rezoluție.

CUM SE PROIECTEAZĂ UN CARACTER GRAFIC

Mai întîi desenați o grilă de 8 × 8 pătrățele, ca în fig. 51. Apoi, înnegriți cîteva din pătrățele, astfel încît întreaga grilă să reprezinte ceva, de exemplu: un păianjen (vezi fig. 52).

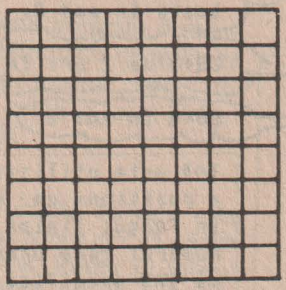


Fig. 51 Grila 8 × 8.

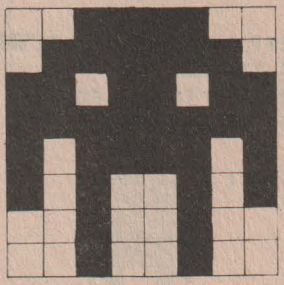


Fig. 52 Păianjen în grilă.

Pătrățelele negre din cadrul grilei vor reprezenta punctele (pixelii) de culoarea (**INK**) caracterului.

Considerați fiecare pătrățel negru (umplut) ca fiind reprezentat de un 1, iar fiecare pătrățel gol ca fiind reprezentat de un 0.

Fiecare caracter grafic utilizator va fi identificat printr-o literă de la **A** la **U** (puteți alege orice literă).

Pentru a programa caracterul, va trebui să introduceți opt linii de program **POKE USR**, fiecare terminîndu-se cu **BIN**, urmat de un număr binar, format din cele 8 cifre 0 și 1 pentru fiecare rînd.

Să atribuim caracterului păianjen litera **p**.

Programul pentru formarea caracterului va fi:

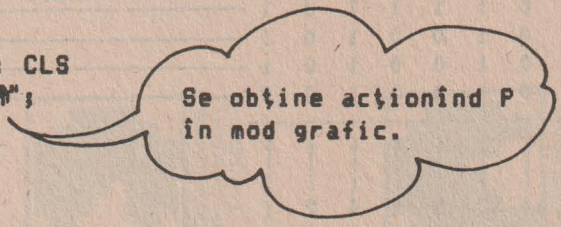
```
 5 REM ** CARACTER PAIANJEN **
       ************************
10 POKE USR "p",BIN 00111100 ——— linia 0 ——→ POKE modifică
11 POKE USR "p"+1,BIN 01111110    linia 1   conținutul unei
12 POKE USR "p"+2,BIN 11011011    linia 2   adrese de memo-
13 POKE USR "p"+3,BIN 11111111       .      rie, în cazul
14 POKE USR "p"+4,BIN 10111101       .      nostru adresa în
15 POKE USR "p"+5,BIN 10100101       .      care se memorează
16 POKE USR "p"+6,BIN 10100101       .      informația pentru
17 POKE USR "p"+7,BIN 00100100              fiecare linie a
                                            caracterului.
```

Liniile (rîndurile) cîte 8 de 0 și 1 pentru caracterul "p".

Rulați acum acest program și, apoi, acționați tasta P în modul grafic G (deci CS + 9 și apoi P). În loc de litera p va apărea păianjenul!

Ștergeți caracterul păianjen (CS + 0) și adăugați următoarele linii (după ce veți ieși, bineînțeles, din modul grafic):

```
20 BORDER 1: PAPER 0: CLS
30 PRINT INK RND*7;"p";
40 GO TO 30
```



Acum apar păianjeni de diverse culori pe întregul ecran.

Atenție! Cînd vă proiectați propriile caractere grafice, nu uitați că nu le veți putea vedea pe ecran decît după ce rulați programul care le definește. Pînă atunci, aceste caractere vor apărea ca litere (de exemplu pentru caracterul păianjen, litera p).

GENERAREA MAI SIMPLĂ A CARACTERELOR GRAFICE UTILIZATOR

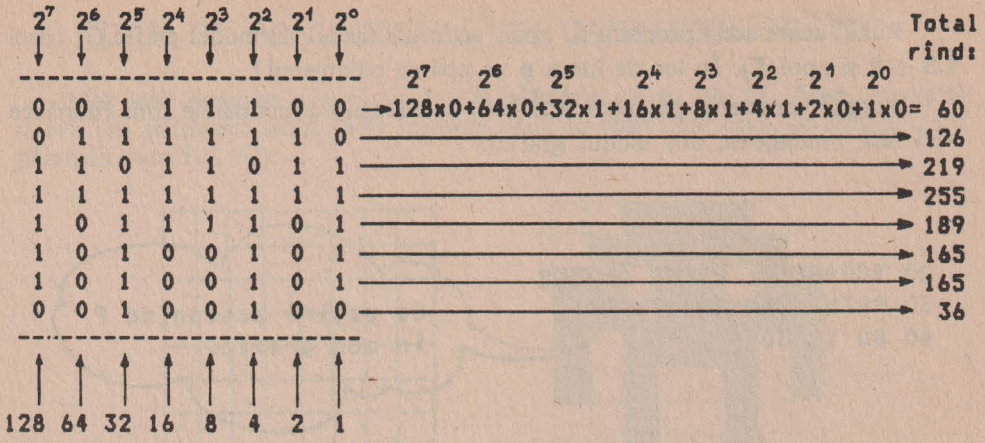
Există un mod mai simplu de a genera caracterele grafice, și anume utilizînd numere zecimale care vor fi citite cu **READ** și **DATA**. Mai întîi, trebuie transformate cele 8 numere binare, formate din 0 și 1, în numere zecimale. Acest lucru se poate face introducînd **PRINT BIN**, urmat de un număr, de exemplu:

```
PRINT BIN 00111100
```

va avea ca rezultat afișarea lui 60, echivalentul zecimal al numărului binar 00111100.

În cazul păianjenului, cele 8 numere zecimale vor fi: 60, 126, 219, 255, 189, 165, 165 și 36.

Alt mod de a transforma numerele binare corespunzătoare unui rînd în numere zecimale este de a realiza acest lucru fără intermediul calculatorului, folosind grila inițială de 8 × 8 pentru care cifrele binare de pe coloane au o pondere determinată (puteri ale lui 2). În cazul păianjenului, lucrul se face ca în fig. 53.



Se obţin, bineînţeles, aceleaşi valori ca şi la transformarea cu **BIN**.

Avînd acum la dispoziţie aceste numere, putem să le introducem prin intermediul instrucţiunilor **READ** şi **DATA**. Iată cum va arăta noul program:

```
 3 REM ** PAIANJEN **
       ***************
10 LET x=0 TO 7
20 READ y
30 POKE USR "p"+x,y
40 NEXT x
50 DATA 60,126,219,255,189,165,165,36
```

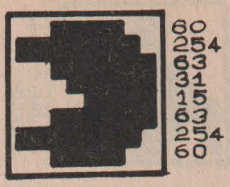
Programul va memora oricare 8 numere zecimale (între 0 şi 255), care vor crea un caracter. În acest scop se modifică **p** din linia 30 cu orice literă doriţi şi se vor introduce numerele pe linia 50, în ordine, separate prin virgule.

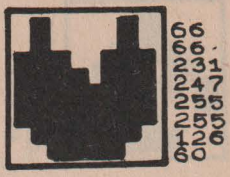
Pentru a obţine caracterul dorit, se va acţiona tasta, corespunzătoare literei, în modul grafic.

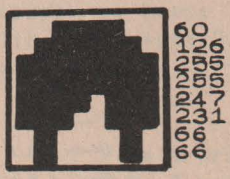
La ce foloseşte crearea de noi caractere grafice în jocuri?

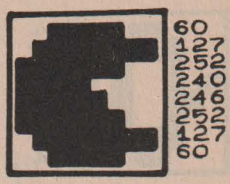
Bineînţeles, pentru a crea diverse personaje (să le numim spiriduşi) şi obiecte (asteroizi, rachete etc.) care apoi vor fi deplasate pe ecran. În figura 54 aveţi la dispoziţie o întreagă colecţie de spiriduşi împreună cu valorile asociate pentru crearea lor. De asemenea, se mai pot folosi pentru desenarea unor figuri (piese), de exemplu: piesele de şah, sau pentru crearea literelor specifice limbii române (care aţi observat că nu se pot obţine în mod obişnuit de la tastatură): ă, â, î, ş, ţ.

Spiriduşi

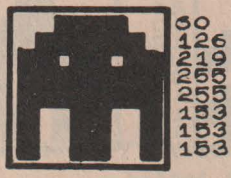


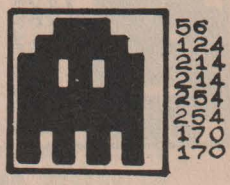




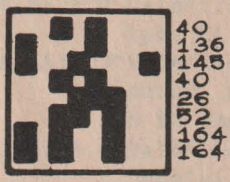


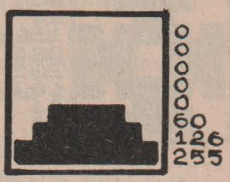
Monştri



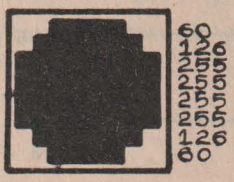


Descompunerea spiriduşului





Pioni



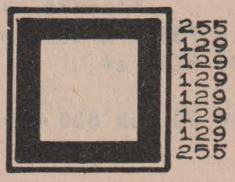
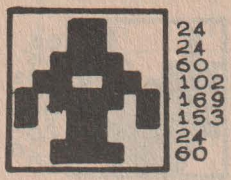
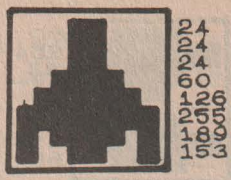


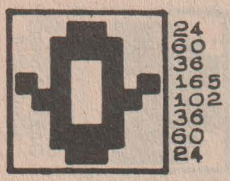
Fig. 54 a.

Rachete

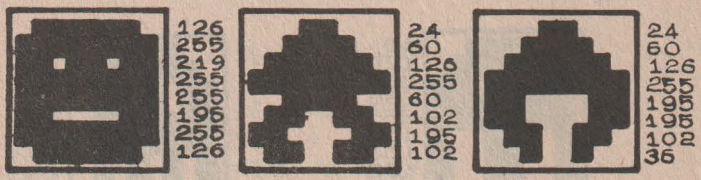




Canoe



Extratereştri



Morişcă

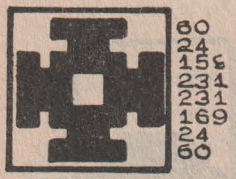


Fig. 54 b.

Iată şi un program care desenează pe ecran tabla de şah, împreună cu piesele aşezate ca pentru începutul unei partide:

```
 5 REM ** TABLA DE SAH **
       ********  **********
10 FOR x=1 TO 6 ──────────► Sînt 6 feluri de piese: pioni,
20 READ a$                  ture, cai, nebuni, regi şi
                            regine.
30 GO SUB 500 ────────────► Apelul subrutinei pentru defi-
                            nirea caracterelor pentru piese.
```

```
40 NEXT x
50 BORDER 4: PAPER 1: CLS
60 FOR l=7 TO 14 STEP 2:
   FOR c=12 TO 19 STEP 2
70 PRINT AT l,c;" ■";AT l+1,c;"■ "
80 NEXT c: NEXT l
90 PRINT AT 7,12;"¥±¤¤¥±¤±¥" ──► Punerea pieselor pe tablă.
                                  Tastele T,C,N,Q,K,N,C,T
                                  toate în mod G.
```

Aici se poate insera codul unei culori pentru a se deosebi piesele unui adversar de ale celuilalt. De exemplu se trece în mod E și apoi se acționează tasta 2 (piesele vor avea culoarea roșie) împreună cu tasta CS.

```
100 PRINT AT 8,12;"±±±±±±±±" ──► De 8 ori tasta P în mod G.
110 PRINT AT 13,12;"±±±±±±±±" ──► De 8 ori tasta P în mod G.
120 PRINT AT 14,12;"¥±¤±¥¤±¥" ──► Tastele T,C,N,K,Q,N,C,T
                                  toate în mod G.
130 GO TO 130
500 FOR n=0 TO 7
510 READ y
520 POKE USR a$+n,y
530 NEXT n
540 RETURN
550 DATA "p",0,0,16,56,56,16,124,0 ──► Valorile pentru definirea
                                  piesei pion. Ea se obține cu
                                  tasta P în mod G.
560 DATA "t",0,84,124,56,56,124,──► Valorile pentru definirea turei
    124,0                         Ea se obține cu T în mod G.
570 DATA "c",0,16,56,120,24,56,──► Valorile pentru definirea calu-
    124,0                         lui. El se obține cu C în mod G.
580 DATA "n",0,16,40,68,108,56,──► Valorile pentru definirea nebu-
    124,0                         nului. El se obține cu
                                  N în mod G.
590 DATA "k",0,16,56,16,56,68,56,0 Valorile pentru definirea
                                  regelui. Se va obține cu
                                  K în mod G.
600 DATA "q",0,84,40,16,108,124,──► Valorile pentru definirea
    124,0                         reginei. Se va obține cu
                                  Q în mod G.
```

22. A. *Găsiți seturile de valori (numerele zecimale) pentru a crea caractere grafice care să reprezinte:*

a) un omuleț

b) o mașinuță

c) litera ă

d) litera î

e) litera ş

f) litera ţ

„B. Indicaţi secvenţa de taste ce trebuie acţionate după ce aţi definit caracterele grafice de la exerciţiul **A**, *pentru a obţine pe ecran afişarea următorului text:*

Un omuleţ ☆ într-o maşinuţă ☆

Presupunem că am ales pentru omuleţ litera o; pentru maşinuţă litera m; pentru ă, litera a; pentru î litera i; pentru ş litera s; pentru ţ litera t.

Răspunsurile la pagina 142.

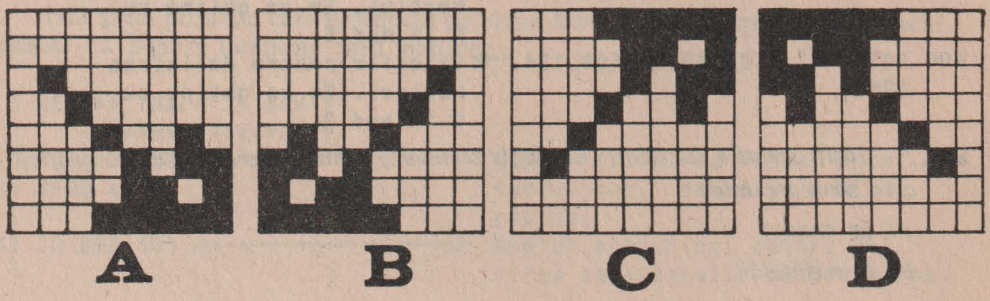
CREAREA DE SPIRIDUŞI DE DIMENSIUNI MAI MARI

Sigur se poate ca într-un joc să vă nemulţumească dimensiunea mică a unui spiriduş, desenul acestuia încadrîndu-se într-o poziţie caracter. Aveţi vreo idee pentru a crea în acelaşi mod spiriduşi de dimensiuni mai mari?

Sigur că da! Puteţi defini părţi ale spiriduşilor pe cîte o poziţie caracter şi, apoi, prin îmbinarea acestor părţi, să formaţi un spiriduş de dimensiuni mai mari

Iată cum arată grilele pentru patru caractere (să zicem **A, B, C** şi **D**) care,. îmbinate, vor forma un "gîndac" de dimensiuni mai mari (fig. 55).

Fig. 55



Valorile pentru cele 4 caractere sînt:

A: 0, 0, 64, 32, 18, 11, 13, 31

B: 0, 0, 2, 4, 72, 208, 176, 248

C: 31, 13, 11, 18, 32, 64, 0, 0

D: 248, 176, 208, 72, 4, 2, 0, 0

Se poate observa simetria acestor caractere. De exemplu: din lista valorilor, se observă simetria lui **A** cu **C** și **B** cu **D**, lista de valori pentru **C** obținîndu-se prin citirea în ordine inversă a listei pentru **A**, iar lista valorilor pentru **D** obținîndu-se prin citirea inversă a listei de valori pentru **B**.

Haideți să aranjăm cele 4 pătrate astfel:

A — stînga sus

B — dreapta sus

C — stînga jos

D — dreapta jos

Vom obține un gîndac mai mare (vezi fig. 56).

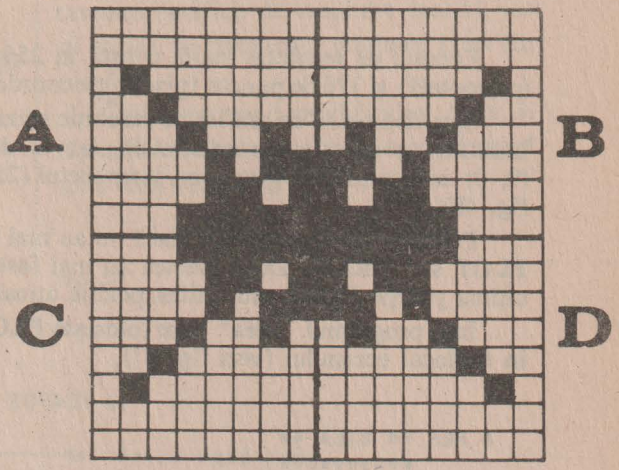


Fig. 56

Pentru a afișa acest gîndac pe ecran nu aveți altceva de făcut decît a aranja cele 4 caractere astfel: caracterul **B** după **A** pe o linie-rînd (să zicem **r**), pe linia următoare (**r** + **1**) caracterul **D** după **C**, astfel încît **C** să fie sub **A** (deci, pe aceeași coloană). De exemplu:

```
PRINT AT r,x;"AB": PRINT AT r+1,x;"CD"
```

Definirea de caractere grafice cu două culori

Puteți simula amestecarea a două culori în cadrul unui caracter grafic utilizator. Pentru aceasta va trebui să creați un caracter care va fi jumătate de culoarea caracterului — **INK** — și cealaltă jumătate de culoarea fondului — **PAPER**. De exemplu: următorul program va defini un pătrățel în care cele 2 culori (pentru **INK** și pentru **PAPER**) se vor amesteca în mod egal:

```
 5 REM ** AMESTECARE DOUA CULORI **
       ******************************
10 FOR x=0 TO 6 STEP 2
20 POKE USR "a"+x,BIN 10101010
30 POKE USR "a"+x+1,BIN 01010101
40 NEXT x
```

Deci, trebuie să definiți două linii de puncte (pixeli) și, apoi, să le utilizați alternativ în cadrul caracterului grafic utilizator.

B. GRAFICA DE REZOLUȚIE ÎNALTĂ

Ecranul de rezoluție înaltă constă în 256 de puncte (pixeli) pe coordonata **X** (orizontală) și 176 de puncte (pixeli) pe coordonata **Y** (verticală).

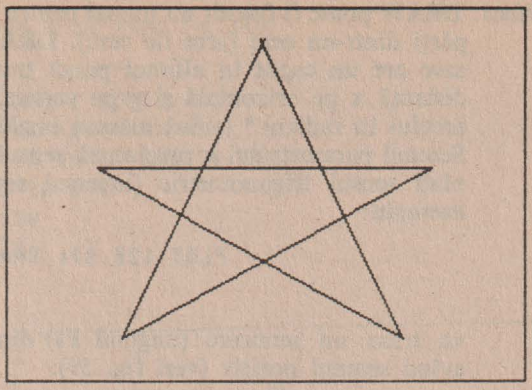
Spre deosebire de grafica de rezoluție scăzută, primul număr folosit îl reprezintă întotdeauna coordonata orizontală, iar al doilea, coordonata verticală. Punctul (0, 0) este colțul din stînga jos, iar punctul (255, 175) este cel din dreapta sus (vezi fig. 48).

Prima poziție este (0 ,0), dacă nu au mai fost utilizate în program instrucțiuni **PLOT** sau **DRAW**. Dacă acestea au mai fost utilizate, atunci această poziție este ultima poziție **PLOT** sau ultima poziție atinsă de cel mai recent **DRAW**.

Iată programul "Stea" care folosește **PLOT** și **DRAW**, pentru a desena o stea în mijlocul ecranului (vezi fig. 57). :

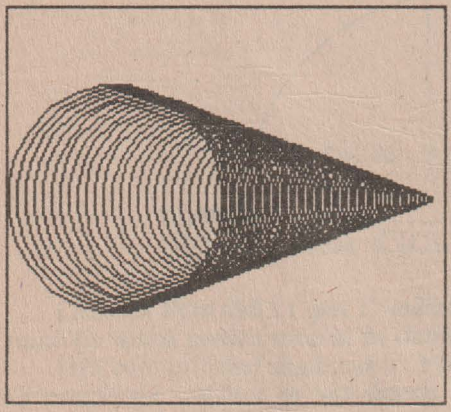
```
 5 REM ** STEA **
       ***********
10 INK 2
20 PLOT 128,174 ------> Mută poziția de start în partea
                        de sus a ecranului.
30 DRAW 70,-140  ---\
40 DRAW -152,80     |-> 5 instrucțiuni DRAW care trasează
50 DRAW 164,0       |   5 linii roșii.
60 DRAW -150,-80    |
70 DRAW 70,140   ---/
```

Fig. 57



Acum adăugați următoarele linii programului:

```
7 BORDER 1: PAPER 6: INK 1: CLS
8 CIRCLE 128,87,87
```



Cuvîntul **CIRCLE** este urmat de 3 numere separate prin virgulă; primele două vor reprezenta coordonatele centrului cercului (grafica de rezoluție înaltă), iar al treilea va reprezenta mărimea razei (în pixeli).

Iată cum prin modificarea razei și a poziției centrului se poate obține desenul unui con (vezi fig. 58):

Fig. 58

```
 5 REM ** CON **
       *********
10 CLS: INK 1: PAPER 4: BORDER 6:
   CLS
20 READ cr ──────────────────►  Prima rază mare.

30 LET x=50 ─────────────────►  Poziția pe orizontală a
                               centrului (variabilă).

40 FOR r=cr TO 1 STEP -1 ────►  Modificarea razei.
50 CIRCLE x,90,r
60 LET x=x+3 ────────────────►  Modificarea poziției pe
                               orizontală.
70 NEXT r
80 DATA 50
```

Notă: **DRAW** poate fi folosit nu numai pentru trasarea liniilor, ci și pentru a trasa părți dintr-un cerc (arce de cerc). **DRAW x, y, u** va trasa un arc de cerc care are un capăt în ultimul punct trasat pe ecran, iar celălalt aflat la o distanță **x** pe orizontală și **y** pe verticală, în timp ce **u** specifică lungimea arcului în radiani * (adică măsura unghiului la centru exprimat în radieni). Semnul parametrului **u** precizează sensul în care se trasează arcul, + însemnînd sensul trigonometric (inversul sensului orar), iar — sensul orar. De exemplu:

```
PLOT 128,87: DRAW -50,0,PI
```

va trasa un semicerc (unghiul **PI**) din centrul ecranului spre stînga, **PI** avînd semnul pozitiv (vezi fig. 59).

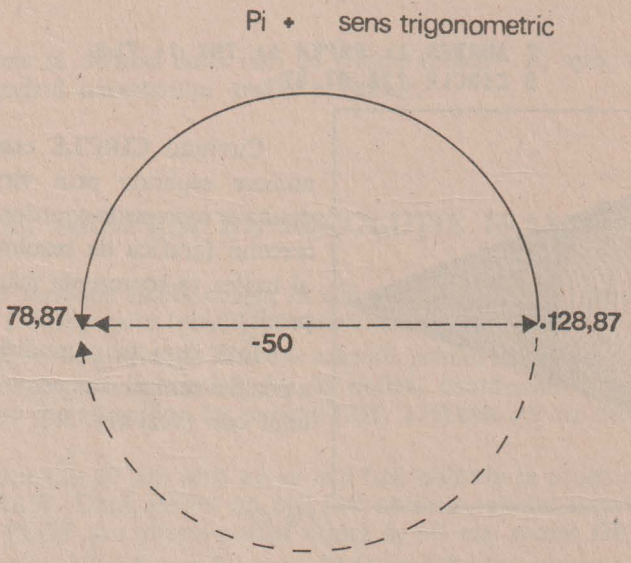


Fig. 59

CUM SE POT COLORA FIGURILE

Formele solide (colorate cu altă culoare decît cea a fondului) se pot obține ușor în grafica de înaltă rezoluție desenînd mai multe linii apropiate una de alta, folosind de exemplu: un ciclu **FOR-NEXT** care modifică poziția pentru **DRAW**, astfel încît aceasta să crească de fiecare dată cu o unitate.

---

* un radian este un arc de cerc care are lungimea egală cu raza.

Iată cum se poate obține un triunghi solid (vezi fig. 60):

```
 5 REM ** TRIUNGHI SOLID **
       **********************
10 BORDER 1: PAPER 6: INK 2: CLS
20 FOR x=-100 TO 100
30 PLOT 128,150
40 DRAW x,-120
50 NEXT x
```

Puteți obține efecte interesante dacă veți trasa liniile care umplu triunghiul cu o distanță între ele (utilizînd **STEP**).

Încercați mai multe valori ale pasului în linia 20 și rulați din nou programul.

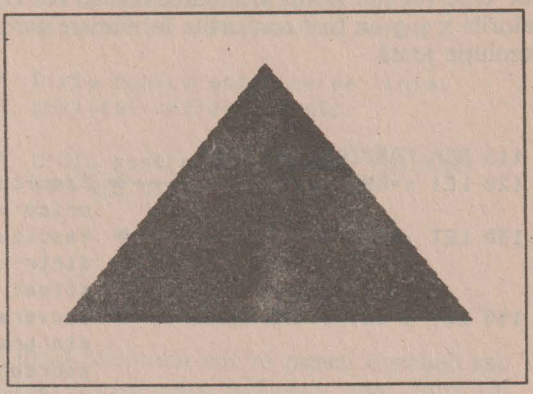


Fig. 60

UTILIZAREA CICLURILOR ÎN GRAFICĂ

Ciclurile **FOR-NEXT** pot fi utilizate cu rezultate spectaculoase cu grafica de rezoluție înaltă pentru crearea de desene formate din linii și figuri regulate.

Iată cum utilizînd două cicluri **FOR-NEXT** și instrucțiunile **PLOT** și **DRAW**, în programul următor se vor desena 5 piramide (vezi fig. 61).

```
  5 REM ** PIRAMIDE **
        **************
 10 BORDER 0: PAPER 1: INK 6
 20 CLS
 30 FOR y=0 TO 20 STEP 2
 40 PLOT 0,y
 50 DRAW 255,0
 60 NEXT y
 70 FOR n=100 TO 220 STEP 30
 80 FOR x=-10-n/10 TO 10+n/10
 90 PLOT n,35+n/10
100 DRAW x,-n/4
110 NEXT x: NEXT n
```

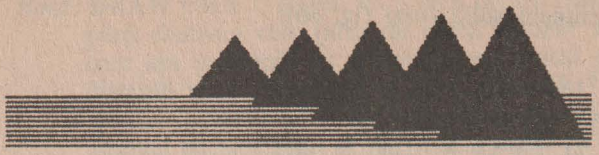


Fig. 61

Acum adăugaţi şi următoarele linii de program. Cînd veţi rula din nou programul, veţi vedea un fascicul de raze pe cerul nopţii care produce mai multe stele. Fascicolul porneşte din colţul ecranului înspre poziţia (x, y), x şi y fiind două variabile care vor lua valori aleatoare. Stelele vor fi desenate în diverse poziţii, după ce valorile x şi y au fost convertite în numere care indică poziţia stelelor în grafica de rezoluţie joasă:

```
115 REM FASCICUL DE RAZE
120 LET x=RND*255 ----------> Fasciculul de raze poate porni din
                              orice punct de pe orizontală.
130 LET y=RND*104+71 -------> Fasciculul de raze poate porni
                              dintr-un punct de pe verticală
                              situat mai sus de 71.
140 LET l=INT ((175-y)/8) --> Conversia coordonatei pe verticală
                              din grafica de înaltă rezoluţie în
                              coordonata pe verticală pentru
                              grafica de rezoluţie scăzută:
                              cînd y=71, l=13 (linia 13)
                              cînd y=174, l=0  (linia 0).
150 LET c=INT (x/8) --------> Conversia coordonatei pe orizontală:
                              cînd x=0, c=0  (coloana 0)
                              cînd x=254, c=31 (coloana 31).
160 PLOT 0,0: DRAW OVER 1;x,y
170 BEEP 0.01,x/4 ----------> O notă muzicală.
180 PLOT 0,0: DRAW OVER 1;x,y  N + 88 în modul E.
190 PRINT AT l,c;"*"          Se vor desena liniile cu culoarea
200 GO TO 120                 fondului. OVER 1 din linia 160
                              permite desenarea fasciculului de
                              raze, iar OVER 1 din linia 180
                              permite înlăturarea fasciculului
                              de raze fără să se modifice res-
                              tul desenului.
```

PROIECTAREA DESENELOR PE ECRAN

Puteţi produce cu ajutorul calculatorului orice desen sau model utilizînd fie grafica de rezoluţie joasă, fie grafica de rezoluţie înaltă, fie amîndouă. Cea mai bună metodă de a proiecta desene este de a realiza mai întîi o schiţă preliminară

pe o coală de hîrtie, pe care în prealabil s-a trasat fin o grilă care să reprezinte atît grafica de înaltă rezoluţie, cît şi pe cea de rezoluţie scăzută, şi apoi, să se scrie programul care realizează desenul utilizînd instrucţiunile: **PRINT, PRINT AT, PLOT, DRAW, CIRCLE**, precum şi caractere grafice, cicluri **FOR-NEXT**, culorile, etc.

Iată programul "**Pătrate**":

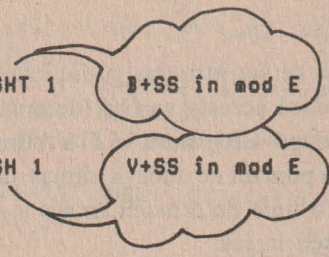
```
 5 REM ** PATRATE **
       ************
10 BORDER 0: PAPER 0: CLS
20 FOR x=7 TO 0 STEP -1 ----> Ciclu pentru schimbarea culorilor
                              şi a mărimii pătratului mare.
30 INK x
40 FOR l=11-x TO 11+x ------> Ciclu pentru modificarea liniei
                              poziţiei pătratului mic.
50 FOR c=16-x TO 16+x ------> Ciclu pentru modificarea coloanei
                              pătratului mic.
60 PRINT AT l,c;" "
70 NEXT c
80 NEXT l
90 NEXT x
```

Modificaţi linia 60, astfel încît în locul pătratului mic să puneţi o steluţă sau oricare alt caracter de pe tastatură. Rulaţi programul şi adăugaţi, apoi, linia 15:

```
15 INVERSE 1
```

Acum steluţele apar la rularea programului în negru (culoarea fondului) pe un fond alb.

Modificaţi linia 15, adăugaţi linia 16 şi rulaţi din nou programul:



BRIGHT realizează o afişare mai strălucitoare a caracterului.

FLASH realizează pulsarea poziţiei caracterului (caracter clipitor), făcînd culorile fondului şi caracterului să alterneze cu o cotă constantă.

Notă:

1. **FLASH, BRIGHT** şi **INVERSE** se pot utiliza şi în cadrul unor instrucţiuni **PRINT**, fiind separate de restul liniei prin intermediul semnului ;.

2. Cu **INVERSE** 1 puteți folosi caracterele grafice complementare corespunzătoare semnelor de pe tastele 1—8, fără a mai fi nevoie de tasta CS.

```
PRINT INVERSE 1;"□" ——→ Tasta 8 în mod 6.
```

pe ecran va apare: ■

Pentru a realiza desene pe ecran, nu este nevoie să se scrie de fiecare dată un program. În loc de acest lucru, se poate folosi un program prin intermediul căruia să se poată realiza direct, fără instrucțiuni, desenele pe ecran. Iată un program simplu care va permite acest lucru:

```
 5 REM ** PROGRAM DE DESENAT **
       ***************************
10 INPUT "CULOARE ";c ——→ Introducerea codului culorii cu
                          care se dorește să se deseneze.
20 BORDER 3: PAPER 0:
   INK c: CLS

30 PLOT 125,85 ——————→ Poziționare în centrul ecranului.
40 LET x=5
50 INPUT k$ ——————→ Se așteaptă introducerea unei
                     litere care va direcționa
                     trasarea unei linii de 5 pași
                     (pixeli).
60 IF k$="d" THEN DRAW x,0
70 IF k$="s" THEN DRAW -x,0
80 IF k$="i" THEN DRAW 0,x
90 IF k$="j" THEN DRAW 0,-x
100 GO TO 50
```

La rulare, calculatorul va aștepta (după ce s-a introdus codul culorii cu care vrem să desenăm) introducerea unei litere; dacă aceasta va fi **d** (dreapta), calculatorul va trasa o linie de 5 pixeli la dreapta (după CR), dacă va fi **s** (stînga), calculatorul va trasa o linie de 5 pixeli la stînga din punctul de unde a rămas ultima oară, dacă va fi **i** (înainte), calculatorul va trasa o linie de 5 pixeli în sus și dacă va fi **j** (jos) calculatorul va trasa o linie de 5 pixeli în jos.

Dacă se acționează o altă tastă în afara celor 4, datorită liniei 100 nu se va întîmpla nimic și se va aștepta acționarea altei taste.

Cînd desenul va fi gata, programul se va putea opri cu STOP (A + SS) și veți putea admira desenul realizat.

CALCULATORUL PICTOR

Programul anterior este însă destul de greoi: pentru fiecare trasare a unui segment de 5 pixeli trebuie acționată tasta CR. În schimb, jocul **"Pictor"** permite trasarea oricăror figuri compuse din dreapte orizontale, verticale și înclinate cu unghiuri de 45 de grade, precum și plasarea "creionului" cu care se desenează în orice loc pe ecran prin posibilitatea ridicării și coborîrii acestuia. Pentru folosirea programului nu va fi necesară acționarea tastei CR atunci cînd se va comanda o anumită acțiune calculatorului. De asemenea, este prevăzută situația în care se comandă trasarea unor linii în afara ecranului grafic; în acest caz, în situația anterioară programul se întrerupea și apărea un mesaj de eroare în care se indica depășirea marginilor ecranului.

Pentru reluarea programului (care nu ar mai fi posibilă decît de la început) ar trebui dată din nou comanda **RUN**. În cazul jocului **"Pictor"**, la atingerea marginilor ecranului grafic, calculatorul va atenționa sonor și va aștepta o altă comandă.

```
 5 REM ** PICTOR **
       ************
10 BORDER 6: PAPER 7: INK 0 ──► Inițializarea atributelor.
15 PRINT "PICTOR"
20 LET a=0 ──────────────────► a este variabila care indică
                               dacă creionul este ridicat sau nu.
                               La început creionul este ridicat
                               (a=0).
30 LET x=125 ┐
40 LET y=87  ┘ ──────────────► Poziția creionului (orizontală și
                               verticală). La început este în
                               centrul ecranului.
50 PLOT x,y ─────────────────► Un punct negru apare în centrul
                               ecranului.
60 LET c$=INKEY$ ◄──── N în mod E. ──► c$ conține ultima
                               tastă acționată.
70 IF c$="" THEN GO TO 60 ──► Dacă nu este apăsată nici o tastă
                               se așteaptă acționarea uneia.
80 IF c$="0" THEN LET a=0: ──► Dacă s-a acționat tasta 0, varia-
   GO TO 60                    bila a devine 0 (creionul sus).
90 IF c$="1" THEN LET a=1: ──► Dacă s-a acționat tasta 1, varia-
   PLOT x,y: GO TO 60          bila a devine 1 (creionul jos).
100 LET dx=0: LET dy=0 ──────► Inițializarea creșterilor pe
                               orizontală și pe verticală.
110 IF c$="5" THEN  LET dx=-1: ──► Dacă s-a acționat tasta 5,
    GO TO 200                  coordonata pe orizontală se mic-
                               șorează cu o unitate ( ← ).
120 IF c$="6" THEN LET dy=-1: ──► Acționarea tastei 6 ( ↓ ).
    GO TO 200
130 IF c$="7" THEN LET dy=1: ──► Acționarea tastei 7 ( ↑ ).
    GO TO 200
140 IF c$="8" THEN LET dx=1: ──► Acționarea tastei 8 (→).
    GO TO 200
```

In locul combinației de taste CS+5 se folosește codul ASCII al caracterului respectiv.

```
150 IF c$=CHR$ 8 THEN LET dx=-1:      Acționarea tastelor
    LET dy=1: GO TO 200               5+CS (←).
160 IF c$=CHR$ 10 THEN LET dx=-1:     Acționarea tastelor
    LET dy=-1: GO TO 200              6+CS (↓).
170 IF c$=CHR$ 11 THEN LET dx=1:      Acționarea tastelor
    LET dy=-1: GO TO 200              7+CS (↑).
180 IF c$=CHR$ 9 THEN LET dx=1        Acționarea tastelor
    LET dy=1: GO TO 200                     8+CS (→).
190 GO TO 60                          Acționarea unei taste
                                      fără efect.

200 IF a=0 THEN PLOT INVERSE 1;
    x,y                               Cazul creionului ridicat.
210 LET x=x+dx                        Viitoarea poziție orizontală.
220 IF x>255 OR x<0 THEN              Mișcările creionului pentru
    BEEP .2,10: BEEP .2,5:            poziții în afara limitelor ori-
    LET x=x-dx                        zontale sînt anulate.
230 LET y=y+dy                        Viitoarea poziție verticală.
240 IF y>175 OR y<0 THEN              Mișcările creionului pentru
    BEEP .2,10: BEEP .2,5:            poziții în afara limitelor
    LET y=y-dy                        verticale sînt anulate.
250 PLOT x,y                          Marcarea noului punct.
260 GO TO 60
```

INKEY$ este foarte des folosit în jocurile pe calculator. La întîlnirea acestui cuvînt cheie, calculatorul citește tastatura pentru a „controla" dacă o tastă a fost acționată. Dacă nu este acționată nici o tastă, calculatorul „așteaptă" în continuare; dacă o anumită tastă este acționată, atunci calculatorul va lua o anumită decizie. Vă dați seama de ce INKEY$ este foarte important pentru jocuri? Deoarece calculatorul va lua o decizie imediat ce o tastă a fost acționată, fără a mai fi necesară acționarea tastei CR.

La comanda **RUN**, un punct negru apare în centrul ecranului: "creionul" este pregătit și puteți realiza desene pe ecran acționînd următoarele taste:

5 — deplasare stînga (←)
6 — deplasare jos (↓)
7 — deplasare sus (↑)
8 — deplasare dreapta (→)
5 + CS — deplasare stînga sus (↖)
6 + CS — deplasare stînga jos (↙)
7 + CS — deplasare dreapta jos (↘)
8 + CS — deplasare dreapta sus (↗)
0 — ridicare "creion"
1 — coborîre "creion"

"Creionul" este inițial în centrul ecranului.

Observați că nu este nevoie să acționați **CR** după ce acționați una din aceste comenzi. O apăsare scurtă a unei taste, va avea ca urmare înaintarea cu un pixel a creionului în direcția indicată.

Ținînd mai mult timp apăsată tasta care indică deplasarea, se vor putea obține linii de orice lungime (dar ele nu vor depăși ecranul grafic). Dacă doriți să începeți o parte a unui desen în alt loc pe ecran decît locul în care se află "creionul", veți acționa tasta **0** (acum nu va mai fi necesar să țineți mai mult timp apăsată această tastă), și, fără să acționați **CR**, veți da în continuare comenzile necesare pentru aducerea "creionului" în locul unde doriți începerea desenului. Acum nu aveți altceva de făcut decît să acționați tasta **1** pentru coborîrea "creionului" și să dați în continuare comenzile pentru desen.

23. A. *Folosiți-vă de jocul „Pictor" și încercați să realizați cu ajutorul lui un desen, la alegere, pe ecran. Ce părere aveți de mașina din fig. 62?*

B. *Încercați să generalizați programul pentru trasări pe alte direcții (alte unghiuri), trasări de linii curbe, trasări cu culori și pe fonduri de culori diferite.*

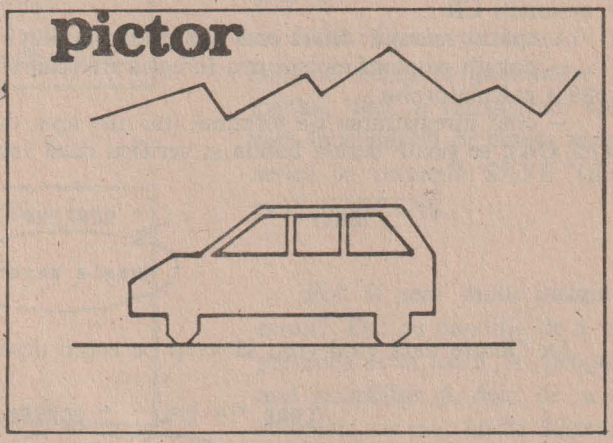


Fig. 62

ECRANE GRAFICE

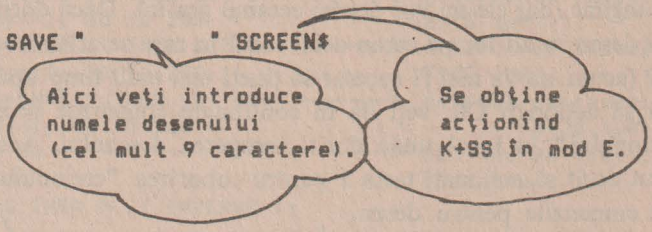
Ați realizat un desen, care vă place, cu jocul "**Pictor**"? Cu siguranță că doriți să-l păstrați cumva, la fel cum păstrați un desen pe hîrtie pentru a-l putea privi și altă dată, sau pentru a-l completa mai tîrziu și a-l face și mai frumos.

De fapt, puteți realiza acest lucru cu ajutorul calculatorului, adică puteți să înregistrați (salvați imaginea grafică pe caseta magnetică și, apoi, să o încărcați apărînd astfel din nou pe ecran) de fiecare dată cînd doriți acest lucru, la fel cum încărcați un program obișnuit de pe caseta magnetică.

Pentru aceasta veţi proceda în felul următor:

— opriţi rularea programului "**Pictor**". Pentru acest lucru veţi acţiona împreună tastele **SPACE** şi **CS**, obţinînd **BREAK** (întreruperea programului);

— introduceţi apoi următoarea comandă:

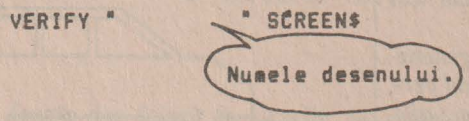


şi acţionaţi **CR**.

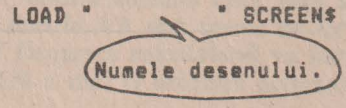
A apărut mesajul: **Start cass and press any key?** În acest caz:

— porniţi casetonfonul pentru înregistrare (clapa **RECORD**) şi acţionaţi orice tastă a calculatorului;.

— cînd înregistrarea s-a terminat (nu mai apar dungi orizontale şi apare mesajul **OK**), se poate derula banda şi verifica dacă înregistrarea a fost corectă:



De fiecare dată cînd vreţi să aveţi pe ecran desenul dorit, veţi da comanda:



veţi potrivi caseta pe care a fost salvat desenul respectiv şi veţi da drumul la casetofon (pe **PLAY**).

Desenul se conturează pe ecran, parte cu parte, pe măsură ce banda se derulează, adică pe măsură ce informaţia este citită.

STRUCTURA UNUI JOC CU ECRAN GRAFIC DE PREZENTARE

Sîntem convinşi că v-aţi gîndit imediat cum puteţi să integraţi într-un joc o imagine de acest fel, de exemplu: o imagine în care să prezentaţi grafic (cu desene şi text) jocul respectiv, după care să înceapă jocul propriu-zis. Acest lucru se poate

realiza asigurîndu-se jocului o structură din mai multe înregistrări (să numim de acum înainte orice înregistrare pe caseta magnetică fie ea program, imagine-ecran sau date, fișier). Această structură va arăta ca în fig. 63.

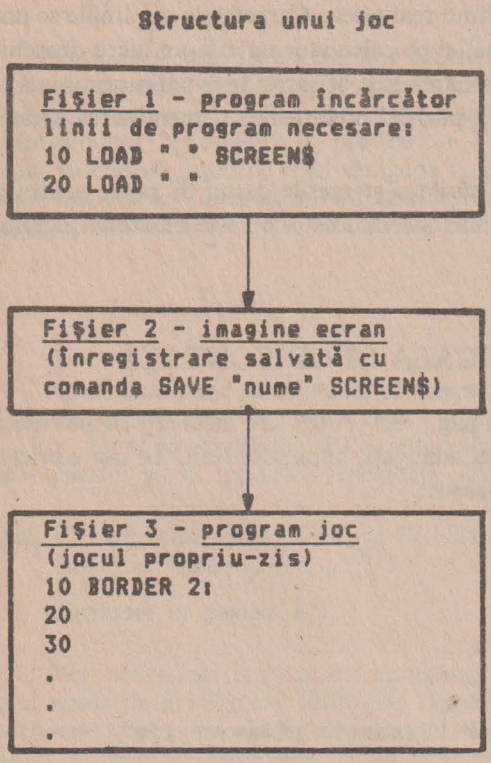


Fig. 63

— primul fișier va fi un program scurt care va asigura încărcarea următoarelor fișiere ale jocului de pe caseta magnetică (imagini-ecran sau programe). Să-l numim **program încărcător**. El va putea conține și informațiile referitoare la culorile folosite pentru desene sau texte, pentru fond sau margine etc.

— al doilea fișier va putea fi o **imagine ecran de prezentare** a jocului, deci, o înregistrare salvată în prealabil pe caseta magnetică cu comanda SAVE "(nume)" SCREEN$.

Pot fi mai multe imagini-ecran? Da, cu condiția de a se prevedea acest lucru în programul încărcător și, deci, de a se completa cu mai multe instrucțiuni de încărcare imagini-ecran, dar trebuie ținut seama de faptul că încărcarea unei imagini-ecran durează ceva mai mult de 20 de secunde și poate fi plictisitor pentru jucător să aștepte foarte mult timp pentru derularea cîtorva imagini.

— următorul fișier este chiar **programul de joc**. În timpul în care are loc încărcarea acestui program, imaginea (de prezentare a jocului) rămîne pe ecran, fiind deci la dispoziția jucătorului, astfel încît acesta se informează în legătură cu regulile jocului (de exemplu); în acest fel timpul consumat cu încărcarea jocului va fi petrecut cu folos, iar după încărcare, jocul va putea începe imediat.

Pentru ca jocul să înceapă imediat și fără a mai fi necesară introducerea comenzii **RUN**, programul de joc va trebui salvat în prealabil cu comanda **SAVE "(nume)" LINE 10** prin care se va realiza automat după încărcare **RUN 10**.

ANIMAŢIA

Vă plac desenele animate?

Un joc pe calculator va fi cu atît mai plăcut, cu cît desenele realizate vor prinde viaţă pe ecran, prin mişcare imitînd realitatea. Caracterele sau liniile se pot mişca pe ecran şi astfel producerea animaţiei pe calculator nu este un lucru deosebit de dificil. Tot ce trebuie făcut este desenarea sau afişarea în poziţia apropiată a liniei sau caracterului (neuitînd ştergerea poziţiei precedente) şi menţinerea acestei "mişcări" pe o traiectorie.

Calculatorul va realiza desenele, afişările şi ştergerile destul de rapid pentru a da senzaţia unei mişcări. Cea mai bună metodă în acest scop este utilizarea ciclului FOR-NEXT.

MIŞCAREA PE VERTICALĂ ŞI ORIZONTALĂ

Reluaţi programul "**Păianjen**" de la pag. 92. Puteţi să-l încărcaţi de pe caseta magnetică dacă l-aţi salvat anterior şi să adăugaţi, apoi, alte linii. În caz contrar, tastaţi încă o dată programul de la început:

```
  2 REM ** ANIMATIE **
        **************
  3 REM PAIANJEN
  5 BORDER 3: PAPER 5: CLS
 10 FOR x=0 TO 7
 20 READ y
 30 POKE USR "p"+x,y ----------> Caracter păianjen ("p").
 40 NEXT x
 50 DATA 60,126,219,255,189,165,
    165,36
 60 FOR x=0 TO 7
 70 READ y
 80 POKE USR "f"+x,y ----------> Caracter fir ("f").
 90 NEXT x
100 DATA 16,16,16,16,16,16,16,16
110 FOR i=0 TO 20
120 PRINT AT 1,3; INK 0;"f"      Tasta F în mod G.

                                 Caracterul va fi afişat aşa
                                 după ce aţi rulat o dată
                                 programul.

130 PRINT AT 1+i,3; INK 2;"p"    Tasta P în mod G.
```

```
140 NEXT 1 ──────────► Animația: cu un ciclu FOR-NEXT
                       caracterul "păianjen" se înlo-
                       cuiește cu caracterul "fir",iar
                       pe poziția actuală se înscrie
                       caracterul "păianjen".
```

Prin intermediul instrucțiunilor dintre liniile 110 și 140, de fiecare dacă cînd ciclul se repetă, "firul" de păianjen se va lungi, adăugîndu-se un nou caracter "|" peste caracterul "păianjen", iar acesta va înainta o poziție în jos prin înscrierea unui caracter "păianjen" în poziția 1 + 1 (linia următoare). În acest fel, păianjenul va coborî rapid pe firul său pînă cînd va atinge linia 21 din partea de joc a ecranului grafic. Calculatorul va calcula foarte rapid noua poziție, astfel încît păianjenul va cădea foarte rapid. Pentru a încetini acțiunea, inserați linia:

```
135 PAUSE 10
```

Aceasta va pune calculatorul să aștepte o cincime de secundă (să facă **pauză** de 1/5 dintr-o secundă), de fiecare dată înainte de afișarea păianjénului pe următoarea poziție. Sigur că modificînd 10 cu alte valori, veți observa cum viteza de cădere a păianjenului se va modifica. Încercați, deci, cîteva valori noi!

24. *Ce număr va trebui pus după* PAUSE *pentru ca păianjenul să aștepte o secundă la capătul fiecărui fir?*

Răspunsul la pagina 142.

Veți vedea mai tîrziu că altă modalitate de a încetini (sau a temporiza) mișcarea este aceea de a intercala între cele două momente succesive ale mișcării o altă activitate pentru calculator, de exemplu: aceea de a produce o notă muzicală de o anumită durată. Într-adevăr, la un moment dat, calculatorul, nu poate să execute decît un singur lucru și de aceea prin producerea unei note muzicale se va realiza implicit și încetinirea mișcării. În plus, aceasta se va desfășura pe un fond muzical. Adăugați următoarele linii programului "**Animație**" și rulați-l din nou. Veți vedea din nou păianjeni în mișcare, dar acum mișcarea se va face și în altă direcție:

```
150 FOR c=3 TO 30
160 PRINT AT 21,c;" " ──────► Stergerea păianjenului
                              de pe vechea poziție.

170 PRINT AT 21,c+1; INK 2; "Ⓜ" ► Mutarea păianjenului pe
                                   noua poziție.
180 NEXT c
                              Mișcarea păianjenului
                              pe orizontală.
```

Acum păianjenii fug într-o parte (mișcarea orizontală), imediat ce ating partea de jos a ecranului grafic. Noile linii adăugate (cu un alt ciclu **FOR-NEXT**) vor provoca afișarea pe rînd a caracterului "păianjen" pe o poziție **c + 1**, adică pe coloana din dreapta față de coloana pe care a fost afișat anterior un caracter "păianjen". Observați că mai întîi este afișat un spațiu gol (linia de program 160) și, apoi, păianjenul este afișat pe poziția de pe coloana următoare. Aceasta va face ca păianjenul să dispară de pe o poziție și să apară pe următoarea, mișcîndu-se la dreapta.

Este totodată de preferat, pentru realizarea animației, să se șteargă caracterul de pe vechea poziție înainte de afișarea lui pe poziția următoare,

În același fel cum s-a realizat mișcarea pe verticală și pe orizontală se pot realiza mișcări și în alte direcții. De exemplu: simularea mișcării pe diagonală (spre direcția dreapta jos) a caracterului "păianjen" se poate realiza prin intermediul următoarelor linii de program (și ștergerea liniilor 170 și 180):

```
110 LET c=0
120 FOR l=0 TO 20
130 PRINT AT l,c;" " ----------> Stergerea păianjenului
                                 de pe vechea poziție.
140 LET c=c+1
150 PRINT AT l+1,c;"🕷" ------> Mutarea păianjenului pe
                                 noua poziție, a cărei
                                 linie și coloană este,
                                 fiecare, mai mare cu o
                                 unitate (poziția este
                                 pe diagonală față de
                                 cea precedentă).
160 NEXT l
```

La rularea programului veți vedea cum păianjenul se va deplasa pe diagonală.

Acum știm cum se poate realiza animația pentru un caracter. Dar cum se poate realiza animația, adică mișcarea, unor obiecte desenate pe ecran și care sînt formate din mai multe caractere grafice, linii, puncte etc.?

Simplu! Se șterge întreg desenul și se redesenează pe poziția apropiată, păstrîndu-se o anumită direcție de deplasare.

Să presupunem că dorim să deplasăm în jos "gîndacul" realizat cu ajutorul a 4 caractere grafice (pag. 97).

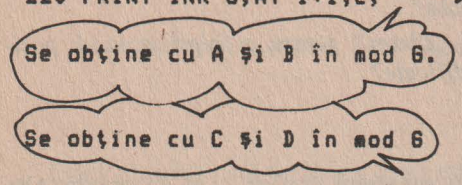
Considerăm că aceste caractere au fost deja definite și sînt corespunzătoare tastelor **A, B, C** și **D**. Pentru a realiza căderea "gîndacului" din partea de sus a ecranului în partea de jos, vom adăuga următoarele linii de program:

```
190 LET c=INT (RND*31)
200 FOR l=0 TO 19
210 PRINT AT l,c;"  "
220 PRINT INK 6;AT l+1,c;"  "
230 PRINT INK 6;AT l+2,c;"  "
240 NEXT l
250 GO TO 190
```

Se stabilește coloana pe care se va deplasa gîndacul.

Șterge partea de sus a gîndacului.

Mută parea de sus a gîndacului pe linia de mai jos (de aceea nici nu mai este necesară ștergerea părții de jos a gîndacului, întrucît în locul acesteia se afișează partea de sus).



Observați cum la rularea programului, "gîndacul" se va deplasa rapid în jos, cu toate că acum calculatorul va avea de șters și de afișat mai multe caractere.

25. *Realizați un program care să deseneze din linii, puncte și caractere grafice un elicopter. Încercați să asigurați deplasarea acestuia înainte, sus, jos și înapoi, cu ajutorul a patru taste corespunzătoare (veți folosi, bineînțeles,* INKEY$*). Veți avea grijă ca, în cazul în care elicopterul depășește marginile ecranului grafic, aparatul vostru să rămînă pe loc.*

Un răspuns posibil îl puteți găsi la pag. 142.

O posibilitate mai eficientă (și mai elegantă) de a realiza mișcarea unui punct, caracter, obiect sau "spiriduș" pe direcțiile de bază (sus, jos, stînga, dreapta), este aceea care ține cont de faptul că o **expresie logică adevărată** are valoarea **1**, iar o **expresie logică falsă** are valoarea 0. De exemplu: cu **PRINT 2 * 3 = 6** vom obține valoarea 1, expresia fiind adevărată, dar cu **PRINT 1 = 2** vom obține valoarea 0, expresia fiind falsă. Pentru realizarea mișcării pe orizontală la stînga (cu tasta **S**) și la dreapta (cu tasta **D**) se poate scrie următoarea linie de program.

```
LET X=X+P*((INKEY$="D")-(INKEY$="S"))
```

în care **X** este valoarea inițială a coordonatei pe orizontală, iar **P** este pasul cu care se dorește deplasarea.

Dacă se acționează tasta **D**, atunci **INKEY$** = "**D**" este adevărată (are valoarea **1**), iar coordonata **X** va deveni $X + P \times (1 - 0)$, deci, va fi mai mare cu mărimea pasului, realizîndu-se deplasarea spre dreapta; invers, dacă se acționează tasta **S** atunci **INKEY$** = "**S**" este adevărată (are valoarea 1) și coordonata **X** va deveni $X + P \times (0 - 1) = X - P$ deci va fi mai mică cu mărimea pasului, realizîndu-se deplasarea spre stînga.

Pentru mişcarea pe verticală, se va folosi în mod similar linia:

```
LET Y=Y+P*((INKEY$="N")-(INKEY$="J"))
```

26. *A. De ce spunem că aceasta reprezintă o metodă mai eficientă de realizare a mişcării pe orizontală sau pe verticală?*

B. Ce linii trebuie modificate în jocul „Pictor" pentru a folosi această posibilitate? Cum vor arăta liniile modificate?

Răspunsurile la pagina 143.

Animaţia se poate realiza nu numai cu caractere grafice, ci şi cu oricare alte caractere (deci şi litere). La ce poate folosi acest lucru în jocuri? Bineînţeles, la prezentarea mai frumoasă a titlului unui joc, a textului (instrucţiunile de utilizare) sau a diverselor mesaje care se afişează în timpul jocului. Iată un exemplu de subrutină care se poate apela din programul principal, în scopul afişării unui text (în cazul de faţă titlul jocului şi tastele de folosit), astfel încît litera care urmează să fie afişată să se plimbe pe ecran şi, apoi, să fie adusă la locul potrivit:

```
  5 REM ** JOC CU PREZENTARE **
        **********************
 15 LET a$="PICTOR ──────────► In variabila a$ se memorează
           TASTE:              mesajul care va fi afişat.
           5 - STINGA, 6 - SUS, 7 - JOS,
           8 - DREAPTA, CS - DIAGONALE"
 16 GO SUB 1000 ─────────────► Apelarea subrutinei de afişare.
 17 REM PROGRAM PRINCIPAL
999 STOP
1000 FOR x=1 TO LEN a$
1010 PRINT AT 0,0;a$( TO x) ──► Afişarea cîte unei litere.
1020 NEXT x
1030 RETURN
```

Încercaţi să introduceţi această subrutină în jocul "**Pictor**" de la pagina 105

FOLOSIREA ŢINTEI

În jocuri, deseori atingerea unui obiect ţintă (avion, vapor, navă, planetă, etc.) de către alt obiect în mişcare trebuie urmată de o anumită acţiune. Dar cum va şti calculatorul cînd a avut loc o ciocnire sau o explozie?

Detectarea unei coliziuni nu este dificilă. Dacă două caractere sînt afişate în poziţia **l, c** (pentru linie şi coloană) şi respectiv **v, h** (pentru vertical şi orizontal) şi dacă **l** = **v** şi **c** = **h**, atunci caracterele vor fi în aceeaşi poziţie. Cu alte cuvinte, **l**,

adică linia pe care se află primul caracter, este identic cu **v**, adică linia pe care se află al doilea caracter; iar **c**, adică coloana primului caracter, este identic cu **h**, adică coloana celui de-al doilea caracter. Puteți scrie, de exemplu, acest lucru ca o linie de program:

```
160 IF l=v AND c=h THEN PRINT AT l,c;"BUUMMM!"
```

Dacă obiectul este un desen (mai mare decît un singur caracter grafic), atunci se va testa condiția de atingere a celor două obiecte.

De exemplu, să presupunem că vrem să testăm dacă "gîndacul" (desenat prin cele 4 caractere **A, B, C** și **D** și care se deplasează în jos conform programului de la pag. 113) intră într-un "coș" din partea de jos a ecranului, în dreptul liniei 2 și a coloanei 5. Adăugăm programului următoarea linie:

```
235 IF l=18 AND c=5 THEN GO SUB 270
```

iar la 270, va fi subrutina care va arăta ce trebuie să facă calculatorul dacă "gîndacul" a intrat în "coș".

Altă metodă de a testa coliziunea este aceea care utilizează culorile.

Puteți șterge programul din memoria calculatorului folosind **NEW**, însă caracterele grafice definite anterior (cele corespunzătoare tastelor **P** pentru "păianjen", **A, B, C, D** pentru "gîndac" etc.) vor rămîne în memoria calculatorului pînă la o resetare completă, folosind, de exemplu, butonul de oprire și pornire. Acum puteți combina alte programe cu utilizarea caracterelor grafice pe care le-ați definit anterior. Să presupunem că veți retasta sau încărca de pe caseta magnetică programul **"Piramide cu fascicol de raze"** de la pag. 102. Apoi, adăugați și următoarele linii de program, ștergînd linia 190.

```
1 REM ** PAIANJENI SI PIRAMIDE **
      ****************************
5 FOR x=0 TO 7 ──────────► Se creează un caracter grafic
6 READ y                   utilizator corespunzător tastei E
7 POKE USR "e"+x,y         care va sugera o explozie.
8 NEXT x
9 DATA 145,82,44,121,158,52,
  74,137
    .
    .                  ──► Programul PIRAMIDE
    .                      (liniile 10 - 110)
```

Inserați aceste linii în programul PIRAMIDE și ștergeți linia 190.

```
114 LET h=RND*31
```

Păianjenii cad din diverse locuri din partea de sus a ecranului (coloana h poate lua diverse valori).

```
115 FOR v=0 TO 20
117 PRINT AT v,h;" ",AT v+1,h;
    INK 4;"🕷"
    . Apare fasciculul
    . de raze în timp ce
    . păianjenul se
    . deplasează în jos.
200 NEXT v
```

Mișcarea păianjenului în jos. Se șterge de pe o poziție și se pune pe poziția următoare (linia următoare și aceeași coloană). Cînd ajunge la piramide începe să le "mănînce" (le șterge).

```
205 PRINT AT 21,h; FLASH 1;
    INK 2; PAPER 6;"🕷"
```

Cînd ajunge pe ultima linie, păianjenul devine clipitor și își schimbă culoarea.

```
210 GO TO 114
```

Își începe mișcarea alt păianjen.

Rulați programul! Acum nu mai apar stele pe cer; în schimb din cer cad din diverse locuri păianjeni care "mănîncă" piramidele.

Adăugați acum și aceste linii:

```
189 REM PAIANJENII EXPLODEAZA
```

ATTR depistează "atributele" unei anumite poziții (linie, coloană) de pe ecran, adică culoarea (INK), fondul (PAPER), luminozitatea (BRIGHT) și clipirea (FLASH) poziției respective.
In programul PAIANJENI SI PIRAMIDE, ATTR va face ca păianjenul să explodeze cînd este galben. Dacă pentru un păianjen ATTR nu este niciodată egal cu 14, atunci el începe să "mănînce" piramide și se oprește pe ultima linie.

Se obține cu L+SS în mod E

```
190 IF ATTR (v+1,h)=14
    THEN GO TO 500
```

```
500 PRINT AT v+1,h; FLASH 1;
    PAPER 2;"✳"
```

Explozie.

Tasta E în mod G.

```
510 PAUSE 100 ——————→ Explozia se menține circa 2
                       secunde, după care își începe
                       mișcarea alt păianjen.
520 GO TO 114 ——————→ Alt păianjen.
```

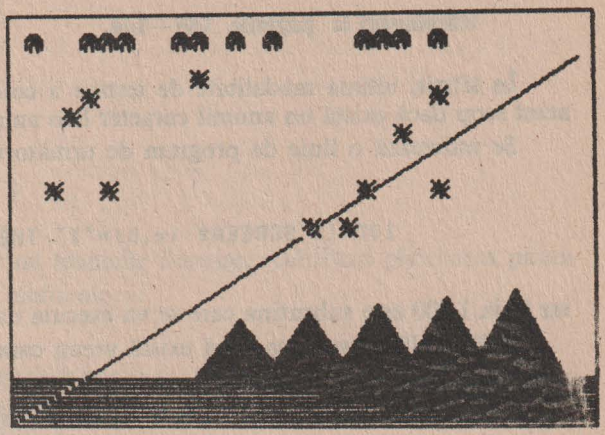


Fig. 64

Cînd fasciculul luminos loveşte păianjenii, aceştia devin galbeni. Cînd linia realizată cu **DRAW** atinge poziţia caracterului "păianjeni", acesta va "lua" aceeaşi culoare cu linia (galben) (vezi fig. 64). În linia 190, se testează care sînt atributele caracterului de pe poziţia determinată de linia **v + 1** şi coloana **h.**

Dacă acestea sînt:	Cod	Valoarea cu care se înmulţeşte		Valoarea obţinută
— culoare (INK) galben (cod 6)	6 ×	1	=	6
— culoare fond (PAPER) albastru (cod 1)	1 ×	8	=	8
— luminozitate (BRIGHT) nu are (cod 0)	0 ×	64	=	0
— clipire (FLASH) nu are (cod 0)	0 ×	128	=	0
	Total		=	14

atunci **ATTR (v + 1, h)** va fi 14 (totalul codurilor atributelor înmulţite cu valoarile 1, 8, 64 şi, respectiv, 128) şi se va executa linia 500 care va simula explozia păianjenului.

Pentru orice combinaţie a atributelor unui caracter, există o valoare unică a lui **ATTR**.

27. *A. Indicați care sînt atributele caracterului de la linia* 10 *și coloana* 10 *pentru care* **ATTR** *are valoarea a)* 98; *b)* 168; *c)* 249.

Realizați schema logică a determinării atributelor caracterului atunci cînd se cunoaște valoarea lui ATTR.

Răspunsuri la paginile 143—144.

În sfîrșit, ultima modalitate de testare a coliziunii este aceea care verifică în acest scop dacă există un anumit caracter la o anumită poziție caracter de pe ecran.

Se utilizează o linie de program de următorul tip:

```
100 IF SCREEN$ (v,h)="X" THEN GO SUB 1000
```

iar linia 1 000 este subrutina care se va executa dacă s-a produs coliziunea.

SCREEN$ va verifica dacă există vreun caracter X la poziția caracter, determinată de linia **v** și coloana **h**.

Observați că această metodă este asemănătoare, într-un fel, cu prima reducînd, însă, multe din verificările (linii de programe cu IF) coordonatelor necesare în astfel de ocazii.

CUM SE POT DEPLASA „SPIRIDUȘII" ÎNTR-UN DECOR

Pînă acum am deplasat "spiriduși" pe ecran afișîndu-i pe noua poziție și ștergîndu-i de pe poziția precedentă punînd în locul lor un spațiu gol. Dar dacă "spiriduș" se "plimbă" într-un decor, metoda folosită va avea ca urmare și deteriorarea decorului, ștergerea unor părți din el. Cum vom putea realiza, deci, mișcarea "spiriduș-ilor" fără să stricăm decorul? Acest lucru se realizează cu instrucțiunea **PRINT OVER 1**, care permite obținerea unei deplasări "transparente" a unui "spiriduș", lăsînd decorul intact. Pentru a vă convinge tastați:

```
PRINT AT 10,10;"■"          și apoi

PRINT AT 10,10; OVER 1;"*"
```

Veți vedea apariția steluței în pătratul negru afișat anterior.

Notă. Dacă veți testa acum din nou **PRINT AT 10,10; OVER 1;** *"*" steluța va dispărea, pătrățelul negru devenind intact. Deci, pentru a realiza o deplasare "transparentă", în loc să ștergeți "spiriduș", îl veți mai afișa o dată.*

Atenție! Instrucțiunea **PRINT OVER 1** nu păstrează, însă, atributele (culoarea, sclipirea, strălucirea) pozițiilor traversate. În acest scop, se pot folosi următoarele instrucțiuni care permit obținerea unei transparențe totale:

```
PAPER 8

INK 8

FLASH 8

BRIGHT 8
```

Iată un program care, utilizînd tehnicile descrise, realizează plimbarea păianjenului într-un decor de pătrate multicolore:

```
  5 REM ** PLIMBARE TRANSPARENTA **
       ****************************
 10 BORDER 0: PAPER 0: INK 4
 20 CLS
 30 LET X=0: LET Y=0: GO SUB 1000 ----> Se stabilește locul
                                        de pornire a păian-
                                        jenului.
 40 PRINT AT Y,X; OVER 1; INK 8;"🕷"
 50 LET HX=X: LET HY=Y
 60 LET a$=INKEY$: IF a$="" THEN
    GO TO 60
 70 LET X=X+(a$="8")-(a$="5")  ----> Tastele cu care se va
 80 LET Y=Y+(a$="6")-(a$="7")        mișca păianjenul: 8 la
                                     dreapta, 5 la stînga,
                                     6 în jos și 7 în sus.
 90 PRINT AT HY,HX; OVER 1; INK 8;"🕷"
100 GO TO 40
1000 REM PAIANJEN SI DECOR
1010 FOR i=0 TO 7
1020 READ a
1030 POKE USR "p"+i,a
1040 NEXT i
1050 DATA 60,126,219,255,185,165,165,
     36
1060 REM DECOR
1070 FOR i=1 TO 10
1080 PRINT AT INT (RND*22),
     INT (RND*31); INK RND*6+1;"■"
1090 NEXT i
2000 RETURN
```

În astfel de jocuri este important să se vadă înainte de a se efectua mişcarea "spiriduşului", dacă nu cumva, se depăşesc marginile ecranului. Dacă răspunsul este afirmativ, programul se va întrerupe cu un mesaj de eroare.

Iată cum trebuie completate liniile 70 şi 80 ale programului "**Plimbare transparentă**" pentru ca modificarea valorilor pentru linie şi coloană să se facă numai după ce s-a testat şi valabilitatea acestora, (linia **HY** trebuie să fie mai mare ca **0** şi mai mică decît **21**, iar coloana **HX** mai mare ca **0** şi mai mică decît **31**):

```
70 LET X=X+(a$="8" AND X<31)-
   (a$="5" AND X>0)
80 LET Y=Y+(a$="6" AND Y<21)-
   (a$="7" AND Y>0)
```

IV. Cum se pot realiza efecte sonore şi muzica cu calculatorul

Calculatorul posedă un sintetizator de sunete cu ajutorul căruia jocurile pot deveni mult mai antrenante, prin intercalarea în desfăşurarea lor a diverselor efecte sonore şi secvenţe muzicale. Vă puteţi imagina la ce se pot folosi efectele sonore în jocuri? La simularea sonoră a funcţionării unor mecanisme (tren, avion, elicopter, diverse tipuri de maşini), decolarea navelor spaţiale, coliziunea unor obiecte (păcănituri, explozii), vîjîitul proiectilelor, zgomotul de paşi, etc. Secvenţele muzicale se pot folosi la prezentarea jocurilor şi în perioadele în care este indicat să se facă scurte pauze.

În jocurile pe bază de întrebare şi răspuns este bine ca un răspuns corect să fie însoţit de o scurtă (dar plăcută) secvenţă muzicală (care poate fi oferită şi drept recompensă), în timp ce, un răspuns incorect să fie însoţit de o scurtă secvenţă care să atragă atenţia asupra greşelii.

Deşi calculatorul nu poate realiza într-un anumit moment decît un singur lucru, faptul că posedă o viteză deosebit de mare va da senzaţia producerii efectelor sonore şi a secvenţelor muzicale în acelaşi timp cu secvenţele grafice care se desfăşoară pe ecran.

Producerea muzicii cu ajutorul calculatorului nu este complicată necesitînd numai cunoştinţe sumare despre sunetele muzicale.

PROGRAMAREA SUNETELOR

Pentru producerea sunetelor prin intermediul calculatorului, se utilizează un singur cuvînt: **BEEP**. Acest cuvînt, care sugerează un sunet scurt (bip), se poate realiza prin acţionarea tastelor **Z** şi **SS** în modul [E].Cuvîntul **BEEP** va fi urmat de

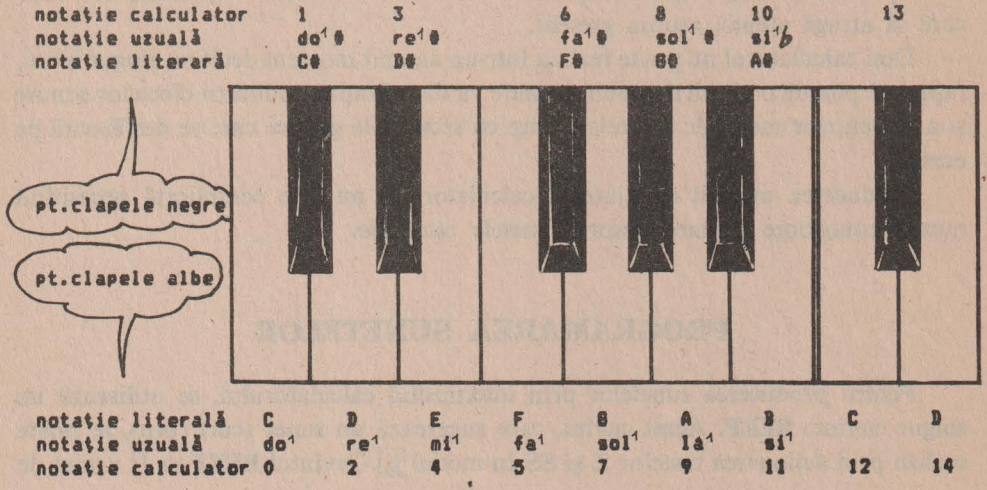
două numere (sau expresii numerice sau variabile care reprezintă numere). Prima valoare numerică va reprezenta durata sunetului (în secunde), iar a doua va reprezenta înălțimea, a cărei mărime depinde de frecvența sunetului. Următorul program demonstrează, practic, durata sunetului la comanda **BEEP**:

```
100 REM ** DEMONSTRATIE PENTRU DURATA **
        *  SUNETULUI LA BEEP           *
        ********************************
110 LET d=0.01
120 FOR n=1 TO 10
130 PRINT AT 1,1;"Durata=";d;" secunde"
140 BEEP d,0
150 PAUSE 25
160 LET d=d*2
170 NEXT n
```

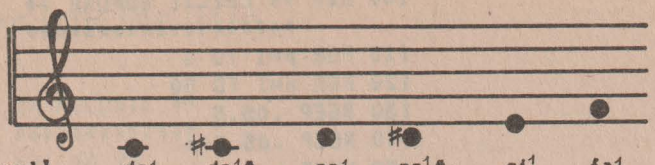
După cum se știe, în muzică, distanța cea mai mică dintre două sunete alăturate este semitonul. Ordonînd sunetele muzicale după înălțime vom obține așa numita *scară muzicală* (vezi fig. 65), în cadrul căreia doar două perechi de sunete alăturate se află la distanța de un semiton. În figura 65 se observă atît pe claviatura unui pian, cît și pe portativ valorile pe care le ia parametrul **p** (al doilea număr al comenzii **BEEP**) pentru a se obține diferitele note muzicale din cadrul octavei centrale (octava 1). Se pot obține și note mai joase sau mai înalte aparținînd altor octave (octava mică, octava a 2-a, octava a 3-a, etc), parametrul **p** putînd lua valori înregi cuprinse între-60 și 69.

Scara muzicală

Fig. 65



Corespondența pe portative:



notație uzuală	do^1	do^1#	re^1	re^1#	mi^1	fa^1
	(do central)					
notație literală	C	C#	D	D#	E	F
notație calculator	0	1	2	3	4	5

(Indicele 1 reprezintă octava 1 sau octava centrală).

Iată un program care demonstrează toate posibilitățile calculatorului de a produce sunete de diverse înălțimi:

```
100 REM ** DEMONSTRATIE PENTRU INALTIMEA **
        *  SUNETULUI LA BEEP              *
        ************************************
110 FOR p=-60 TO 69
120 PRINT AT 1,1;"Inaltime=";p
130 BEEP 0.5,p
140 PAUSE 12
150 PRINT AT 1,10;"     "
160 NEXT p
```

EFECTE SONORE

Cu **BEEP** se pot produce efecte sonore. Cîteodată însă va trebui să se efectueze mai multe încercări pentru a descoperi succesiunea de sunete potrivită. În programul următor, înălțimea sunetului este urcată și coborîtă în mod regulat, obținîndu-se un sunet tipic de sirenă:

```
100 REM ** SIRENA **
        ************
110 FOR p=5 TO 15
120 BEEP .01,p
130 NEXT p
140 FOR p=15 TO 5 STEP -1
150 BEEP .01,p
160 NEXT p
170 GO TO 110
```

Altă posibilitate de a realiza efecte sonore cu comanda **BEEP** se referă la repetarea unei succesiuni de sunete a căror înălțime diferă foarte puțin una de alta. În acest scop se pot utiliza mai multe cicluri **FOR-NEXT** ca în programul următor:

```
100 REM ** EFECTE SONORE **
        ********************
110 FOR p=1 TO 4
120 FOR n=1 TO 50
130 BEEP .05,5
140 BEEP .05,5+p
150 NEXT n
160 PAUSE 10
170 NEXT p
```

MUZICĂ CU CALCULATORUL

Deoarece poate produce o mare varietate de sunete, calculatorul poate fi folosit la interpretarea unor partituri muzicale. În programarea secvențelor muzicale, se poate urmări o linie melodică, precum și un anumit ritm. Din păcate, însă, cu calculatorul avut la dispoziție nu se pot produce decît secvențe muzicale pe o singură voce (monodie), deoarece, după cum am mai arătat, într-un moment, calculatorul nu poate realiza decît un singur lucru, neputînd produce în același timp două sau mai multe sunete (mai multe voci). De asemenea, toate notele vor fi produse în același mod, neexistînd posibilitatea ca unele să fie mai puternice (forte) sau mai slabe (pianissimo).

Programul următor demonstrează cum se poate modifica ritmul (tempoul) într-o secvență muzicală, prin modificarea parametrului durată (**d**). Programul va produce gama DO cîntată, însă, din ce în ce mai repede.

```
100 REM ** SCARA CU MODIFICAREA **
        *  TEMPOULUI             *
        ***************************
110 FOR d=0.25 TO 0 STEP -0.02
120 RESTORE
130 FOR n=1 TO 8
140 READ p
150 BEEP d,p
160 NEXT n
170 PAUSE 25
180 NEXT d
190 DATA 0,2,4,5,7,9,11,12
```

Se folosește pentru a reciti datele din linia DATA.

În general, considerăm sunetele dintr-o melodie note muzicale. În acest fel vom putea introduce printr-un program orice partitură muzicală cu condiția existenței unei singure voci.

Iată şi o scurtă melodie realizată pe calculator: se observă preferinţa de a grup. toate înălţimile notelor într-o linie DATA.

```
100 REM ** MELODIE **
        **************
110 FOR n=1 TO 26
120 READ d
130 BEEP .25,d
140 NEXT n
150 DATA 11,14,11,11,14,11,12,
    14,12,9,11,12,11,7,11,14,
    11,11,14,11,12,14,12,9,11,7
```

CALCULATORUL — INSTRUMENT MUZICAL

Simulînd grafic pe ecran claviatura unui pian, de exemplu, putem realiza un program prin care acţionarea unor taste ale calculatorului să producă notele corespunzătoare clapelor pianului.

Programul următor utilizează caracterele grafice definite pentru a simula grafic o parte a unei claviaturi de pian (o octavă şi ceva). Putem obţine notele respective prin acţionarea unor taste ale calculatorului. În general, tastele de pe rîndul de sus (tastele numerice 1, 2, 4, 5, 6, 8 şi 9) reprezintă clapele negre ale pianului, iar cele de pe rîndul al 2-lea reprezintă pe cele albe. Pentru a obţine gama DO, de exemplu, se vor acţiona, succesiv, tastele **W, E, R, T, Y, U, I** şi **O**.

Durata fiecărei note este aceeaşi. Dacă doriţi ca melodia cîntată să fie mai vioaie (sau mai lentă) puteţi modifica durata notei din cadrul liniei 230.

```
10 REM ** CALCULATORUL PIAN **
       ************************
20 FOR n=0 TO 7
30 POKE USR "p"+n,16 ------> Formarea caracterului
                             "clapă albă".
40 POKE USR "i"+n,252 -----> Formarea caracterului
                             "clapă neagră".
50 NEXT n
60 LET p$=""
70 LET n$=""
80 LET p=0
90 LET x=0
100 PAPER 0: INK 7: BORDER 0: CLS
110 PRINT
```

```
120 FOR m=1 TO 5
130 PRINT TAB 10;"|■■|■■■|■■"
140 NEXT m
150 FOR m=1 TO 4
160 PRINT TAB 10;"||||||||||"
170 NEXT m
180 LET k$=INKEY$
190 IF k$="" THEN GO TO 180
200 IF p$=k$ THEN GO TO 220
210 GO SUB 1000
220 PRINT OVER 1;AT x,9+p;n$
230 BEEP 0.3,a
240 LET p$=k$
250 PRINT OVER 0;AT 0,10;
    "                   ";
    AT 9,10;"||||||||||"
260 GO TO 180
1000 IF k$="1" THEN LET a=1: LET p=2
1005 IF k$="2" THEN LET a=3: LET p=3
1010 IF k$="4" THEN LET a=6: LET p=5
1015 IF k$="5" THEN LET a=8: LET p=6
1020 IF k$="6" THEN LET a=10: LET p=7
1025 IF k$="8" THEN LET a=13: LET p=9
1030 IF k$="9" THEN LET a=15: LET p=10
1035 IF k$="q" THEN LET a=-1: LET x=8:
     LET p=1
1040 IF k$="w" THEN LET a=0: LET p=2
1045 IF k$="e" THEN LET a=2: LET p=3
1050 IF k$="r" THEN LET a=4: LET p=4
1055 IF k$="t" THEN LET a=5: LET p=5
1060 IF k$="y" THEN LET a=7: LET p=6
1065 IF k$="u" THEN LET a=9: LET p=7
1070 IF k$="i" THEN LET a=11: LET p=8
1075 IF k$="o" THEN LET a=12: LET p=9
1080 IF k$="p" THEN LET a=14: LET p=10
1085 LET x=9: LET n$="■"
1090 IF k$="1" OR k$="2" OR k$="4" OR
     k$="5" OR k$="6" OR k$="8" OR k$="9"
     THEN LET x=0: LET n$="■"
1100 RETURN
```

Tastele PIIP... în mod 6.

De 10 ori P în mod 6.

Apelarea subrutinei pentru stabilirea înălțimii sunetului (a) și a poziției clapei (p).

19 spații libere.

Dacă se acționează tasta 1 (k$="1"), atunci se va auzi nota do (a=1), marcîndu-se grafic clapa respectivă.

Tasta 5 în mod 6.

Marcarea grafică a clapei (albă sau neagră) care se acționează.

Tastele CS+5 în mod 6.

Acum puteți cînta o melodie la... calculator!

CALCULATORUL MUZICIAN

Programul precedent transformă calculatorul într-un instrument muzical. Dar, oare, nu se poate mai mult, adică să se realizeze un program care să transforme calculatorul într-un ajutor de nădejde al unui muzician? În acest scop, calculatorul va trebui să se comporte ca un instrument muzical, dar, în acelaşi timp, să aibă capacitatea de a memora melodii, de a le vizualiza sau reda (la cerere), de a le putea modifica.

Redactarea melodiilor sub forma partiturilor pe ecran se realizează, într-un fel asemănător cu redactarea textelor pe ecran (prelucrare şi editare de texte). Vor trebui desenate pe ecran portativul şi notele de pe portativ pentru o anumită secvenţă melodică. De asemenea, va trebui să existe posibilitatea ca o notă să fie ştearsă, punîndu-se alta în locul ei. Pentru note (eventual pătrimi, doimi, etc.), se vor putea proiecta caractere grafice speciale. Apoi, va trebui să existe o legătură între notele care apar pe ecran şi sunetele care sînt emise de calculator. În acest scop valorile pentru duratele şi înălţimile sunetelor vor trebui memorate în variabile.

Toate aceste probleme le puteţi rezolva singuri pe baza cunoştinţelor acumulate pînă în prezent.

28. *Încercaţi să realizaţi un program care să întrunească condiţiile descrise mai sus, adică să poată memora melodii şi să le poată reda, atît sonor, cît şi grafic*

Răspunsul la pagina 144.

SĂ APLICĂM TEHNICILE ÎNVĂȚATE :

Jocul „TARZAN"

Pentru realizarea unui joc complex, trebuie să existe un scop bine determinat în cadrul unui scenariu, care poate fi atins de jucător în funcție de abilitatea sa (gîndire logică, inteligență, reflexe, îndemînare), folosind anumite reguli stabilite.

Vom încerca să realizăm împreună un astfel de joc. La început ne vom imagina scenariul după care se desfășoară jocul:

Locul acțiunii: jungla.

Acțiunea (obiectivul): nouă exploratori curajoși trebuie să traverseze o prăpastie, pe fundul căreia curge un pîrîu.

Desfășurarea acțiunii: singurul mijloc prin care exploratorii pot ajunge de la un mal la celălalt este să se agațe pe rînd, în momentul potrivit, de una din liane (care se balansează în vînt), în stilul cunoscutului personaj Tarzan, fiul junglei.

Descrierea jocului pe calculator: prezentarea jocului (scenariu, reguli); desenare decor (prăpastia, torentul, liana) și personaje (exploratorii); balansul lianei; plecarea cîte unui explorator și încercarea de a prinde liana (dacă un explorator reușește să prindă liana, aceasta îl ajută să treacă pe malul celălalt, dacă nu, va cădea în apele pîrîului); prezentarea rezultatului la sfîrșitul jocului (cîți exploratori au trecut pe malul celălalt și cîți au căzut în apă); posibilitatea de reluare a jocului prin dialog cu calculatorul.

Mijloace folosite de jucător (taste acționate): pentru conducerea exploratorului în încercarea de a prinde liana, jucătorul poate acționa oricare din taste.

Mijloace (tehnici) folosite de programator pentru realizarea jocului:

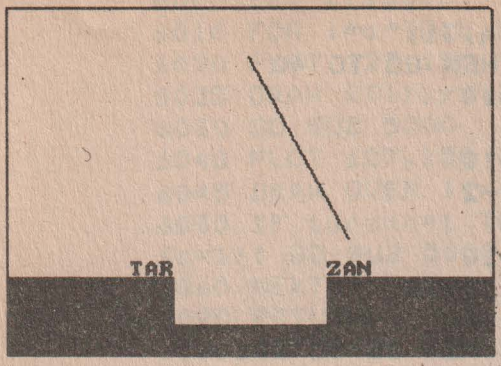


Fig. 66

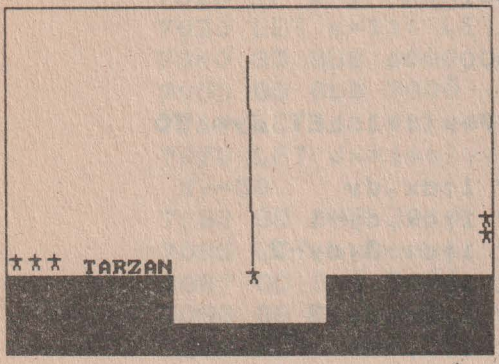


Fig. 67

— prezentare joc: deplasarea pe ecran a şirului de caractere care formează titlul (vezi fig. 66);
— realizare decor: grafica de rezoluţie înaltă (linii, puncte) pentru desenarea lianei; grafica de rezoluţie scăzută (caractere grafice) pentru desenarea prăpastiei şi torentului;
— definirea de noi caractere grafice pentru caracterul explorator;
— animaţia pentru balansarea lianei, deplasarea exploratorilor, saltul şi prinderea lianei (vezi fig. 67);
— deplasarea într-un decor (deplasarea exploratorilor);
— efecte sonore la prezentarea şi sfîrşitul jocului, precum şi la evenimentele trecerii exploratorului pe celălalt mal sau a căderii lui în apă;
— dialog calculator-jucător pentru eventuala reluare a jocului.

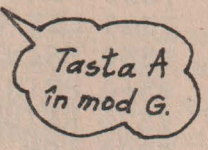
Iată jocul realizat (de data aceasta programul se prezintă aşa cum apare el la listare pe ecranul TV):

```
  5 REM    **  TARZAN  **
           *            *
           **************
  10 GO SUB 9000
  20 GO SUB 8000
  30 GO SUB 7000
  40 PLOT OVER 1;127,150: PRINT
AT 19,11;
  45 FOR i=1 TO m: PRINT INK 2;
FLASH 1;"*";: NEXT i
  50 INPUT "Inca un joc? ";r$
  60 IF r$="" OR r$="d" OR r$
="da" OR r$="D" OR r$="DA" THEN
RUN
  70 STOP
1000 LET ty=50: LET tx=240
1010 LET dx=tx: LET dy=ty
1020 LET tarzan=0: LET s=0
```

Tasta A în mod G.

```
1030 LET q=q+1
1040 PRINT AT q,31;" "
1050 IF om=0 THEN GO TO 40
1060 GO SUB 3010
1070 RETURN
2000 GO SUB 3010
2010 LET ty=ty-2
2020 LET dy=ty
2030 GO SUB 3010
2040 BEEP 0.01,ty/2
2050 GO SUB 3010
2060 IF ty>30 THEN GO TO 2010
2070 BEEP 1,-10
2085 LET om=om-1
2090 LET m=m+1
2095 GO SUB 1000
2100 RETURN
3000 LET dx=127+x(i): LET dy=150
+y(i)-2
3010 PLOT OVER 1;dx,dy
3020 PLOT OVER 1;dx,dy-1
3030 PLOT OVER 1;dx-3,dy-2
3040 DRAW OVER 1;6,0
3050 PLOT OVER 1;dx,dy-3
3060 DRAW OVER 1;2,-4
3070 PLOT OVER 1;dx+1,dy-3
3080 DRAW OVER 1;-2,-4
3090 RETURN
4000>LET tarzan=1
4010 LET dx=tx: LET dy=ty
4020 GO SUB 3010
4030 LET tx=tx-10: LET dx=tx
4040 GO SUB 3010
4050 IF ABS (tx-127-x(i))<5 AND
steag=1  THEN  LET  s=1: GO SUB
3010: GO TO 4070
4060 IF tx<160 THEN GO SUB 2000
4070 RETURN
5000 IF s=1 THEN GO SUB 3000: GO
 TO 5020
5010 IF INKEY$<>"" OR  tarzan=1
THEN GO SUB 4000
5020 RETURN
```

```
6000 LET steag=0
6010 FOR i=a TO b STEP c
6020 PLOT 127,150
6025 DRAW x(i),y(i)
6030 GO SUB 5000
6040 PLOT 127,150
6045 DRAW OVER 1;x(i),y(i)
6050 IF tarzan=1 THEN LET steag=
(c=1): GO SUB 5000
6060 NEXT i
6070 RETURN
7000 LET a=1: LET b=11: LET c=1
7010 GO SUB 6000
7020 IF s=1 THEN GO TO 7060
7030 LET a=11: LET b=1: LET c=-1
7040 GO SUB 6000
7050 GO SUB 7000
7060 LET salvat=salvat+1
7070 LET dx=(salvat-1)*10+5: LET
 dy=50
7080 GO SUB 3010
7085 LET om=om-1
7087 GO SUB 1000
7088 GO TO 7030
7090 RETURN
8000 LET om=9: LET salvat=0
8010 FOR i=0 TO 7
8020 READ a
8030 POKE USR "A"+i,a
8040 NEXT i
8050 DATA 16,16,254,24,24,24,36,
36
8070 FOR q=6 TO 14
8080 PRINT AT q,31;"▲"
8090 NEXT q
8100 LET q=5: LET m=0
8110 GO SUB 1000
8120 RETURN
9000 BORDER  7: PAPER 7: INK 0:
CLS
9010 DIM x(11): DIM y(11)
9020 LET i=0
9030 FOR j=-PI/6 TO PI/6 STEP .1
9040 LET i=i+1
```



```
9050 LET x(i)=-105*SIN j
9060 LET y(i)=-105*COS j
9070 NEXT j
9080 LET a$="[redacted]"
9090 LET b$="TARZAN": LET c$=""
9100 LET d=11
9110 GO SUB 9500
9120 PRINT AT 16,21;b$
9130 LET a=1: LET b=11: LET c=1
9140 BEEP 0.01,10: GO SUB 9900
9150 LET  c$=c$+b$(1) : LET  b$=
b$(2 TO )
9160 LET d=d-1
9170 PRINT AT 16,21;"       ";AT
16,21;b$;AT 16,d;c$
9180 LET a=11: LET b=1: LET c=-1
9190 BEEP 0.01,20: GO SUB 9900
9200 IF b$()"" THEN GO TO 9130
9210 PRINT AT 16,d; INK 2; FLASH
 1;c$
9220 PLOT OVER 1;127,150
9230 FOR i=1 TO 10
9240 FOR j=20 TO 30
9250 BEEP 0.01,j
9260 NEXT j
9270 NEXT i
9280 PRINT AT 16,d;
9290 RETURN
9500 PRINT AT 17,0;
9510 FOR i=17 TO 21
9520 PRINT  TAB 0; INK 4;a$;TAB
21; INK 4;a$
9530 NEXT i
9540 LET a$=a$(2 TO )
9550 PRINT AT 20,11; INK 1;a$
9560 PRINT AT 21,11; INK 1;a$
9570 RETURN
9900 FOR i=a TO b STEP c
9910 PLOT 127,150
9920 DRAW x(i),y(i)
9930 PLOT 127,150
9935 DRAW OVER 1;x(i),y(i)
9940 NEXT i
9950 RETURN
```

STRUCTURA, DESCRIEREA ŞI VARIABILE IMPORTANTE ALE PROGRAMULUI

● **Programul principal** (liniile 5—70): apelul subrutinei de prezentare a jocului (linia 10), apelul subrutinei de iniţializare (linia 20), apelul subrutinei de joc (linia 30), afişarea exploratorilor căzuţi în apele torentului (linia 40 şi 45), posibilitatea reluării jocului (linia 60).

● **Subrutina de"iniţializare"a unui explorator** prin care se pune în mişcare un nou explorator de fiecare dată cînd predecesorul a ajuns pe malul celălalt sau a căzut în prăpastie (liniile 1 000—1 070): stabilirea coordonatelor de plecare a exploratorului (linia 1 000), stabilirea coordonatelor desenului (linia 1 010), exploratorul nu este în mişcare (linia 1 020), oprirea jocului cînd nu mai sînt exploratori (linia 1 050) — singura situaţie de ieşire către programul principal, apelarea subrutine pentru afişarea unui explorator (linia 1 060).

● **Subrutina de cădere a exploratorului** (liniile 2000—2100): căderea exploratorului însoţită de sunete adecvate (liniile 2000—2070), apelul subrutinei de iniţializare a unui explorator (linia 2095).

● **Subrutina de afişare a unui explorator** (liniile 3000—3090): permite afişarea caracterului explorator în orice loc pe ecran şi urmărirea corectă a deplasării lianei.

● **Subrutina de deplasare a unui explorator** (liniile 4000—4070): punerea pe 1 a variabilei **tarzan** pentru indicarea faptului că exploratorul este în mişcare (linia 4000), afişarea exploratorului (liniile 4010 şi 4020), deplasarea şi afişarea exploratorului (liniile 4030 şi 4040), test care verifică dacă exploratorul a reuşit să se agaţe de liană (linia 4050: o distanţă mică faţă de extremitatea lianei şi acelaşi sens de deplasare cu cel al lianei. Dacă testul este trecut variabila **s** se pune pe 1, aceasta semnalizînd faptul că un explorator s-a prins de liană), test care verifică dacă exploratorul cade în prăpastie (linia 4060).

● **Subrutina de dispecerat** (liniile 5000—5020): cazul în care liana este ocupată (linia 5010), cazul în care jucătorul acţionează o tastă sau un explorator este pe punctul de a se deplasa (linia 5020).

● **Subrutina de deplasare a lianei** (liniile 6000—6070): iniţializarea variabilei **steag** pentru a arăta dacă liana poate fi apucată de explorator (linia 6000), parametrizarea unui ciclu pentru utilizarea aceleiaşi subrutine, oricare ar fi sensul de deplasare a lianei (linia 6010), afişarea lianei (linia 6020), apelarea subrutinei de dispecerat (linia 6030), ştergerea lianei (linia 6040), cazul în care un explorator este în mişcare (linia 6050).

● **Subrutina de joc** (liniile 7000—7090): deplasarea lianei de la dreapta la stînga (linia 7000), apelarea subrutinei de deplasare a lianei (linia 7010); dacă un

explorator este suspendat de liană, atunci va fi depus pe mal (linia 7020), liana se va deplasa de la stînga la dreapta (linia 7030), apelarea subrutinei de deplasare a lianei (linia 7040), reîntoarcerea la începutul jocului (linia 7050), cazul în care exploratorul a ajuns pe mal (linia 7060), afişarea exploratorului în poziţia de sosire (7070 şi 7080), apelul subrutinei de iniţializare a unui explorator (linia 7087), reîntoarcerea în mijlocul ciclului de joc pentru continuarea mişcării lianei (linia 7088).

● **Subrutina de iniţializare** (liniile 8000—8120): iniţializarea variabilei **om**, care memorează numărul total de exploratori, precum şi a variabilei **salvat**, care memorează numărul de exploratori care au ajuns pe malul opus (linia 8000), definirea caracterului grafic explorator — tasta **A** — (liniile 8070—8090), apelarea subrutinei de iniţializare a unui explorator (linia 8110).

● **Subrutina de prezentare** (liniile 9000—9290): stabilirea atributelor de culoare (linia 9000), dimensionarea vectorilor **x** şi **y** care vor conţine coordonatele punctelor situate pe traiectoria lianei, la extremitatea sa (linia 9010), calculul coordonatelor punctelor de la extremitatea lianei (ljniile 9020—9070), iniţializarea variabilelor **a$, b$, c$** şi **d** utilizate la prezentarea jocului (liniile 9080—9100), apelul subrutinei de afişare a prăpastiei (linia 9110), prezentarea jocului prin "trecerea" peste prăpastie a cuvîntului TARZAN (liniile 9130—9200), subrutina de efecte sonore (liniile 9230—9270).

● **Subrutina de afişare a prăpastiei** (liniile 9500—9570).

● **Subrutina de deplasare a şirului de caractere** pentru prezentare (liniile 9900—9950).

Pe scheletul unui astfel de joc, se pot adăuga şi alte elemente care pot face jocul şi mai complex. De exemplu, unduirea apei; pot apărea şi animale care să îngreuneze ajungerea cu bine a exploratorilor (tigri, crocodili, etc.); pot avea loc lupte cu aceste animale, etc. Programarea unor astfel de tipuri de jocuri poate demonstra, însă, că anumite structuri sînt ineficiente în execuţia programului, subrutinele care necesită volum mare de calcule avînd ca efect încetinirea unor cicluri din cadrul programului. De asemenea, programarea jocurilor poate deveni foarte complexă atunci cînd se vor imagina tactici şi strategii de urmat de către calculator la un anumit răspuns din partea jucătorului.

V. Răspunsuri la exerciţiile şi temele propuse

1. În urma corectării descrierii algoritmului în pseudocod după diagrama de tip arbore, acesta va arăta astfel:

Comentariu ** — Algoritm pentru ca robotul să facă o ceaşcă de ceai **

Comentariu * — Modul: Fierberea apei — sarcina 1.1. *

1.1.1. Umple ceainicul cu apă.

1.1.2. Pune ceainicul pe plită.

1.1.3. Aşteaptă pînă cînd fierbe apa.

1.1.4. Ia ceainicul de pe plită.

Comentariu * Sfîrşitul sarcinii 1.1.*

Comentariu * Modul: Pune ceaiul în ceaşcă — sarcina 1.2. *

1.2.1. Ia o ceaşcă de ceai.

1.2.2. Pune un pachet de ceai în ceaşcă.

Comentariu * Sfîrşitul sarcinii 1.2. *

Comentariu * Modul: Pune apa în ceaşcă — sarcina 1.3. *

1.3.1. Adu ceaşca la ceainic.

1.3.2. Pune apă în ceaşcă.

1.3.3. Amestecă ceaiul cu linguriţa.

Comentariu * Sfîrşitul sarcinii 1.3. *

Comentariu ** Sfîrşitul algoritmului — ceaşca cu ceai **

2. Se vor descompune în continuare subsarcinile. De exemplu: sarcina **1.1.1. Umple ceainicul cu apă,** va deveni (vezi fig. 3):

1.1.1.1. Pune ceainicul sub robinet.

1.1.1.2. Răsucește robinetul spre dreapta.

1.1.1.3. Așteaptă pînă cînd ceainicul e plin.

1.1.1.4. Răsucește robinetul spre stînga.

Se pot adăuga și subsarcini noi pe același nivel. De exemplu, pe nivelul 2, se poate adăuga o subsarcină:

1.3. Pune zahăr în ceașcă.

iar subsarcina **1.3. Pune apă în ceașcă** va deveni 1.4.

3. Depinde și de tipul calculatorului. Dacă este vorba despre **HC, TIM, COBRA, CIP, JET, aMIC** sau **PRAE,** puteți să vă inspirați din **"Partenerul meu de joc — calculatorul"**, pag. 12.

4. a) Vezi fig. 68.

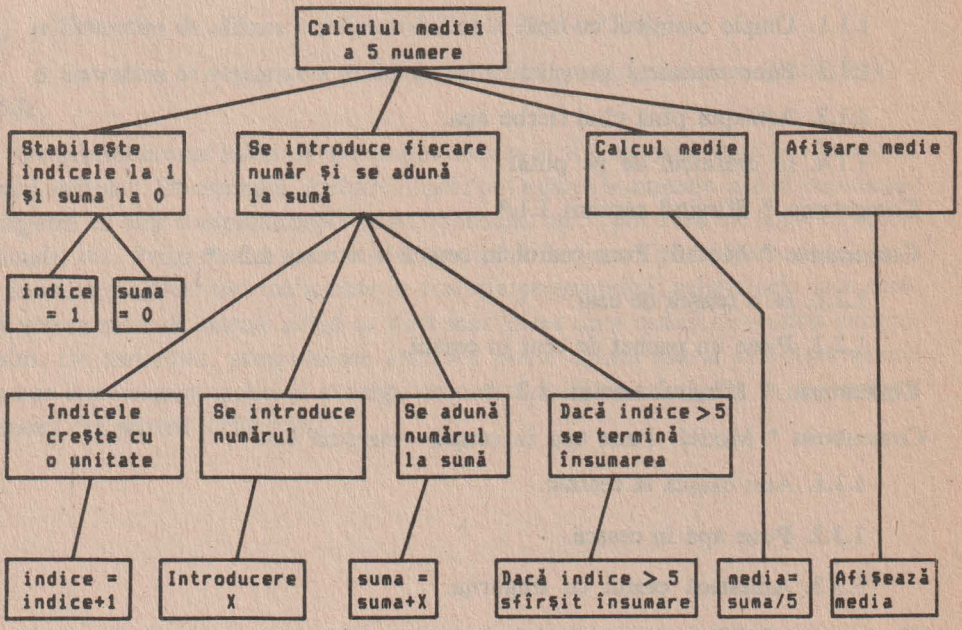
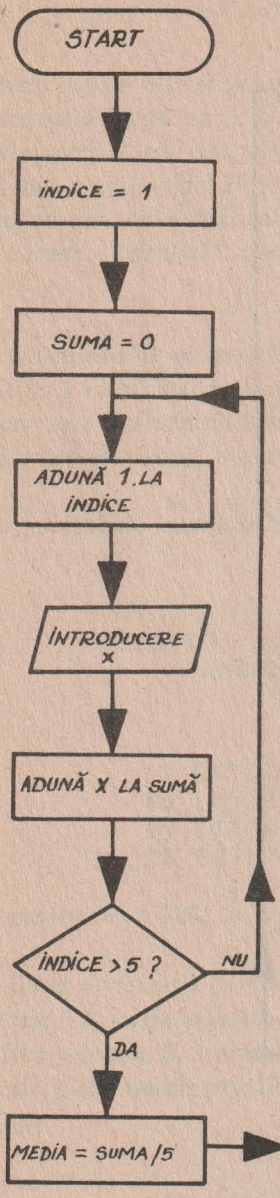


Fig. 68 Diagrama de tip arbore pentru calculul mediei a 5 numere.

b) şi c) vezi fig. 69.

Schema logică



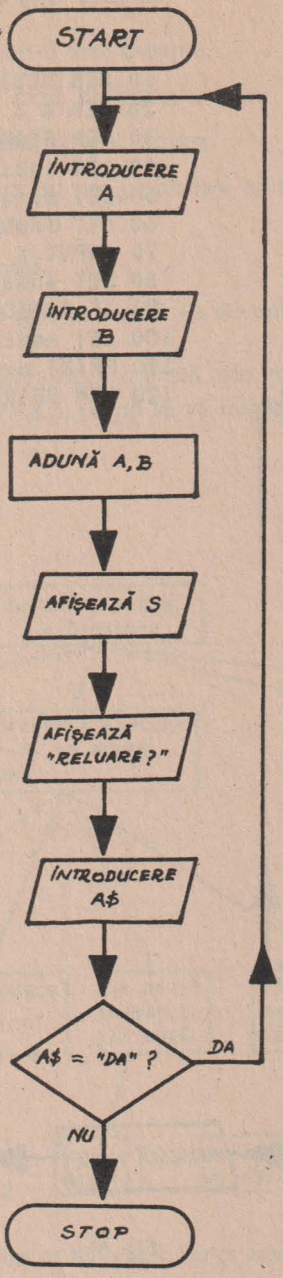
Programul

```
10 REM MEDIA
20 REM A 5 NUMERE
30 REM START
40 LET indice=1
50 LET suma=0
60 LET indice=indice+1
70 INPUT X
80 LET suma=suma+X
90 IF INDICE =< 5 THEN 60 TO 60
100 LET media=suma/5
110 PRINT media
120 REM SFIRSIT
```

Fig. 69

5. a) şi b) Vezi fig. 70.

Schema logică



6. a) $(101101)_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 32 + 8 + 4 + 1 = \mathbf{45}$

   b) $(3A7)H = 3 \cdot 16^2 + 10 \cdot 16^1 + 7 \cdot 16^0 = 768 + 160 + 7 = \mathbf{935}$

7. Cel mai mare număr care se poate înscrie într-un octet se obține "punînd" în fiecare "fereastră" cîte un 1 și, astfel, vom avea: $1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + \ldots$ $\ldots 1 \cdot 2^0$. Mai simplu, cel mai mare număr care se poate înscrie într-un octet se obține scăzînd o unitate din 2^8. Acest număr este **255**.
   Bineînțeles, cel mai mic număr care se poate înscrie într-un octet se obține "punînd" în fiecare "fereastră" cîte un 0, rezultînd astfel pentru întreg octetul tot **0**.

8. Deoarece funcția **RND** generează un număr aleator mai mare (sau egal cu **0**) și mai mic ca **1** (deci niciodată egal cu **1**). Astfel **RND** × **100** va fi un număr aleator mai mare sau egal cu **0** și mai mic decît **100**. Pentru a putea atinge și această margine se adaugă o unitate.

9. Deoarece numărul de întrebări N este un număr impar.

10.
```
212 IF Y$<>Z$ THEN PRINT
    "RASPUNSUL CORECT:";Y$
```

11.
```
  7 LET I=0
 10 LET I=I+1
 15 LET A=INT (RND*(10+1))
 25 PRINT "OPERATIA NR ";I
```

12. Afișarea mediei (linia **280**) se va face cu valoarea **INT (A * 100)/100**.

13. Se poate folosi o variabilă (S) care ia inițial valoarea **0** și, dacă se face cel puțin o modificare, i se va da valoarea **1**. Dacă la sfîrșitul unei treceri prin listă variabila va avea valoarea **0**, înseamnă că ordonarea este terminată și nu mai este necesară nici o altă trecere prin listă. În acest scop, se adaugă programului următoarele linii:

```
235 LET S=0
295 LET S=1
305 IF S=0 THEN GO TO 320
```

```
10 LET C=27
20 PRINT "PRIMUL NUMAR?
30 INPUT X
40 LET X$=CHR$ (X-C)
50 PRINT X$
60 PRINT "URMATORUL NUMAR?"
70 GO TO 30
```

15. JOI ZIUA LUI ŞTEFAN STOP AFLĂ DACĂ ARE UN JOC DE ŞAH ŞI UN CALCULATOR STOP.

16. Textul era scris deandoaselea. Pentru decodificare, se vor modifica în program liniile **110** şi **220** astfel:

```
FOR J=L TO 1 STEP -1
```

17. Codurile **ASCII** pentru literele mari se găsesc între **65** şi **90**. Trebuie modificată linia **90**.

18. Recunoaşterea se bazează pe faptul că majoritatea numelor (de fapt a prenumelor) persoanelor de sex feminin din limba română se termină cu litera **a**. Un exemplu de program ar putea fi următorul:

```
10 REM  BABA OARBA
20 PRINT "CUM TE CHEAMA ?"
30 INPUT N$
40 LET L=LEN N$
50 IF N$(L)="a" OR N$(L)="A"
   THEN GO TO 90
60 PRINT "CRED CA ESTI BAIAT"
70 PRINT
80 GO TO 20
90 PRINT "CRED CA ESTI FATA"
100 PRINT
110 GO TO 20
```

19. 1-**INSERT**: se inserează litera **i**; 2-**INSERT**: se inserează cuvîntul **zi**; 3-**INSERT**: se inserează litera **ă**; 4-**DELETE**: se şterge litera **t**; 5-**DELETE**: se şterge cuvîntul **şi**.

20. Iată un rezultat obținut de noi:

Număr aruncări	Rezultate			
	Stema		Banul	
	Frecvența	Procent	Frecvența	Procent
10	3	30%	7	70%
100	45	45%	55	55%
1 000	509	50,9%	491	49,1%

Se observă că dacă numărul de aruncări crește, diferența dintre numărul de apariții ale unei fețe (frecvența) și jumătatea numărului de aruncări crește (pentru 10 aruncări diferența este de $5-3=2$, pentru 100, diferența este de $50-45=5$, iar pentru 1 000 este de $509-500=9$). În schimb, procentul se apropie tot mai mult de 50% (comparați coloanele marcate). Dacă veți repeta experimentul, veți constata că observațiile făcute își mențin valabilitatea. De fapt ați experimentat ceea ce este cunoscut sub numele de **legea numerelor mari**, care sună astfel: pentru un număr foarte mare de încercări (aruncări) procentul aparițiilor se va apropia tot mai mult de probabilitatea de apariție (50%).

21. Iată un răspuns posibil:

```
10 CLS
20 PRINT AT 8,5;"
30 PRINT AT 9,5;"
40 PRINT AT 10,5;"
50 PRINT AT 11,5;"
60 PRINT AT 12,5;"
```

Rezultatul va arăta ca în fig. 71.

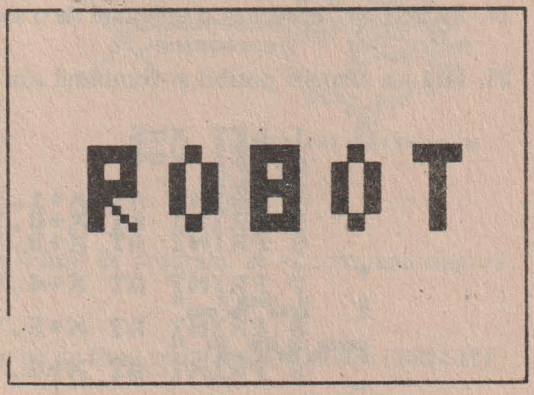


Fig. 71

22. Iată unele seturi posibile (ale voastre pot arăta şi altfel):

A. a1) 16, 56, 16, 58, 84, 48, 40, 68

a2) 8, 28, 61, 90, 24, 164, 66, 1 (fugind)

b) 60, 189, 255, 189, 60, 189, 255, 189

c) 40, 16, 56, 4, 60, 68, 60, 0

d) 16, 40, 0, 40, 16, 16, 56, 0

e) 0, 0, 56, 64, 56, 4, 120, 16

f) 0, 16, 16, 56, 16, 16, 12, 8

B. [P] [SS + P] [CS + U] [N] [SPACE] [O] [M] [U] [L] [E] [CS + 9]

PRINT " U n o m u l e

[T] [SPACE] [O] [SPACE] [I] [CS + 9] [N] [T] [R] [SS + J]

ţ î n t r —

[O] [SPACE] [M] [A] [CS + 9] [S] [CS + 9] [I] [N] [U]

o m a ş i n u

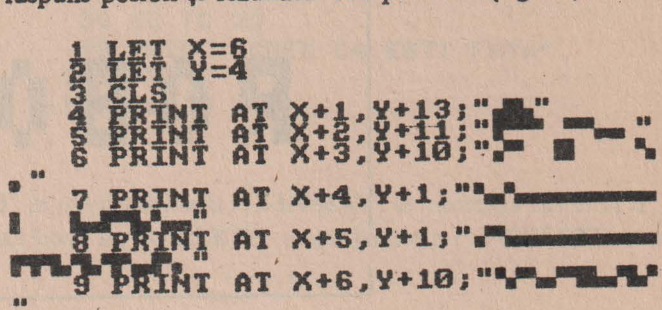
[CS + 9] [T] [A] [SPACE] [M] [CS + 9] [SS + P] [CR]

ţ ă

23. Aveţi toată libertatea de a realiza practic această temă.

24. **PAUSE 50** provoacă o aşteptare de o secundă din partea calculatorului.

25. Iată un răspuns posibil şi rezultatul său pe ecran (fig. 72):



```
10 PRINT AT X+7,Y+11;"
11 PRINT AT X+8,Y+11;"
12 PRINT AT X+9,Y+10;"
"
13 PRINT AT X,Y+8;"
 "
14 PRINT AT X+3,Y;"
15 PRINT AT X+6,Y;"
16 PRINT AT X,Y+8;"

 "
17 PRINT AT X+3,Y;"     "
18 PRINT AT X+6,Y;"     "
19 LET A$=INKEY$
20 IF A$="Q" THEN GO TO 26
21 IF A$="A" THEN GO TO 29
22 IF A$="D" THEN GO TO 32
23 IF A$="S" THEN GO TO 35
24 IF A$="B" THEN GO TO 38
25 GO TO 13
26 IF X=0 THEN GO TO 3
27 LET X=X-1
28 GO TO 3
29 IF X=12 THEN GO TO 3
30 LET X=X+1
31 GO TO 3
32 IF Y=12 THEN GO TO 3
33 LET Y=Y+1
34 GO TO 3
35 IF Y=0 THEN GO TO 3
36 LET Y=Y-1
37 GO TO 3
38> PRINT AT 20,12;"SFIRSIT"
```

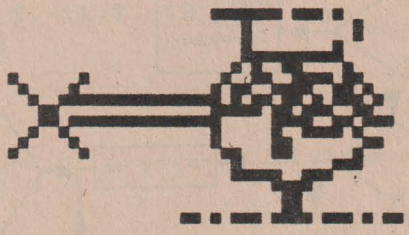


Fig. 72

26. Se scurtează programul: în loc de 4 linii de program, se va scrie una singură; nu se mai folosesc IF-urile.

27. A. a) Caracterul are culoarea roşie pe un fond verde şi este luminos (**BRIGHT**);

b) Caracterul are culoarea neagră pe un fond bleu şi este clipitor (**FLASH**);

c) Caracterul are culoarea albastră pe un fond alb, este luminos (**BRIGHT**) şi clipitor (**FLASH**).

B. Metoda de determinare a atributelor cînd se cunoaşte valoarea lui **ATTR** este următoarea:

— se testează dacă valoarea lui **ATTR** este mai mare decît numărul **128**; dacă da, atunci caracterul este clipitor (**FLASH** = **1**), iar valoarea care se va testa în continuare (noul **ATTR**) va fi diferenţa care rămîne prin scăderea lui **128**· dacă nu, atunci:

— se testează dacă valoarea lui **ATTR** este mai mare decît numărul **64**; dacă da, atunci caracterul este luminos (**BRIGHT** = **1**), iar valoarea care se va testa în continuare (noul **ATTR**) va fi diferenţa care rămîne prin scăderea lui **64**: dacă nu, atunci:

— codul culorii fondului (**PAPER**) va fi egal cu partea întreagă a împărţirii lui **ATTR** la **8**, iar codul culorii caracterului (**INK**) va fi egal cu restul scăderii **ATTR - 8 * INT (ATTR/8)**, adică, practic, restul după eliminarea codului culorii. Schema logică a puteţi urmări în fig. 73.

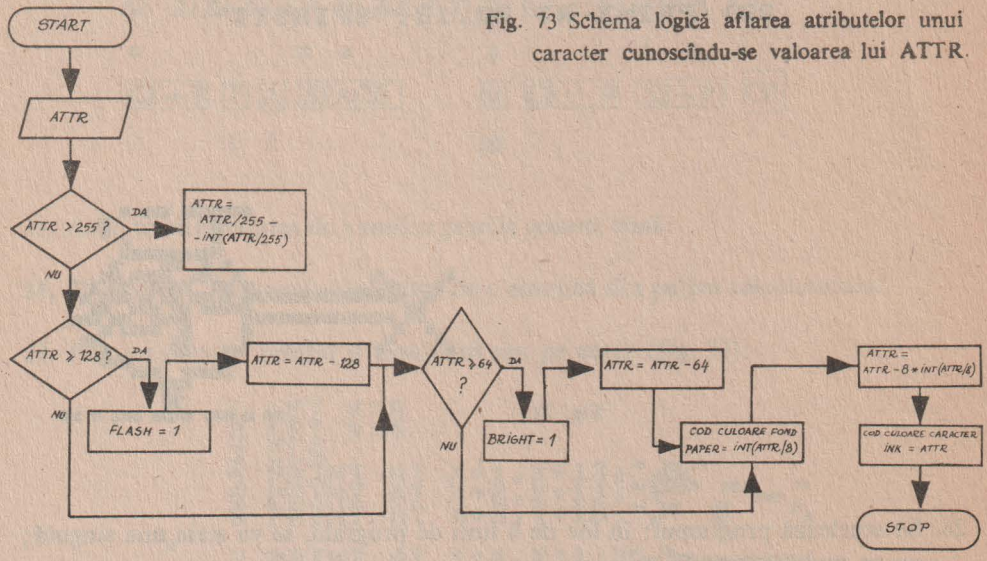


Fig. 73 Schema logică aflarea atributelor unui caracter cunoscîndu-se valoarea lui ATTR.

28. Prezentăm programul "**Muzicianul**", care rezolvă problemele expuse (programul vostru poate arăta altfel):

"Muzicianul" foloseşte tastatura, astfel încît se pot compune melodii (maximum 5 a cîte 32 de note, pe două octave). Pentru obţinerea notelor se vor folosi următoarele taste: Q (do #), W (re), E (re #), R (mi), T (fa), Y (fa #), A (mi), S (fa), D (fa #), F (sol), G (sol #), H (la), J (si b), K (si), L (do), Z (la), X (si b), C (si), V (do), B (do #), N (re), M (re #).

Pentru obţinerea notelor pe ecran s-a proiectat un caracter grafic corespunzător tastei O. Toate notele vor avea aceeaşi durată, dar aceasta se poate modifica la o nouă audiere a melodiei. Se pot înscrie pauze (care au durata egală cu a notei) cu ajutorul tastei **BREAK**.

Ultima notă se poate şterge şi, apoi, înlocui cu alta prin intermediul tastei **0**

```
  3 REM  * MUZICIANUL *
         **************
  5 BORDER 0: PAPER 7: INK 0
 10 PRINT "MUZICIANUL"
 15 BEEP 0.2,20
 20 PRINT AT 2,0;"Pregatire"
 25 DIM a(5,32): DIM c(255)
 30 FOR n=1 TO 255
 35 LET c(n)=-100
 40 NEXT n
 45 DATA 4,4,4,4,28,60,60,24
 50 FOR i=0 TO 7
 52 READ e
 55 POKE USR "o"+i,e
 57 NEXT i
 60 DATA -5,-8,-10,-3,6,-2,-1,0
 65 DATA -100,1,2,3,-6,-7,-100,
-100,4,7,-4
 67 DATA 8,-100,-9,5,-11,9,-12
 70 FOR n=97 TO 122
 80 READ c(n)
 85 LET c(n)=c(n)
 90 NEXT n
100 LET m=1
110 CLS
112 LET n=0
115 BEEP 0.2,20
117 PRINT AT 0,2;"Puteti compun
e !"
120 LET t$=INKEY$
130 IF t$="" THEN GO TO 120
140 IF t$="0" THEN PRINT AT a(m
,n)-9,n-1;" ": LET n=n-1: PAUSE
15: GO TO 120
```

```
150 LET t=CODE t$
155 IF t=32 THEN LET n=n+1: LET
a(m,n)=256: PRINT AT 21,n-1;"-"
: GO TO 210
170 LET n=n+1
180 LET a(m,n)=c(t)
190 PRINT AT a(m,n)-9,n-1;"j"
200 BEEP 0.1,a(m,n)
210 IF n<32 THEN GO TO 120
220 IF m=5 THEN GO TO 300
230 BEEP 0.2,20: INPUT "alta me
lodie? (d/n) ";r$
240 IF r$="n" THEN GO TO 300
250 IF r$="d" THEN LET m=m+1: G
O TO 110
260 GO TO 230
300 BEEP 0.2,20: INPUT "Doriti
auditie? (d/n) ";r$
310>IF r$="n" THEN BEEP 0.2,13:
BEEP 0.2,16: INK 0: STOP
320 IF r$<>"d" THEN GO TO 300
325 BEEP 0.2,20: INPUT "durata
fiecarei note (in sec) ";dur
330 FOR p=1 TO m
335 CLS
340 FOR n=1 TO 32
345 IF a(p,n)=256 THEN PRINT AT
21,n-1;" ": PAUSE dur*50: GO TO
370
350 PRINT AT a(p,n)-9,n-1;"j"
360 BEEP dur,a(p,n)
370 NEXT n
380 NEXT p
400 BEEP 0.2,20: INPUT "Mai dor
iti? (d/n) ";r$
410 IF r$="d" THEN GO TO 325
420 IF r$<>"n" THEN GO TO 400
430 BEEP 0.2,13: BEEP 0.2,16
450 STOP
```

Tasta 0 în mod G

Tasta 0 în mod G

Acum puteți compune melodii pe care calculatorul le va memora și reda la cerere.

ANEXĂ

INDEXUL CUVINTELOR CHEIE

Prin cuvinte cheie se înţeleg cuvintele introduse în memoria calculatorului şi care prezintă o anumită semnificaţie pentru acesta. La calculatoarele la care se referă lucrarea de faţă, orice cuvînt cheie (pentru limbajul BASIC) se introduce prin acţionarea unei anumite taste sau a unei combinaţii de taste. Orice cuvînt cheie poate face parte din următoarele categorii:

- **comandă** - cuvînt cheie a cărei introducere are ca urmare execuţia imediată de către calculator a unei acţiuni. *Exemple:* RUN, LIST, LOAD etc.

- **instrucţiune** - cuvînt cheie care are ca urmare o acţiune executată de calculator şi care poate fi utilizat într-o linie de program. Acţiunea este îndeplinită numai atunci cînd programul va fi executat. *Exemple:* INPUT, READ, PRINT etc.

Notă. Un cuvînt cheie care reprezintă o comandă va funcţiona ca o instrucţiune atunci cînd este folosit într-o linie de program. *De exemplu:* 1000 LIST, cuvîntul cheie LIST reprezintă o instrucţiune, acţiunea de listare a programului realizîndu-se cînd execuţia programului va ajunge la linia 1000.

Invers, o instrucţiune poate fi considerată comandă cînd va fi folosită ca atare. *De exemplu*: PRINT 3+4, cuvîntul cheie PRINT reprezintă o comandă deoarece acţiunea de afişare 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 sau instrucţiune, deoarece deşi se poate utiliza şi ca instrucţiune, se utilizează în primul rînd ca şi comandă. Invers, INPUT se va trece la categoria instrucţiune/comandă, deoarece deşi se poate folosi şi în mod comandă, se utilizează în primul rînd în linii de program, deci ca instrucţiune.

- **funcţie** - cuvînt cheie a cărui folosire produce o valoare. Acest cuvînt se utilizează în combinaţie cu o comandă sau o instrucţiune.

Exemple: INT, RND, ABS, SIN, SQR.

Exemplu de utilizare: PRINT INT 8.1

- **operator logic** - cuvînt cheie care este utilizat într-o comandă sau instrucţiune şi care determină dacă o condiţie este adevărată sau falsă.

La calculatoarele la care se referă lucrarea de faţă se utilizează doar trei operatori logici, şi anume, **AND, OR şi NOT.**

- alte cuvinte cheie - cuvinte cheie care sînt utilizate în cadrul unor comenzi sau instrucţiuni şi care reprezintă particule de propoziţii făcînd o legătură între cuvinte.

Exemple: THEN, AT, STEP etc.

În scopul cunoaşterii şi folosirii cuvintelor cheie, adică a realizării de programe, vor trebui cunoscute:

- localizarea cuvintelor cheie pe tastatură (pe ce tastă se găseşte şi în ce mod se obţine);
- ce fel de cuvînt cheie este (categoria);
- scopul utilizării cuvîntului cheie (în ce caz se foloseşte);
- formatul cuvîntului cheie (sintaxa).

Deoarece în lucrare sînt oferite numeroase exemple de folosire a cuvintelor cheie în programe, deseori fiind indicată şi modalitatea de obţinere s-a considerat util să se facă doar o trecere în revistă (memento) în oridine alfabetică a tuturor cuvintelor cheie folosite, indicîndu-se doar semnificaţia (din care rezultă şi scopul principal de folosire), categoria în care se încadrează cuvîntul precum şi paginile din lucrare în care puteţi regăsi cuvîntul cheie respectiv într-un anumit program. Indexul nu conţine toate cuvintele cheie din limbajul BASIC, ci doar marea majoritate. Restul de cuvinte le puteţi găsi pe tastatură,iar pentru utilizarea lor puteţi consulta un manual de referinţă.

ABS (valoare **ABS**solută). Funcţie. Produce valoarea absolută a unei valori numerice, adică valoarea fără semn pozitiv sau negativ. *Pag:* 37, 130.

AND (ŞI). Operator logic. Leagă două sau mai multe condiţii în cadrul unei instrucţiuni. Dacă toate condiţiile legate prin AND sînt adevărate, atunci combinaţia va fi adevărată. *Pag:* 37, 48, 115, 120.

AT (afişează LA sau scrie LA). Particulă de legătură folosită în cadrul instrucţiunilor PRINT, LPRINT sau INPUT pentru a localiza o afişare pe ecran sau o imprimare pe hîrtie. *Pag;* 85, 87, 88, 89, 95, 97, 102, 103, 110-114, 116, 118, 119, 126, 130, 131, 132.

ATTR (**ATTR**ibute). Funcţie. Depistează atributele (culoarea cu care se afişează culoarea fondului, luminozitatea, intermitenţa afişării) unui caracter a cărei poziţie pe ecran este specificată. *Pag:* 116, 117.

BEEP (sunet). Instrucţiune/comandă. Produce o notă muzicală de o anumită durată şi înălţime. *Pag:* 102, 106, 122, 123, 124, 125, 126.

BIN (număr **BIN**ar). Funcţie. Transformă un număr binar într-unul zecimal. *Pag:* 90, 91, 98.

BORDER (BORDURA, marginea, chenarul ecranului). Instrucţiune/comandă. Determină culoarea marginii ecranului. *Pag:* 80-83, 87,88,91,95,101,103, 105, 110, 119,131.

BRIGHT (luminozitatea). Instrucţiune/comandă. Determină afişarea caracterelor într-o culoare mai luminoasă decît normal. *Pag*: 103, 117.

CHR$ (şir de CHaRactere) Funcţie. Produce orice caracter sau cuvînt cheie care se găseşte pe tastatură. *Pag:* 59,61, 106.

CIRCLE (CERC). Instrucţiune/comandă. Trasează un cerc pe ecran. *Pag:* 99.

CLS (CLear Screen, şterge ecranul). Instrucţiune/comandă. Şterge de pe ecran orice text sau grafică, lăsînd întregul ecran cu culoarea fondului. *Pag:* 77, 99, 101, 104, 110, 119.

CODE (CODul). Funcţie. Produce codul numeric (ASCII) al unui caracter sau cuvînt cheie care se găseşte pe tastatura calculatorului. *Pag:* 59-62.

COS (COSinus). Funcţie. Produce cosinusul unui unghi. *Pag*: 132.

DATA (DATe). Instrucţiune. Specifică o listă de date. Se foloseşte în program în conjuncţie cu instrucţiunea READ. *Pag:* 44, 45, 47, 50, 69, 83, 84, 92, 95, 110, 115, 119, 124, 125, 131.

DIM(DIMensiune). Instrucţiune. Rezervă un spaţiu de memorie. *Pag:* 40, 41, 44, 50, 51, 55, 57, 61, 62, 64, 67, 69, 131.

DRAW (desenează linie). Instrucţiune/comandă. Desenează pe ecran linii drepte sau curbe. *Pag:* 98, 100, 101, 130, 132.

FLASH (afişare intermitentă, clipitoare). Instrucţiune/comandă. Afişează intermitent o poziţie caracter (culoarea caracterului allternează cu culoarea fondului cu o frecvenţă constantă). *Pag:* 103, 116, 117.

FOR (repetă). Instrucţiune/comandă. Se începe o structură ciclică (buclă). Se utilizează în conjuncţie cu NEXT care încheie bucla. Secţiunea dintre FOR şi NEXT se repetă de un număr de ori indicat în linia FOR. *Pag:* 38, 40, 42-45, 50, 52, 55, 57... în toate programele în care sînt necesare structuri ciclice.

GO SUB (GO SUBroutine, mergi la subrutină). Instrucţiune/comandă. Salt necondiţionat de linie indicat. *Pag:* 37, 38, 42, 43, 44, 45...

IF (dacă). Instrucţiune/comandă. Testarea unei condiţii. *Pag:* 36, 37, 38, 42, 43,44, 46, 48, 49 etc.

INK (cerneala, culoarea caracterului). Instrucţiune/comandă. Specifică culoarea cu care se afişează caracterele, punctele şi liniile pe ecran. *Pag:* 80, 81, 82, 83, 84, 87, 88, 89, 91, 98, 99, 101, 103, 104, 105, 110, 111, 113, 116, 119, 129, 132.

INKEY$ (INput KEY şir de caractere). Funcţie. Detectează ce tastă a fost acţionată (citeşte tastatura). *Pag:* 105, 106, 113, 119, 126,130.

INPUT (INTRODUCE). Instrucţiune/comandă. Se introduc datele de la tastatură în timpul execuţiei programului. *Pag:* 36, 38, 40, 42, 43, 49, 52, 55, 57, 58, 60...etc.

INT (INTreg). Funcţie. Produce numere întregi. *Pag:* 36, 49, 57, 69, 113, 119.

INVERSE (INVERS). Instrucţiune/comandă. Inversează culoarea fondului cu cea a caracterului în cadrul unei poziţii caracter. *Pag:* 103, 106.

LEN (LENght of string, lungimea unui şir de caractere). Funcţie. Produce lungimea unui şir de caractere. *Pag:* 61, 66, 68, 114.

LET (fie, pune ceva într-o locaţie de memorie). Instrucţiune/comandă. Se atribuie o valoare unei variabile (se pune o valoare într-o locaţie de memorie). Pag: 36, 38, 39, 40, 42- 45,... etc

LINE (LINiE de program). Se utilizează cu SAVE pentru salvarea unui program cu lansare automată de la numărul de linie indicat. Pag: 109.

LIST (LISTează). Comandă/instrucţiune. Afişează lista instrucţiunilor programului din memoria calculatorului. *Pag:* 40.

LOAD (INCARCA). Comandă/instrucţiune. Încarcă un program de pe un suport extern în memoria calculatorului. *Pag:* 108, 109.

LPRINT (scrie la imprimantă). Instrucţiune/comandă. Afişează date pe hîrtia de la imprimantă. *Pag:* 72,73.

OR (SAU). Operator logic. Leagă două sau mai multe condiţii în cadrul unei instrucţiuni. Dacă cel puţin o condiţie este adevărată atunci combinaţia va fi adevărată . *Pag:* 110, 126, 129, 130.

OVER (peste). Instrucţiune/comandă. Afişează un caracter pe deasupra altuia prin suprapunere. *Pag:* 118, 119, 126, 129, 130, 131, 132.

PAPER (hîrtia, culoarea fondului). Instrucţiune/comandă. Selectează culoarea fondului pentru întregul ecran grafic sau pentru caracter, precum şi pentru punctele şi liniile desenate în cadrul unei anumite poziţii caracter. *Pag:* 80, 81, 82, 83, 84, 87-89, 91, 95,99, 101, 103, 104, 105, 110, 116, 117, 119, 131.

PAUSE (PAUZĂ). Instrucţiune/comandă. Suspendă execuţia unui program pe o perioadă de timp definită sau nedefinită. *Pag:* 111, 117, 122, 123, 124.

PI (π ; 3,14). Funcţie. Introduce valoarea lui PI (π) . *Pag*: 100, 131.

PLOT (punct). Instrucţiune/comandă. Desenează un punct într-o anumită poziţie pe ecran. *Pag:* 98, 100, 102, 104, 105, 106, 129.

POKE. Instrucţiune/comandă. Modifică valoarea unui octet dintr-o locaţie de memorie. *Pag:* 90, 92, 95, 98, 110, 115, 119, 125, 131.

PRINT (AFISEAZĂ). Instrucţiune/comandă. Afişează date pe ecran. *Pag:* 41, 41, 42, 43, 44, 45... (practic în toate programele).

READ (CITESTE). Instrucţiune/comandă. Valorile dintr-o linie DATA sînt atribuite variabilelor. *Pag:* 44, 45, 50, 69, 83, 84, 92, 94, 95, 99, 110, 115, 119, 124.

REM (REMark, comentariu). Instrucţiune. Introduce comentarii în cadrul programului. *Pag:* 36, 38, 41, 42, 43, 44, 45, 47, 49, 50, 51, 52...etc.

RESTORE (RESTAUREAZĂ). Instrucţiune/comandă. Oferă posibilitatea de recitire a datelor dintr-o anumită linie DATA. *Pag:* 124.

RETURN(REÎNTOARCE-TE). Instrucţiune/comandă. Termină o subrutină şi se întoarce în programul (principal) de unde s-a apelat subrutina. *Pag:* 73, 95, 114, 119, 126, 130, 131, 132.

RND (RaNDom number, număr aleator). Funcţie. Generează un număr aleator mai mare sau egal cu 0 şi mai mic ca 1. *Pag:* 36, 40, 49, 77, 91, 102, 119.

RUN (ALEARGĂ, dă-i drumul). Comandă/instrucţiune. Se începe execuţia unui program. *Pag:* 36, 109.

SAVE (SALVEAZĂ). Comandă/instrucţiune. Memorează (salvează) fişierele (programe sau date) pe un suport extern. *Pag:* 108, 109.

SCREEN$ (ECRAN). Se utilizează cu LOAD sau SAVE pentru a încărca şi, respectiv, salva o imagine ecran. *Pag:* 108, 109.

SIN (SINus). Funcţie. Produce sinusul unui unghi. *Pag:* 132.

STEP (pas). Se utilizează în cadrul unei linii de program FOR pentru a evidenţia pasul cu care se modifică valoarea variabilei. *Pag:* 83, 84, 95, 98, 99, 101, 103, 123, 124, 131, 132.

STOP (STOP, opreşte-te). Instrucţiune/comandă. Opreşte programul într-un anumit punct. *Pag:* 36, 43, 45, 50, 55, 61, 65, 68, 114.

TAB (TABulare). Determină coloana pe care se va afişa într-o instrucţiune cu PRINT. *Pag:* 126.

THEN (atunci). Se utilizează într-o instrucţiune IF. Dacă este adevărată condiţia ATUNCI instrucţiunea (instrucţiunile) care urmează după THEN se vor executa, iar dacă nu este adevărată se vor executa instrucţiunile începînd cu linia următoare de program. *Pag:* 36, 37, 38, 42,43... (vezi IF).

TO (pînă la). Se foloseşte într-o instrucţiune FOR pentru a determina valorile indicelui din ciclul FOR-NEXT. *Pag:* 38, 40, 42... (vezi FOR).

USR (User SubRoutine). Funcţie. Apelează o subrutină în cod sau produce adresa de început a uneia dintre cele 21 de secţiuni de memorie rezervate caracterelor grafice definit de utilizator (UDGuri). *Pag:* 90, 92, 95, 98, 115, 119, 125, 131.

VERIFY(VERIFICĂ). Comandă/instrucţiune. Verifică dacă un program a fost salvat corect, comparînd programul din memorie cu cel citit de pe un suport extern. *Pag:* 108.

POSTFAŢĂ

Jocul - o cale spre libertate

"De-ar da Domnul să se nască un om care să-i înveţe pe copiii noştri să trăiască şi să gîndească devreme, în schimb să citească tîrziu." (**Goethe**)

Criza profundă pe care o parcurge învăţămîntul contemporan, cel puţin în civilizaţia occidentală, la limita căreia vieţuim şi spre care aspirăm, este o confruntare, cu efecte distructive, între *mijloace* şi *scopuri*. Efortul pentru atingerea scopurilor educaţionale este sistematic deturnat către dobîndirea mijloacelor prin care învăţăcelul poate fi educat. Copilul, epuizat prin efortul de asimilare a mijloacelor, scapă de fapt procesului educaţional pentru a cărui împlinire nu mai rămîne disponibil.

Elevul învaţă să citească, dar nu mai are *cînd* să citească, învaţă diverse formalisme, dar nu întrevede la *ce* le-ar putea folosi, deprinde limbi străine, dar nu are cînd să le exerseze, învaţă istorie şi geografie, dar nu-şi poate face o reprezentare proprie despre ce ar putea fi un spaţiu cultural.

Mijloacele consumă prea mult din disponibilitatea copilului către formare, în primul rînd datorită faptului că *nu corespund* psihologiei şi mentalităţii vîrstelor la care se fac cei mai importanţi paşi ai procesului educaţional

Orice exerciţiu este un act gratuit, dar jocul, ca exerciţiu, este unul *explicit gratuit*. Această evidenţă îl face acceptat, chiar dorit, de mentalitatea oricărui copil, şi nu numai, deoarece conservă, chiar iluzoriu, sentimentul de libertate, de acţiune care nu constrînge din perspectiva unui scop. Cel care proiectează educaţia ca joc nu este obligat să inventeze şi să pună în practică diverse tehnici adiţionale care să suplinească lipsa de motivaţie a celui educat. Absenţa acestor procedee parazite scade riscul unor efecte nedorite, antieducaţionale.

Rigiditatea sistemelor de învăţămînt poate fi scuzată invocînd lipsa unor instrumente suficient de flexibile şi de expresive, în absenţa cărora s-au dezvoltat uneori tehnici greoaie, inexpresive, neatractive şi, în consecinţă, ineficiente.

Nu avem, se pare, nici posibilitatea de a ne întoarce la profunzimea actului educaţional care, la vechii greci, punea pe acelaşi plan muzica, gimnastica şi matematica într-o armonie pe care nu mai sîntem în stare să o declanşăm. Nu ne rămîne decît să ne punem speranţa într-o înţeleaptă folosire a celui mai avansat obiect tehnic pe care l-am inventat - calculatorul. Dacă, folosindu-l, vom debloca învăţămîntul din impasul în care se află, vom avea un argument în plus să credem că nu am inventat degeaba această maşinărie.

Va trebui să ne apropiem de acest instrument pe o cale care să fie cît mai accesibilă, mai prietenoasă, să nu recădem în eroarea de a-l gîndi numai ca pe un mijloc. **Jocul** este calea ce poate dezvălui consistent virtuţile acestui instrument ca mijloc. Tot jocul este, în continuare, mijlocul cel mai eficient de atingere a unor *scopuri* educaţionale. Putem ajunge în miezul unei problematici esenţiale, stimulaţi de relaxarea oferită prin frumuseţea gestului gratuit la care predispune jocul.

Ideea de joc nu presupune obligatoriu calculatorul, dar se împlineşte avîndu-l ca suport în procesul educaţional, datorită flexibilităţii nelimitate de care dispune cel ce gîndeşte pedagogic într-un astfel de context.

Există şi alte căi pe care ne putem apropia de calculator, căi pe care le şi putem exploata pentru a ne atinge scopurile. Dar este foarte important să ţinem cont de faptul că implicăm acest instrument în procesul de instruire a copilului, într-o perioadă în care evoluţia personalităţii lui are o dinamică foarte mare. Riscurile la care-l expunem sînt maxime şi sîntem obligaţi să gîndim acest proces cu cea mai mare grijă. Folosind jocul, ca anvelopă pentru atingerea scopului educaţional, minimizăm riscurile la care expunem procesul de formare al copilului.

Un avantaj esenţial este oferit şi de capacitatea instrumentului folosit de a se adapt diversităţii comportamentale a copiilor. Iar jocul, ca interfaţă între educat şi procesul de educare, sporeşte flexibilitatea întregului angrenaj. Capacitatea de adaptare la predispoziţiile individuale este scopul fundamental al cercetărilor pedagogice moderne. Ne oferim o şansă în plus implicînd tehnica de calcul şi maximizăm această şansă dacă folosim jocul ca principală formă de captare a interesului copilului în acest proces.

Ar fi bine să nu uităm că jocul este şi unul din acele mijloace prin care omul poate supravieţui, pe tot parcursul vieţii, tendinţelor de înregimentare rigidă pe care comunitatea i le impune. Procesul educaţional tradiţional inhibă tocmai aceste predispoziţii naturale salvatoare. Am putea atunci vedea în stimularea prin joc şi o cale de a redobîndi o parte din libertatea pe care o credeam definitiv pierdută.

martie, 1992 **Gh. Ştefan**

CUPRINS

Cuvînt înainte 5
Cîteva precizări pentru profesori 6
Cum se utilizează GHIDUL 7
Cîteva precizări privind modul de ținere a evidenței jocurilor și a informațiilor referitoare la ele 9

I. FUNDAMENTE ALE REALIZĂRII JOCURILOR PE CALCULATOR

Rezolvarea problemei 12
Un exemplu de realizare a unei diagrame de tip arbore 14
Structuri de control 16
Descrierea algoritmului 19
Scheme logice 20
Blocurile utilizate pentru scheme logice 20
Baze de numerație 28
Baza doi și octetul 32

II. ÎNCEPEM SĂ REALIZĂM JOCURI

Ghicește numărul 35
Cine este campionul? 39
Calculatorul joacă numai cu prieteni 43
Calculatorul profesor 45
Calculatorul profesor de limbi străine 46
Tabla înmulțirii 49
Calculatorul poet 49
Calculul mediilor și sortarea datelor .. 51
Program de sortare 54
Program de căutare 57
Cifrarea mesajelor și dezlegarea enigmelor 59
Descifrarea textelor codificate 61
Clasament alfabetic 63
Studiul frazei. Lingvistică cu calculatorul 66
Frecvența literelor 67
Conversații cu calculatorul 69
Calculatorul vă ajută să scrieți scrisori și să trimiteți felicitări 70
Experimente și modele cu ajutorul calculatorului 76

III. CUM SE POT REALIZA DESENE CU CALCULATORUL

Utilizarea culorilor 79
Cum se realizează programe cu culori 82
Program pentru realizarea de histograme 82
Desene pe ecran 84

Grafica de rezoluție scăzută 86
Utilizarea caracterelor grafice pentru desene 86
Programarea desenelor cu caractere grafice 88
Cum se pot crea noi caractere grafice 89
Cum se proiectează un caracter grafic 90
Generarea mai simplă a caracterelor grafice utilizator 91
Crearea de „spiriduși" de dimensiuni mai mari 96
Definirea de caractere grafice cu două culori 98
B. Grafica de rezoluție înaltă 98
Cum se pot colora figurile 100
Utilizarea ciclurilor în grafică 101
Proiectarea desenelor pe ecran 102
Calculatorul pictor 105
Ecrane grafice 107
Structura unui joc cu ecran grafic de prezentare 108
Animația 110
Mișcarea pe verticală și pe orizontală 110
Folosirea țintei 114
Cum se pot deplasa „spiridușii" într-un decor 118

IV. CUM SE POT REALIZA EFECTE SONORE ȘI MUZICA CU CALCULATORUL

Programarea sunetelor 121
Efecte sonore 123
Muzică cu calculatorul..... 124
Calculatorul-instrument muzical. 125
Calculatorul muzician 127

Să aplicăm tehnicile învățate: **Jocul „Tarzan"** 128

V. RĂSPUNSURI LA EXERCIȚIILE ȘI TEMELE PROPUSE 135

Editura Agni

Tel. 615.55.59 / 633.45.31 Fax. 210.93.36
Bucureşti CP : 30 -107

Titlu: **Provocarea algoritmilor**
(Probleme pentru concursurile de informatică)

Autor: **Victor Mitrana**

Nr. pagini: 160 **Format:** 17 x 24 cm **ISBN:** 973-96347-2-9 **Preţ :** 2800 lei

Data apariţiei: septembrie 1994

Din cuprins: Probleme pentru gimnaziu. Programe rezolvate în BASIC. Algoritmi algebrici. Algoritmi peste şiruri şi secvenţe. Algoritmi geometrici. Algoritmi cu grafuri. Bibliografie.

Despre carte: Cartea conţine atît probleme rezolvate (36) cît şi probleme propuse (10), multe din ele apărînd pentru prima dată în această lucrare. Pentru problemele rezolvate se încearcă ajungerea la cel mai performant algoritm din aproape în aproape. Algoritmii sînt descrişi într-un limbaj pseudocod, foarte apropiat de limbajul PASCAL şi care este prezen at în prima parte a cărţii. Pentru toate problemele rezolvate se dau la sfîrşitul cărţii program le în BASIC sau PASCAL, ce pot fi rulate pe orice calculator.

Se adresează: elevilor din învăţămîntul preuniversitar care cunosc sau nu un limbaj de programare, elevilor care doresc să se pregătească pentru concursuri de informatică, cadrelor didactice, precum şi studenţilor din facultăţile de profil.

Despre autor: Dnul Victor Mitrana este lector la Catedra de Informatică a Facultăţii de Matematică, Universitatea Bucureşti, doctor în matematică (teoria algoritmilor, limbaje formale). Face parte din echipa care pregăteşte lotul reprezentativ pentru concursurile internaţionale de informatică. De asemenea este membru în juriile concursurilor naţionale.

Editura Agni

Tel: 615.55.59 / 633.45.31 Fax: 210.93.36

În seria **Biblioteca de informatică**, *destinată elevilor, au apărut:*

- **Cum să realizăm jocuri pe calculator** *de Ion Diamandi* **(reeditare)**
- **Hello BASIC** *de Luminiţa State* **(epuizată)**
- **Calculatorul, coleg de bancă** *de Ion Diamandi* **(reeditare)**
- **Cum se scrie un algoritm? Simplu** *de Adrian Atanasiu*
- **Cine eşti tu, Basic?** *de Marian Gheorghe*
- **Cine ştie LOGO?** *de Ion Diamandi*
- **Minunata lume a HC-ului** *de Vlad Atanasiu*
- **CLIPPER compilator DBASE** *de Anton Mihai Cerghizan*
- **Provocarea algoritmilor** *de Victor Mitrana*
  (Probleme pentru concursurile de informatică)

Vor apărea :

- **Programarea OBJECT WINDOWS** *de Liana Cecal*
- **Quick BASIC** *de Alexandru Popovici*
- **Algoritmi fundamentali in C++** *de Răzvan Andonie şi Ilie Gârbacea*

Cărţile noastre se pot procura şi prin sistemul **Cartea prin poştă** cu plata ramburs (la primirea coletului). Pentru aceasta este suficientă trimiterea unei scrisori simple după modelul de mai jos:

Numele_____ **Localitatea**_____

Str_____ **Nr**__ **Bl**__ **Ap**__ **Judet**_____ **Cod**_____

Vă rog să-mi expediaţi prin colet poştal cu ramburs cartea (cărţile)

_____ **nr. exemplare**____

Semnătura,

Adresa noastră poştală:

Editura AGNI, CP: 30-107, BUCUREŞTI

Pentru difuzarea cărţilor noastre în şcoli, cluburi ale copiilor, cercuri de informatică etc. Editura AGNI oferă reduceri de preţuri. Astfel, pentru comenzi între 5 şi 20 exemplare, reducerea va fi de 10%. Pentru comenzi de peste 20 exemplare, reducerea va fi de 15%. Cheltuielile de expediţie vor fi suportate de Editura AGNI.

CUM SĂ REALIZĂM JOCURI PE CALCULATOR

este un ghid practic pentru utilizarea calculatoarelor de tip **Sinclair Spectrum - HC, CIP, JET, COBRA, TIM-S** atât la şcoală, cât şi în cadrul familial. Lucrarea se adresează profesorilor şi părinţilor, dar în special copiilor care se iniţiază în informatică.

Pentru învăţarea folosirii calculatorului s-a ales un subiect extrem de iubit de copii: jocurile. Cartea propune activităţi atât distractive, cât şi instructive cu calculatorul. Astfel, calculatorul devine pe rând pictor, poet, compozitor, profesor de limbi străine. Toate acestea se obţin prin intermediul celui mai comod limbaj de programare pentru tipurile de calculatoare considerate şi, în plus, recomandat începătorilor - **limbajul BASIC.**

ION DIAMANDI

A mai publicat lucrările:
Dialog cu viitorul (1988); Partenerul meu de joc, calculatorul (1988,1989,1990); LOGO - o nouă metodă de a învăţa cu ajutorul calculatorului (1991); Din spectacolul informaticii - calculatorul personal (1992); Calculatorul, coleg de bancă (1993); Cine ştie LOGO (1994) etc.

ISBN: 973-96347-4-5

Preţ: 2700 lei