


ION DIAMANDI
CUM
SĂ REALIZĂM
JOCURI
PE



CALCULATOR

ION DIAMANDI



CUM
SĂ
REALIZĂM
JOCURI
PE
CALCULATOR

Editura Agni

București 1992

CUVÎNT ÎNAINTE

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

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

Lucrarea de față se înscrie pe această linie, propunîndu-se învățarea 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 și reeditată în 1989.

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

Citeva precizări pentru profesori

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

— copiii le place atât să practice jocuri, cât și să conceapă și să realizeze singuri jocuri după ideile proprii. Astfel, practicind 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 atât turnul, cât 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 cât 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 cât 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), **Biblioteca 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 repertoriu 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 **Catag**, 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 rând, 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**.

Până acum, v-ați și întâlnit, 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 gândiți 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 decât 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 algoritmi, 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 tirziu, 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.

Problema: 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 și realizarea unei scheme logice

EXEMPLU DE REALIZARE A UNEI DIAGrame DE TIP ARBORE

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

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

1.
Fă-mi o
ceașcă cu ceai

Deocamdată, robotul nu poate să îndeplinească această sarcină, deoarece el recunoaște numai obiectele (de exemplu: ceașcă, ceainic, priză, bucătărie etc.) și 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):

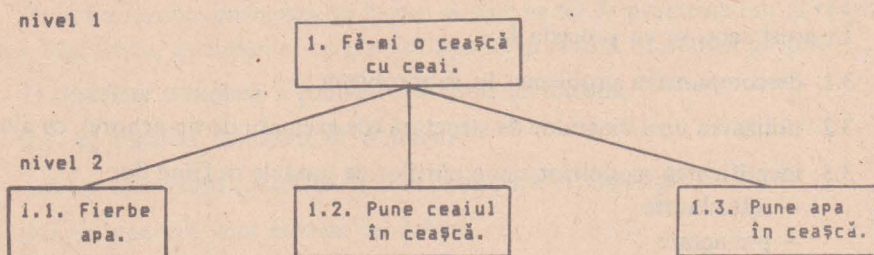
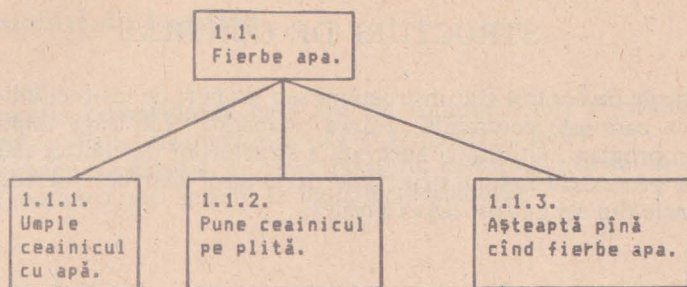


Fig. 1

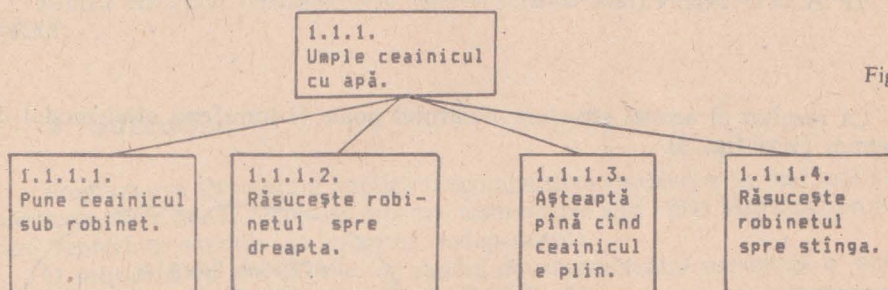
Fiecare din aceste subsarcini este încă destul de greu de îndeplinit de către robot, astfel încît fiecare trebuie descompusă la rîndul ei (vezi fig. 2).

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).

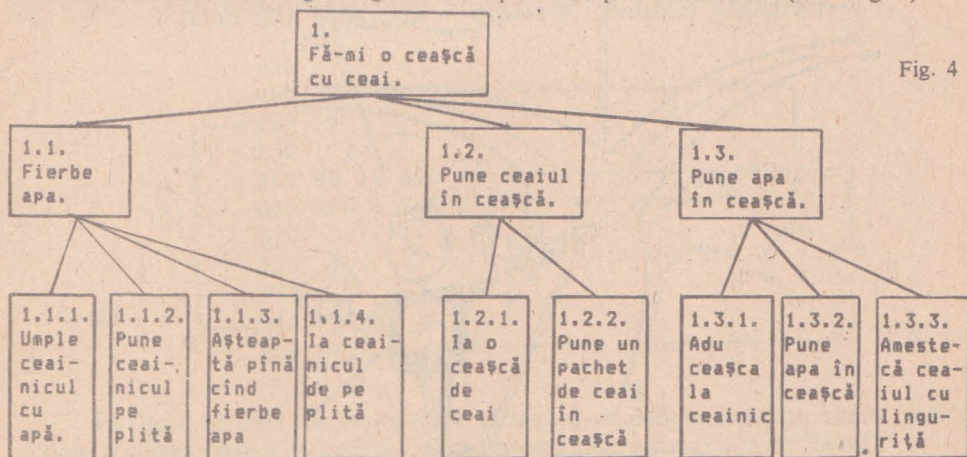
Fig. 3



Majoritatea programelor conțin activități de introducere, prelucrare de date și extragere de rezultate. Proiectind 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 = 0 THEN (face ceva)

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

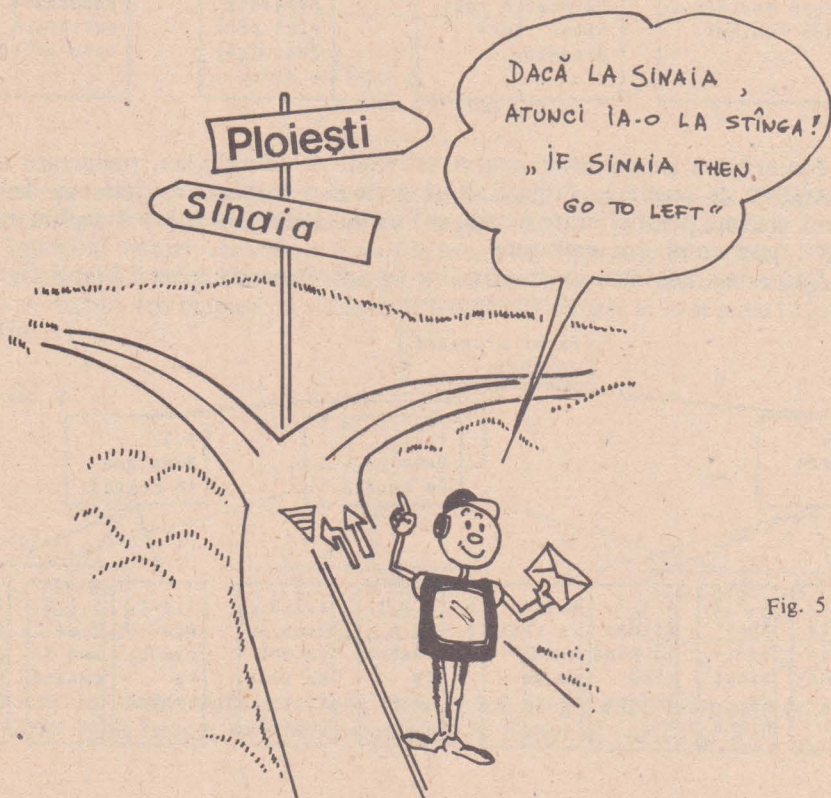


Fig. 5

2. Structuri de transfer

Acestea pot fi:

a) de transfer **necondiționat**, care este un transfer direct al controlului către un alt segment, utilizând 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.

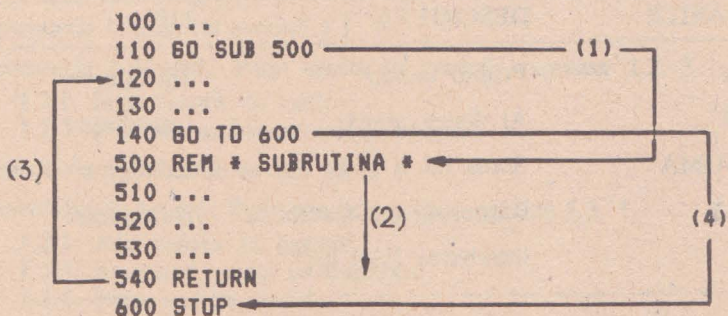


Fig. 6

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).

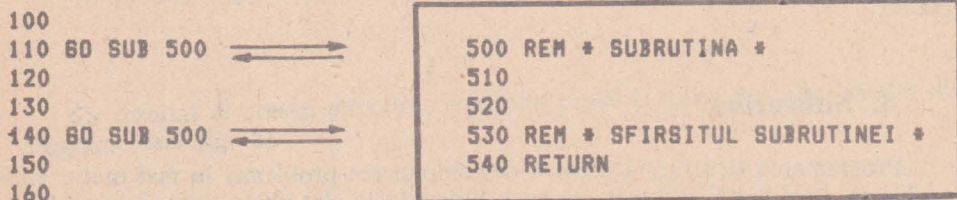
Subrutinele sînt deseori utilizate pentru repetarea unor segmente de instrucțiuni în cadrul aceluiași program.

De exemplu, să presupunem că într-un joc calculatorul pune întrebări și jucătorul răspunde. Fiecare răspuns bun va fi marcat de cîteva secvențe sonore, de felicitare și fiecare răspuns greșit de alte secvențe sonore, de avertizare. În acest caz, este indicat să se realizeze cîte un grup de instrucțiuni de sine stătător (subrutină) pentru fiecare melodie și să se apeleze cînd este cazul (răspuns bun sau greșit), nemairepetindu-se în cadrul programului secvențele de instrucțiuni. 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 și, 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:

VARIABLE	DESCRIERE	TIP
A	Primul număr	Introducere
B	Al doilea număr	Introducere
SUMA	Suma lui A și 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 pași a problemei (algoritmul) în propoziții simple;

— se realizează schema logică, arătându-se funcționarea programului de la început la sfârșit;

— se testează dacă algoritmul funcționează, 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 și va include toate descrierile din blocuri. Primul nivel al descrierii va fi titlul programului sau algoritmului. Următorul nivel va fi reprezentat de titlurile secțiunilor din program. Fiecare din aceste secțiuni va cuprinde un grup de module. Ultimul nivel al diagramei de tip arbore va fi reprezentat de instrucțiunile specifice pe care calculatorul le poate rezolva.

Utilizând o diagramă de tip arbore, se poate scrie, de exemplu, algoritmul prin care comandăm robotului o ceașcă de ceai ca o secvență de comezi pe care el le înțelege, sau algoritmul pentru realizarea pe calculator a unei agende telefonice ca o secvență de instrucțiuni care vor fi codificate în limbaj BASIC. Pentru pseudocod, vom folosi scurte propoziții în limba română și 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 ceașcă de ceai:

Comentariu ** Algoritm pentru ca robotul să facă o ceașcă 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. Așteaptă pînă cînd fierbe apa.

Comentariu * Sfirșitul sarcinii 1.1. *

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

1.2.1. Ia o ceașcă de ceai.

1.2.2. Pune un pachet de ceai în ceașcă.

Comentariu * Sfirșitul sarcinii 1.2. *

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

1.3.1. Adu ceașca la ceainic.

1.3.3. Amestecă ceaiul cu lingurița.

1.3.4. Pune apa în ceainic.

Comentariu * Sfirșitul sarcinii 1.3. *

Comentariu ** Sfirșitul algoritmului — ceașca de ceai **

EXERCITII ȘI PROBLEME

1. Descrierea în pseudocod a algoritmului pentru ca robotul să facă o cească de ceai conține câteva greșeli:
 - a) unele comenzi sînt altfel formulate față de cele din diagramă și robotul nu va fi capabil să le recunoască;
 - b) unele comenzi sînt înscrise într-o ordine greșită;
 - c) unele comenzi lipsesc din descrierea algoritmului.Corectăți descrierea algoritmului, astfel încît robotul să poată să facă o cească 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).

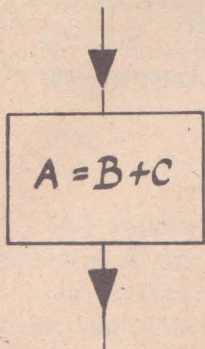


Fig. 9

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

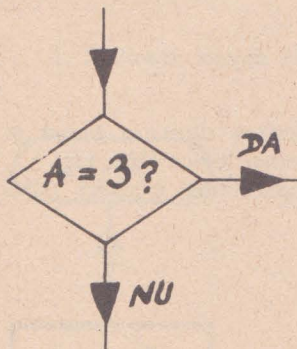


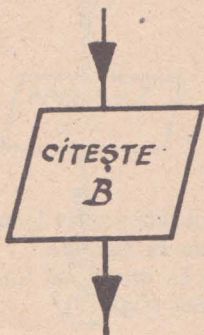
Fig. 10

blocul de decizie cu test de condiție 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 instrucțiune de tip $IF A = 3 THEN \dots$

Testul are două ramuri de ieșire, corespunzătoare pentru DA și pentru NU, după cum condiția este adevărată sau falsă, ceea ce determină fluxul în program.

Fig. 11



blocul de introducere date sau de extragere (prin afișarea rezultatelor pe ecran sau listarea la imprimantă) este reprezentat printr-un paralelogram în interiorul căruia se specifică tipul (introducere sau extragere) și datele sau rezultatele (fig. 11). De exemplu: $CITEȘTE B$ (vezi figura) sau $AFIȘEAZA A$. Acestea se vor traduce în program prin instrucțiuni, ca: $READ B$ și $PRINT A$.

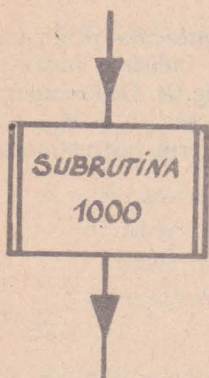


Fig. 12

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**.

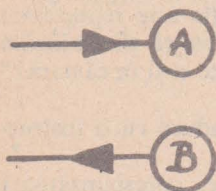


Fig. 13

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.

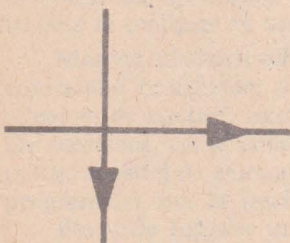


Fig. 14

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

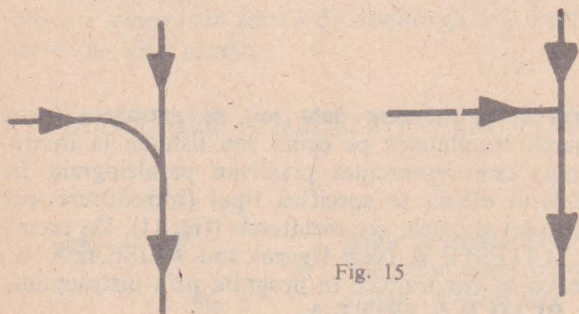


Fig. 15

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

STOP

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

Fig. 16

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

Iată câteva exemple (care reprezintă în același timp și tipuri principale) de scheme logice, împreună cu programele pe care le reprezintă:

1. Secvență simplă (fig. 17).

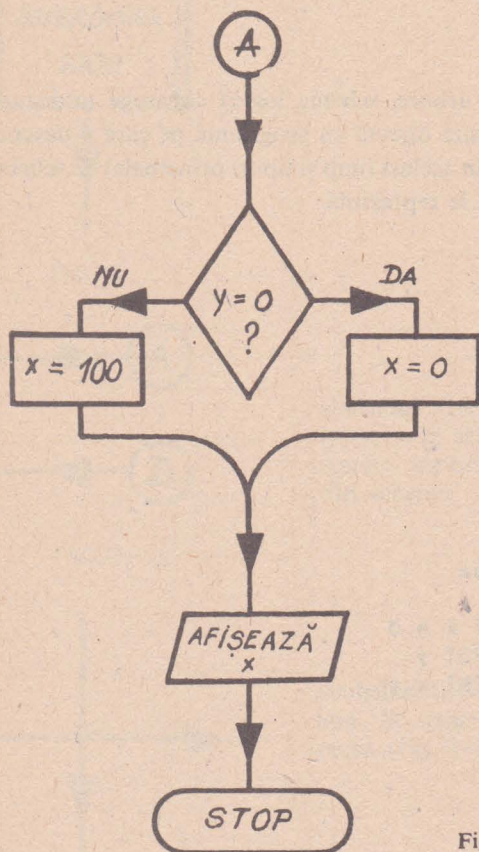


Program

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

Fig. 17

2. Decizie (fig. 18).



Program

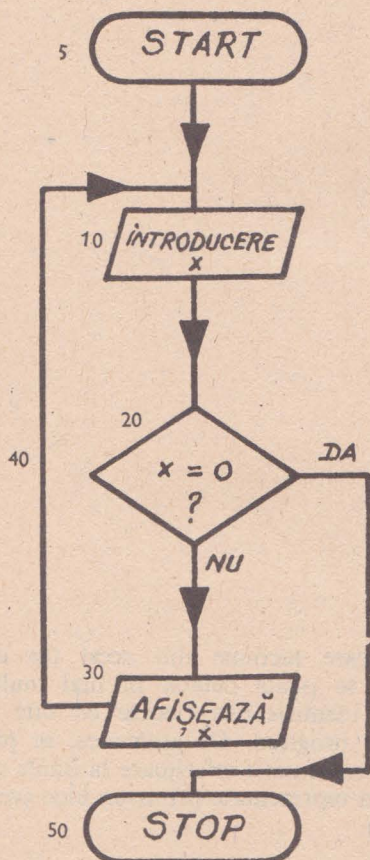
```
40 IF y = 0 THEN GO TO 70
50 LET x = 100
60 GO TO 80
70 LET x = 0
80 PRINT x
90 STOP
```

Fig. 18

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

Uneori, pentru ușurarea înțelegerii legăturii dintre schema logică și program, se pot trece lângă 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 SFIRISIT
```

Fig. 19

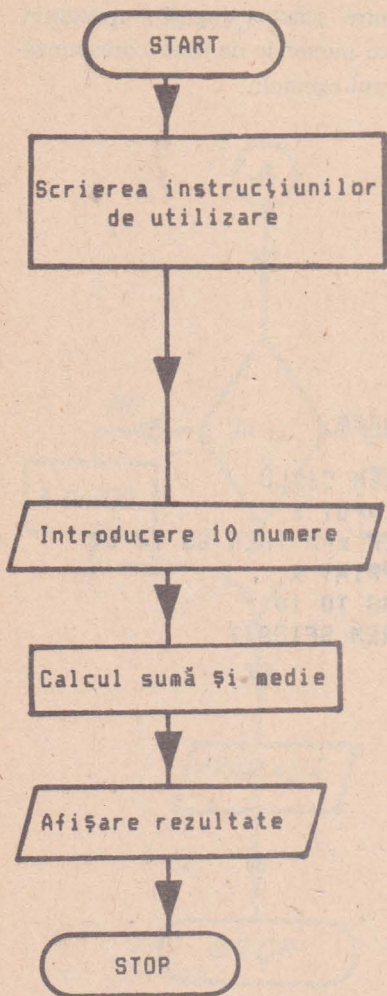


Fig. 20



Fig. 21

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.

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

... ciclul 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):

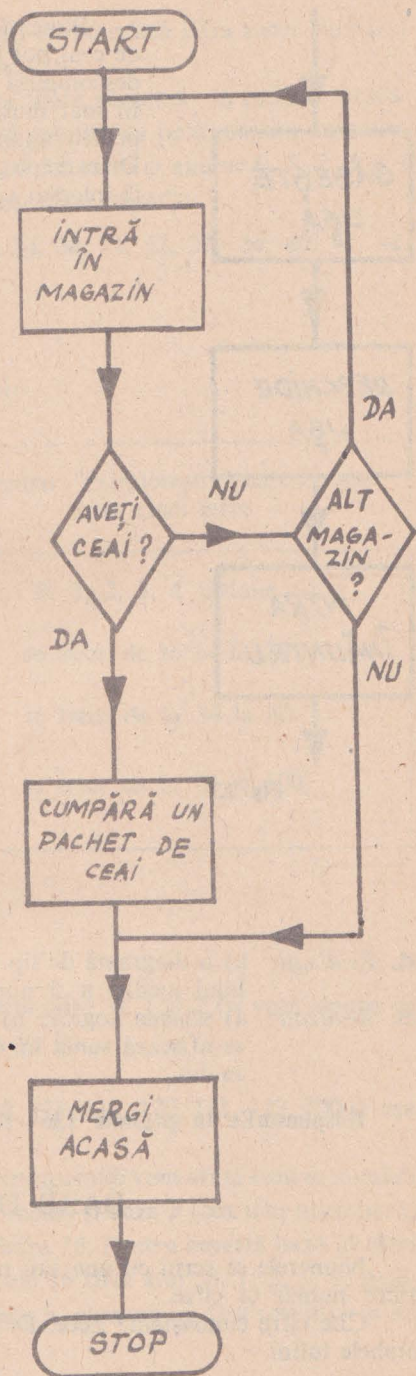


Fig. 22

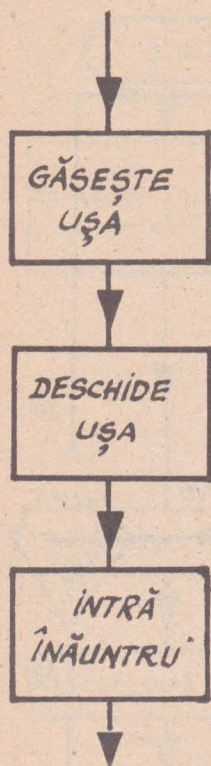


Fig. 23

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.

Câte cifre cunoaștem? Zece. De ce zece? Poate pentru că avem zece degete la ambele mâini.

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 "extraterestri" 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 nemaiavind 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	25	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	100100	1100	210	51	44	36	24
...etc.							
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 câteva lucruri:

● Mai întâi, 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).

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

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

● 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:

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

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

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

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	—	—	—	—	—	UNI- TĂȚI
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 hirtie 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

TRANSMITE

PRIMEȘTE

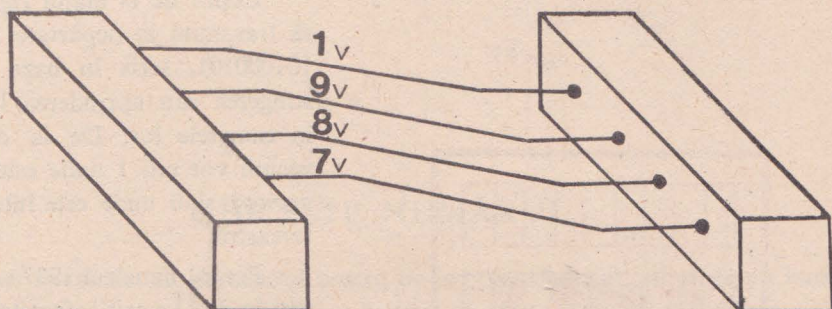


Fig. 24

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).

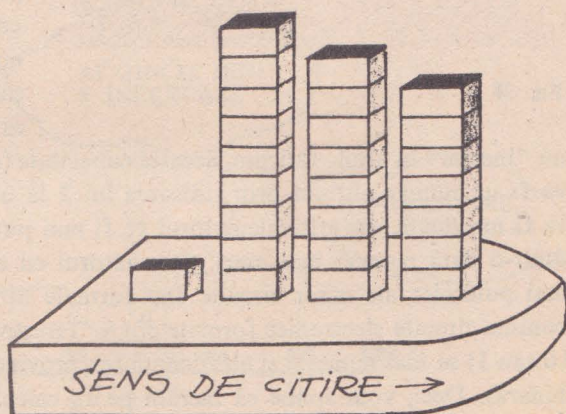


Fig. 25

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 decât 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!

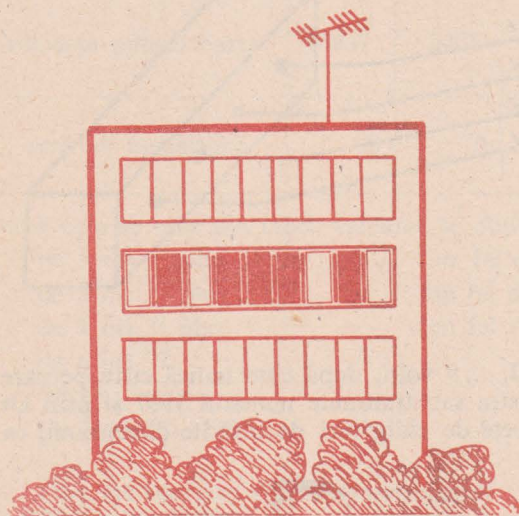


Fig. 26

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 cât numărul va fi mai mare, cu atât 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, putând efectua mai multe încercări (introduceri de numere). Dacă numărul introdus este mai mic decât 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 ÎNCEARCA!".

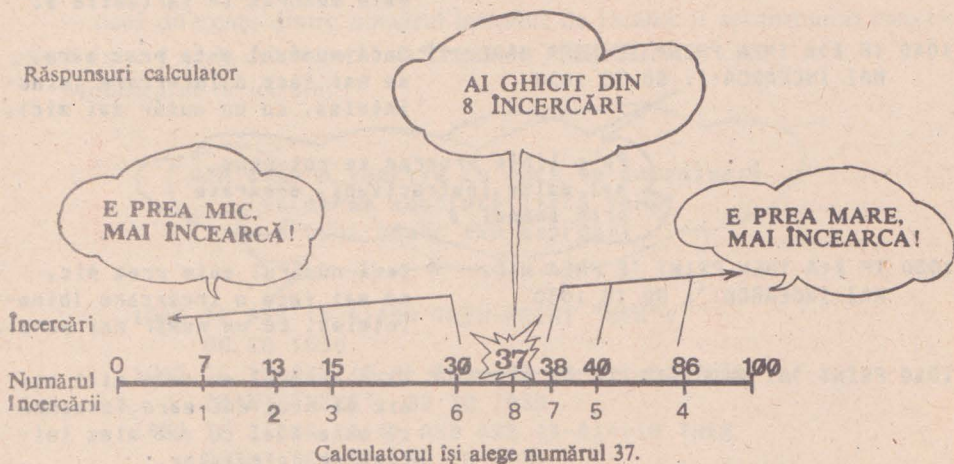


Fig. 27 GHICEȘTE NUMĂRUL

Restrângând 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 calculatorul 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 număr oarecare, întreg și pozitiv mai mic sau egal cu 100, pe care îl va memora în variabila A.

```
1030 INPUT "TASTEAZA NUMARUL ";B
```

Numărul introdus de jucător este memorat în variabila B.

```
1040 IF B>A THEN PRINT "E PREA MARE.
      MAI INCEARCA!"; 60 TO 1030
```

Dacă numărul este prea mare, se mai face o încercare (bineînțeles, 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.
      MAI INCEARCA!"; 60 TO 1030
```

Dacă numărul este prea mic, se mai face o încercare (bineînțeles, 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 inițial de calculator.

```
1061 PRINT
1070 STOP
```

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

GHICEȘTE NUMARUL

TASTEAZA NUMARUL: 7
E PEA MIC. MAI INCEARCA
TASTEAZA NUMARUL: 13
E PEA MIC. MAI INCEARCA
TASTEAZA NUMARUL: 15
E PEA MIC. MAI INCEARCA
TASTEAZA NUMARUL: 86
E PEA 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 decât 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
80 TO 1000
```

→ Dacă la întrebarea calculatorului se acționează tasta D (literă mare sau literă mică) atunci se va mai putea juca o dată.

Acum puteți juca foarte bine având ca partener calculatorul. Haideți să vedem cum putem face ca la joc să participe mai mulți copii, care să se întrecă între ei! Aveți vreo idee?

Sigur că da! V-ați adus aminte de numărarea reușitelor! Putem calcula de câte 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 0. Deci, introducem linia:

```
1025 LET I=0
```

Cînd vrea să calculăm o sumă cu ajutorul calculatorului, folosim o variabilă căreia îi dăm la început valoarea 0.

Cînd vrea 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)

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

Calculul produsului (P)

```
10 LET P=1
20 FOR I=1 TO 10
30 LET P=P*I
40 NEXT I
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, câte 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 încât să se întrecă între ei 5 jucători: Ștefan, Daniela, Oana, Petrică și Știe-Tot.

Cu rezultatele obținute de fiecare (din câte 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 câte o variabilă. Dar ce ne facem atunci când va trebui să memorăm 100 sau chiar 1000 de valori? Vom defini câte 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 prezintă un set de variabile, toate având același nume și distingându-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 definierea unei variabile de tip tablou, este indicată folosirea instrucțiunii DIM la începutul programului, ea având următorul efect:

- rezervă spațiul de memorie necesar tabloului definit:
- inițializează toate elementele tabloului cu 0.

LISTA DE INSTRUCȚIUNI
(Programul: JOC GHICESTE NUMARUL
CU CINCI JUCATORI)

1015 DIM I(5)

1016 DIM N\$(5,8)

1017 FOR J=1 TO 5

1018 INPUT "TASTEAZA-TI
NUMELE" ;N\$(J) ?
DANIELA

1020 LET A=INT (RND*100)

1030 INPUT "TASTEAZA-TI
NUMARUL" ;B

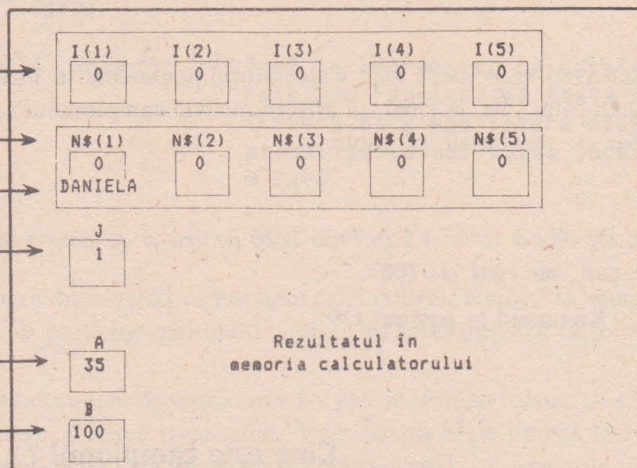


Fig. 29

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 rămân 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 oricâte 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 ele 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) = a(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 tastezi 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 încercări  
                a reușit cel de-al treilea jucător  
                să găsească numărul.
```

```
1016 DIM N$(5,8) → Se rezervă un spațiu suficient de mare  
                  pentru a memora numele celor 5 jucători  
                  care participă. Cel mai lung nume  
                  va fi de maxim 8 caractere (STIE-TOT).  
                  N$(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 câte încercări s-a
                    găsit numărul (suma), se inițializează cu
                    zero variabila care reprezintă numărul de
                    încercări, înainte de prima încercare.

1030 INPUT "TASTEAZA NUMARUL ";J
1035 LET I(J)=I(J)+1 → După fiecare încercare, numărul
                    de încercări crește cu o unitate.

1036 REM MOTORUL JOCULUI
1040 IF B>A THEN PRINT "E PREA MARE.
                    MAI INCEARCA!"; 80 TO 1030
1050 IF B<A THEN PRINT "E PREA MIC.
                    MAI INCEARCA!"; 80 TO 1030

```

Apostroful (') într-o linie PRINT va determina continuarea afișării pe rândul următor (salt pe rândul 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, neuitând 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) → Considerăm ca fiind minimul primul element din listă.
- 1080 FOR J=2 TO 5
1090 IF MIN<=I(J) THEN GO TO 1120 → Comparăm minimul cu următorul element din listă. Dacă acesta este mai mare, se trece la următoarea comparație.
- 1100 LET MIN=I(J) → Dacă este mai mic, atunci îl alegem pe el ca minim.
- 1110 LET C=J → Nu uităm să salvăm indicele minimului (J) în altă variabilă (C), pentru a ști cărui nume îi corespunde minimul.
- 1120 NEXT J → Comparările se fac pînă cînd lista se epuizează.
- 1130 PRINT N\$(C); " ESTE CAMPION"
" A GHICIT NUMARUL DIN ";MIN;
" INCERCARI" → După ce lista s-a terminat, se afișează rezultatul.

Calculatorul joacă numai cu prietenii

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

1. cerem calculatorului să solicite introducerea unei parole (să zicem "SESAM") și dacă jucătorul nu o va cunoaște, nu se va putea juca. Bineînțeles, vom avea grijă să aducem la cunoștință prietenilor care este parola. Vom adăuga programului "Ghicește 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", "DANA",  
"PETRICA", "STIE-TOT" → Lista de nume pe care calculatorul le va recunoaște.
```

```
120 DIM N$(5,8) → Se rezervă în memoria calculatorului un spațiu N$, suficient de mare pentru ca să se memoreze cele 5 nume.
```

```
130 FOR I=1 TO 5  
140 READ N$(I) → Linia 140 se repetă de 5 ori și de fiecare dată unul din nume se va pune în N$.  
150 NEXT I
```

```
160 LET K=0
```

```
170 PRINT "TASTEAZA, TE ROG, NUMELE TAU"
```

```
180 INPUT X$ → 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).
```

```
190 LET L=LEN (X$)
```

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

```
200 FOR I=1 TO 5
```

```
210 IF X$=N$(I) (1 TO L) THEN  
LET K=1
```

```
220 NEXT I → 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.
```

```
230 IF K=0 THEN GO TO 260 → 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, recunoaște persoana și jocul va putea începe.
 250 GO TO 1000

260 PRINT "NU MA JOC CU → Dacă A=0, calculatorul ignoră liniile 240 și 250 și afișează mesajul din linia 260.
 PERSOANE PE CARE NU LE CUNOSC"
 270 STOP

CALCULATORUL PROFESOR

Iată trei jocuri cu ajutorul cărora puteți să testați cunoștințele voastre sau ale prietenilor la geografie, istorie, limbi străine și matematică.

Care este capitala?

Tastați toate datele și răspunsurile cu litere mari, deoarece calculatorul tratează literele mici și 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.

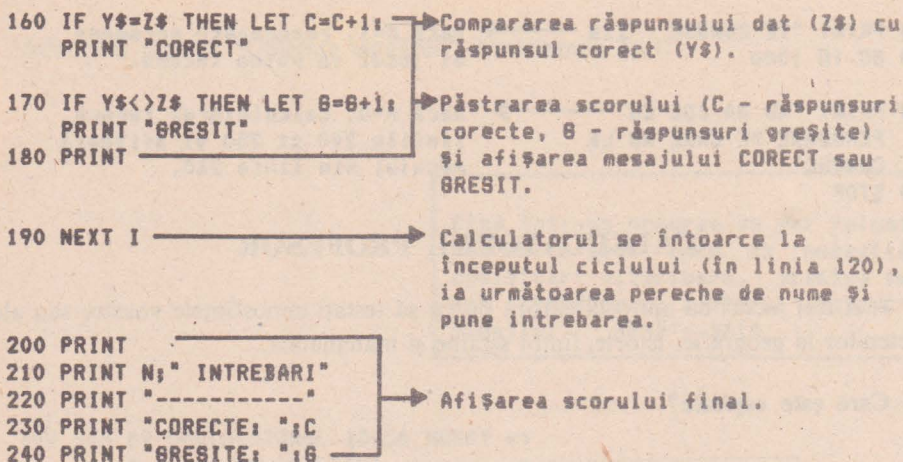
60 LET B=0 → Variabila B memorează numărul de răspunsuri greșite.

70 DATA "FRANTA", "PARIS"
 80 DATA "BULGARIA", "SOFIA" → Perechi de întrebări și răspunsuri.
 90 DATA "CHINA", "BEIJING"
 100 DATA "CANADA", "OTTAWA"
 110 DATA "ITALIA", "ROMA"

120 FOR I=1 TO N → Inceputul ciclului care repetă liniile 120-190.

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

150 INPUT Z\$ → Răspunsul este memorat în variabila Z\$.



Programul testează cunoștințele despre capitalele țărilor. Calculatorul pune o întrebare și așteaptă introducerea răspunsului, după care anunță dacă răspunsul a fost corect sau greșit (vezi fig. 30).

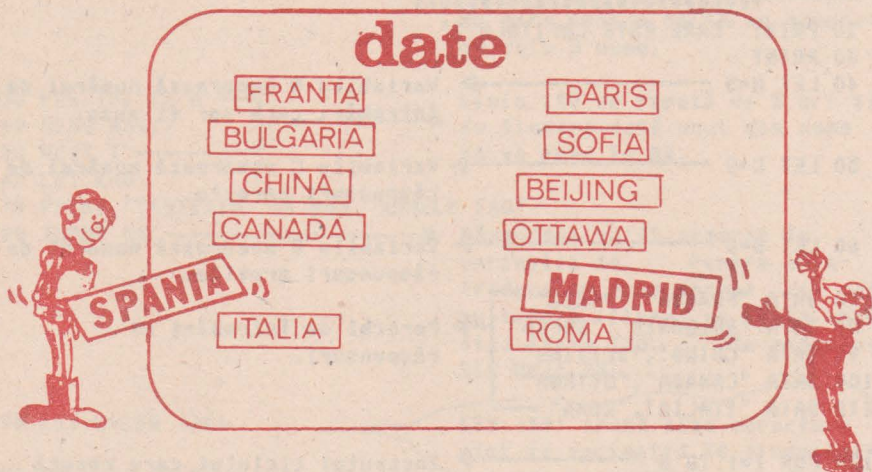


Fig. 30

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

Vor fi șapte perechi de cuvinte în test, astfel încât vom modifica pe N în 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"

Perechi de întrebări și răspunsuri.

Dacă mai adăugați perechi de întrebări și răspunsuri până la linia 140, nu uitați să modificați valoarea lui N din linia 40.

140 PRINT "CUM SE SPUNE IN FRANCEZA ";X\$;" ?"

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



FAMILII DE ANIMALE

DELFIN, MAIFER
ALBATROS, PASARE
SOPIRILA, REPTILA
TERMITA, INSECTA
MELC, MOLUSCA
CRAP, PESTE



MENCARURI TRADITIONALE

SPAGHETE, ITALIENESTI
CURRY, INDIANA
PAELA, SPANIOLA
MUSACA, GRECEASCA
SARMALE, ROMANESTI

ALTE IDEI
SI SUGESTII

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

1. Puteți face testul oricât de lung doriți (cu mai multe întrebări) adăugând mai multe linii DATA.
2. Puteți alcătui teste pentru orice domeniu: biologie, istorie, cunoștințe generale (vezi figurile 31 și 32).

DAMENI CELEBRI

HENRI COANDA, INVENTATOR
JANE FONDA, ACTRITA FILM
ELVIS PRESLEY, CINTARET ROCK
PELE, FOTBALIST
ILIE NASTASE, TENISMAN



Fig. 31

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

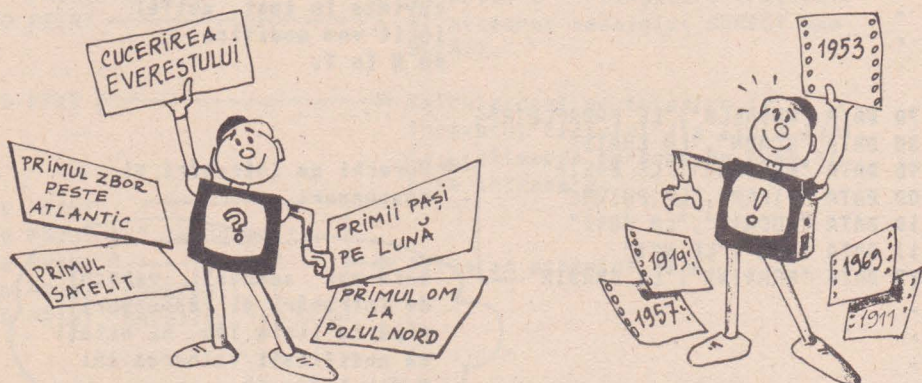


Fig. 32

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=O THEN PRINT "SLAB"
320 IF C=B THEN PRINT "JUMATATE
CORECT"
330 IF C>B 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 INVĂȚA".

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 încât 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 ÎNMULȚIRII **
```

```
*****
```

```
10 LET A=INT (RND*(10+1))
```

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

```
20 LET B=INT (RND*(10+1))
```

→ Se generează alt număr oarecare între 0 și 10.

```
30 INPUT ("CIT FACE ",A,"X",B,"=");C
```

Parantezele dintr-un INPUT determină afișarea atât a mesajelor cit și a variabilelor din interiorul lor !

```
40 IF C=A*B THEN PRINT "AI  
RASPUNS BINE": GO TO 10
```

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

```
50 PRINT "RAU, MAI ÎNCEARCA":  
GO TO 30
```

→ 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 **  
*****
```

Puteti modifica
cuvintele din program și veti
realiza alte poezii!

```
20 DATA "LACUL","PADUREA","IAZUL",  
"CRINGUL","COPACUL"  
30 DATA "ALB","BALBEN","NEGRU",  
"INVERZIT","RUBINIU"  
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"
```

→ 30 de cuvinte care vor fi
utilizate de calculator
in poezie.

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

→ Rezervarea unui spatiu de
memorie suficient pentru 30
de cuvinte de cite maxim 8
litere.

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

→ Cuvintele vor fi memorate
in variabila C\$.

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

→ Ciclul pentru numărul de
strofe.

```
140 FOR J=1 TO 4
```

→ Ciclul pentru numărul de
versuri in strofa.

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

→ K reprezintă din cite cu-
vinte va fi format versul
(intre 2 si 4).

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

→ Calculatorul alege un nu-
măr R între 1 și 30 și
scrie cuvintul corespunzător
numărului R in această
poziție in C\$.

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

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 reușit calculatorul să realizeze, în replică, un frumos tablou de iarnă.



Fig. 33

CALCULUL MEDIILOR ȘI SORTAREA DATELOR

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

```
10 REM ** CALCUL MEDIE **  
*****  
20 PRINT "CALCULUL MEDIEI"  
  
30 DIM X(50)
```

→ Puteți introduce până 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 câte 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
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.
100 PRINT "LISTEI DE NUMERE"

```

```

120 LET N=N+1
130 IF N>50 THEN 80 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 80 TO 180 → Dacă tasteați 999, calcu-
latorul merge în linia 180 și
începe calculele.

```

```

170 80 TO 120
180 REM CALCULE PENTRU MEDIE
190 LET N=N-1
200 PRINT "AI INTRODUS ";N;" NUMERE"
210 PRINT
220 LET T=0
230 FOR I=1 TO N → Ciclu pentru calculul totalului
(sunai) numerelor introduse.
240 LET T=T+X(I)
250 NEXT I
260 PRINT "TOTALUL ESTE ";T → Pentru a calcula media (A), calcu-
latorul va împărți totalul T la
numărul N
270 LET A=T/N

```

```

280 PRINT "MEDIA ESTE ";A
290 REM CALCUL MAX SI MIN
300 LET MAX=X(1)
310 LET MIN=X(1)
320 FOR I=2 TO N
330 IF X(I)>MAX THEN
LET MAX=X(I)
340 IF X(I)<MIN THEN
LET MIN=X(I)
350 NEXT I → La început primul număr este consi-
derat, în același timp, și cel mai
mare și cel mai mic număr din listă.
Apoi, calculatorul compară toate
numerele pentru a-l găsi pe cel mai
mic și pe cel mai mare.
360 PRINT "CEL MAI MARE NUMAR ESTE ";MAX
370 PRINT "CEL MAI MIC NUMAR ESTE ";MIN
380 PRINT
390 80 TO 30

```

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

Răspunsul la pagina 139.

Încercați să calculați cu ajutorul programului „Calcul medie” următoarele medii:

- media obținută la o materie pe un trimestru;
- media generală pe an;
- media notelor tuturor elevilor din clasă (media clasei);
- vârsta medie din familie;
- masa medie (în kg.) din familie;
- înălțimea medie a băieților din clasă;
- înălțimea medie a fetelor din clasă;
- înălțimea medie a elevilor din clasă;
- media timpului petrecut în fața ecranului TV pe zi.

Dacă aveți la bicicletă un indicator pentru kilometraj, puteți calcula distanța parcursă în interval de o săptămână (sau o lună) și împărțind această distanță la perioada de timp, adică la numărul de zile (sau săptămâni), puteți afla ce distanță ați parcurs în medie pe zi sau pe săptămână (vezi fig. 34).

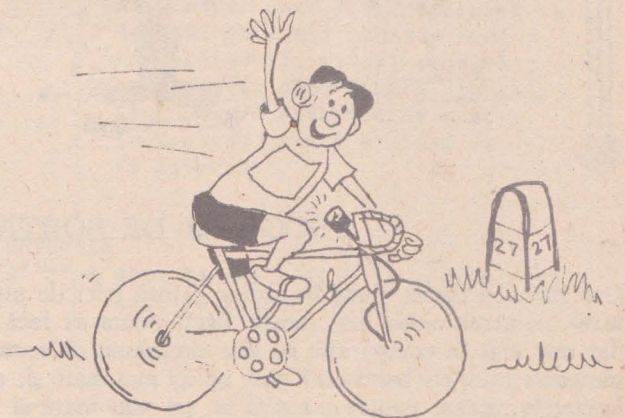


Fig. 34

METEOROLOGIE

Dacă sînteți interesați în meteorologie, puteți afla cu ajutorul calculatorului temperatura medie a aerului și căderile medii de apă pe o perioadă (lună, săptămîină, an) (vezi fig. 35). Pentru aceasta aveți nevoie de un termometru și un cilindru gradat cu ajutorul cărora veți înregistra zilnic datele respective.

La sfîrșitul perioadei, folosind programul de medie, veți obține valorile medii, precum și valorile minime și maxime ale aerului și căderile de apă din perioada respectivă.

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

Dacă aveți o rudă sau un prieten cu preocupări asemănătoare, dar care locuiește într-o localitate la o distanță mare, îi puteți cere să facă înregistrări pe care le puteți compara cu ale voastre.

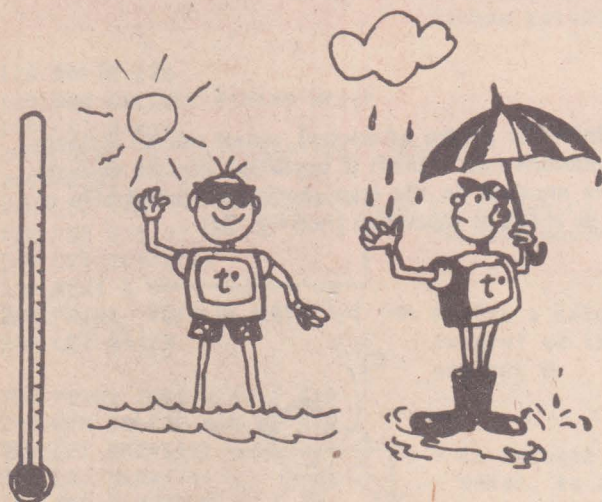


Fig. 35

PROGRAM DE SORTARE

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


```
10 REM ** PROGRAM DE SORTARE **
*****
```

```
20 DIM A(20)
```

Puteti modifica numarul 20
dacă doriți să sortați mai
multe numera.

```
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?"
```

Calculatorul întreabă cite
numere sînt în listă și me-
morează răspunsul în
variabila N.

```
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
```

Numerele sînt puse în varia-
bila de tip tablou A în ordi-
nea în care le testați.

```
170 REM AFISAREA LISTEI  
NEORDONATA
```

```
175 PRINT "LISTA NEORDONATA"
```

```
180 FOR I=1 TO N
```

```
190 PRINT A(I); " ";
```

```
200 NEXT I
```

```
210 PRINT
```

Numerele sînt afișate în or-
dinea în care au fost intro-
duse, unul după altul (cu un
spațiu între ele).

```
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
```

Calculatorul compară numerele
din listă între ele, două
cite 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.

```
320 REM AFISAREA LISTEI ORDONATE
```

```
325 PRINT "LISTA ORDONATA"
```

```
330 FOR I=1 TO N
```

```
340 PRINT A(I); " ";
```

```
350 NEXT I
```

Acum numerele sînt în ordine,
deci le putem lista.

```
360 PRINT
```

```
370 STOP
```

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 când al doilea este mai mic ca primul. Vom obține următoarea listă:

2 4 1 5 3 6

Ce observați?

În primul rând că lista nu este încă ordonată, deci va trebui să reluăm procedeul de la început, iar în al doilea rând, 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 $(N-1)$? 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 comparațiile 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.

120 REM AFISARE LISTA NEORDONATA	→	Afișare elemente lista nesortată.
125 PRINT "LISTA NEORDONATA"		
130 FOR I=1 TO N		
140 PRINT A(I); " ";		
150 NEXT I		
160 PRINT	→	Sortare elemente.
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	→	Afișare elemente lista sortată.
250 NEXT J		
260 NEXT I		
270 REM AFISARE LISTA SORTATA	→	Afișare elemente lista sortată.
275 PRINT "LISTA ORDONATA"		
280 FOR I=1 TO N		
290 PRINT A(I); " ";		
300 NEXT I		
310 PRINT	→	Numărul care va fi căutat este memorat în variabila X.
320 PRINT "TASTAȚI NUMARUL DE CAUTAT"		
330 PRINT "INTRE 100 SI 199"	→	Primul element din listă.
340 INPUT X		
350 PRINT	→	Ultimul element din listă.
360 REM CAUTARE		
370 LET L=1	→	Numărul de încercări (inițial 0)
380 LET H=N		
390 LET C=0	→	O metodă de a se împărți lista în două părți și găsirea elementului din mijloc (M).
400 LET M=INT ((H+L)/2)		
410 LET C=C+1		
420 IF X=A(M) THEN GO TO 490	→	Numărul de încercări crește cu o unitate.
430 IF L>=H THEN GO TO 520		
440 IF X>A(M) THEN GO TO 470		
450 LET H=M-1	→	Dacă numărul căutat este mai mare decât cel din mijloc, atunci se va cerceta lista din partea dreaptă (L=M+1); dacă nu, se va cerceta lista din partea stângă (H=M-1).
460 GO TO 400		
470 LET L=M+1		
480 GO TO 400		

```

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 ȘI 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 țări, numit ASCII*. În zilele noastre, majoritatea calculatoarelor utilizează codul ASCII pentru codificarea caracterelor. Puteți obține pe ecran codurile ASCII rulînd programul "Codul ASCII". Veți observa că sînt codificate chiar și caracterele grafice și cuvintele pe care le folosiți în programe (instrucțiunile, comenzile și funcțiile).

Însă, veți observa că nu sînt codificate literele Ș, Ț, Ă, Î, ele neregăsindu-se nici pe tastatură. Se vor utiliza în scopul codificării numai literele din alfabetul englez (26 de litere). Puteți să obțineți caracterul corespunzător unui anumit număr (cod) folosind într-un program funcția CHR\$ (prescurtarea de la CHARACTER). Iată programul care, ca rezultat, va afișa în dreptul fiecărui număr caracterul pe care îl codifică.

Pînă la numărul 32 codurile reprezintă caractere neafișabile. Se începe cu numărul 32, care reprezintă codificarea spațiului gol (SPACE) și 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 și coduri și apare mesajul „scroll?“, va trebui să acționați o tastă, oricare doriți (în afară de N sau BREAK) pentru a obține și restul de coduri.

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

Astfel, puteți realiza mesaje codificate în numere și puteți 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 informații).

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$
50 LET X=CODE (X$)+C
60 PRINT X
70 PRINT "TASTEAZA URMATOAREA LITERA"
80 GO TO 30

```

Se introduce fiecare literă a mesajului.

Pe ecran se va afișa codul acestei litere.

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

Atenție ! Mesajul pentru codificare se va introduce fără semne de punctuație!

```

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. Sinteți curioși cum realizează acest lucru? Programul afișează, toate cele 26 de versiuni posibile ale mesajului, iar voi veți fi capabili (sintem 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 câteva linii suplimentare și câteva modificări pe care le puteți adăuga programului "Decodificator de mesaje", prin care acesta va deveni „mai deștept”: acum programul poate recunoaște anumite cuvinte trecind prin cele 26 de versiuni posibile ale mesajului, iar dacă acest lucru se va întâmpla (calculatorul recunoaște un cuvânt), atunci se va afișa versiunea curentă a mesajului, care va fi și cea corect decodificată.

Atenție ! Nu vă pierdeți răbdarea. Pentru descifrarea mesajului, calculatorul poate lucra chiar mai mult de un minut !

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 hotărâți să alegeți 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 GO TO 280
210 IF R$="SI" THEN GO TO 280
220 IF R$="LA" THEN GO TO 280
```

→ Puteți pune pe liniile 190-220 propriile cuvinte formate din două litere care credeți că au șanse 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)

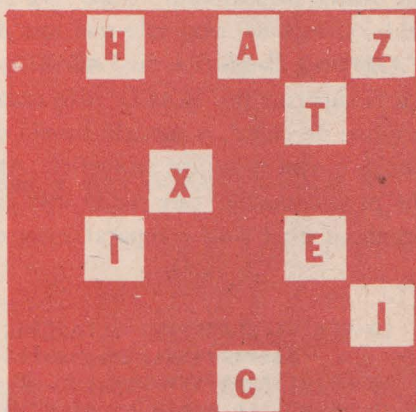


Fig. 37

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 interior (mai mic) decît cel de-al doilea. Atunci cînd amîndouă cuvintele încep cu aceeași literă, compararea se continuă asupra următoarelor două caractere, și așa mai departe, pînă este întilnită o diferență. Dacă diferența 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 (spațiul 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. Bineînțeles, "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 întii, a primului cuvînt în ordine alfabetică. Apoi, se reia procedeul fără să se mai țină 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
50 DIM N$(NC,20)
60 REM INTRODUCERE CUVINTE
70 PRINT "INTRODUCETI CUVINTUL"
80 FOR I=1 TO NC
90 INPUT N$(I)
100 NEXT I
110 REM AFISARE FISIER NEORDONAT DE CUVINTE
120 FOR I=1 TO NC
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)
200 LET POZ=I

```

→ Numărul de cuvinte.

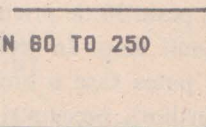


→ Rezervarea spațiului de memorie pentru fișierul de cuvinte.

→ Introducere cuvinte.

→ Afișarea cuvintelor în ordinea în care au fost introduse.

→ Considerăm primul cuvînt ca cel mai mic.

→ Indică poziția celui mai mic cuvînt.

<pre> 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) </pre>	 <p>Ciclul de căutare al celui mai mic cuvânt.</p>
<pre> 270 LET N\$(POZ)=N\$(I) 280 LET N\$(I)=M\$ 290 NEXT I 300 PRINT N\$(NC) 310 STOP </pre>	 <p>Schimbarea ordinii cuvintelor.</p> <p style="margin-top: 20px;">  </p> <p>Afișarea ultimului cuvânt.</p>

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 aveți peste 30 de prieteni și cunoștințe, aranjarea în ordine alfabetică vă va lua foarte mult timp. În același fel, puteți să vă puneți în ordine cărțile din bibliotecă (se va putea face o ordonare alfabetică atât după numele autorului, cât și după titlurile cărților), precum și să vă țineți în ordine casele cu programe, realizând 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 condiția separării fiecărui cuvânt printr-un spațiu (și numai unul), așa cum se și face de obicei în practică. O excepție este luată în considerație: liniuța de unire, deoarece aceasta separă practic două cuvinte. La introducerea textului nu trebuie plasat spațiu decât după semnul de punctuație: dacă acesta ar avea de o parte și de alta spațiu, atunci ar fi considerat un cuvânt.

```

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)
130 IF C$=" " OR C$="-" THEN
    LET N=N+1
140 NEXT I → Analiza frazei caracter
               cu caracter.

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 cuptorul 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 !



Fig. 39

FRAZA ARE 86 CUVINTE

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 (minuscul).

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

Răspunsul la pagina 140

10 REM ** FRECVNTA 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 APARIŢIEI 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 câteva 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

Încercați cu texte mai lungi pentru a obține cele mai reprezentative rezultate pentru frecvența literelor din limba română. Știați că după ce s-au realizat mai multe teste de acest fel, pe baza rezultatelor obținute (frecvențele de apariție ale literelor) s-a proiectat aranjarea clapelor (tastelor) la mașinile de scris și la tastaturi, astfel încât literele cu cele mai mari frecvențe de apariție să fie cât mai apropiate de degetele cu care se tastează mai ușor?

CONVERSAȚII CU CALCULATORUL

În jocul următor, calculatorul va părea a fi dotat cu inteligență proprie, putînd face conversații (uneori inteligente) cu oricare din voi. Totuși, din program se poate observa că el nu recunoaște decît cuvintele DA și NU, iar pentru restul de răspunsuri la mesajele tastate de interlocutor, el alege la întimplare cuvintele dintr-o listă de date. Din această cauză, uneori, răspunsurile nu se potrivesc, pîrînd stranii.

Folosiți numai litere mari !

```
10 REM ** CONVERSATIE **
*****
20 DATA "DE ACORD","DA","NU","SIGUR"
30 DATA "BINE","OH!","DACA SPUI TU!"
40 DATA "ESTE DARE 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$="NU" THEN GO TO 370
230 IF X$="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 propoziții care vor fi alese la întimplare drept răspuns.

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

Memorarea șirurilor de caractere (a cuvintelor).

Inceput politicos de conversație.

Se introduce replica de către interlocutor.

Se alege drept răspuns unul din cele 14 șiruri 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 corecetați 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, bineînțele, un casetofon.

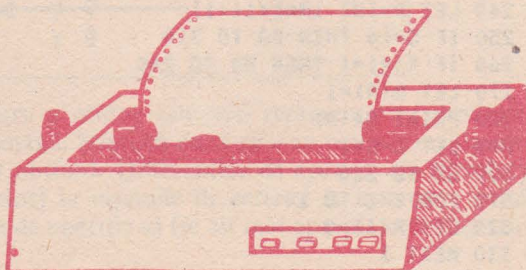


Fig. 41

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 sfârșitul rândului. Programul realizează automat o reordonare a cuvintelor, astfel încât 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. Intra o am jucat foarte mult REVERBI contra calculatorului 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 SCOLA, 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

*
*
**
*

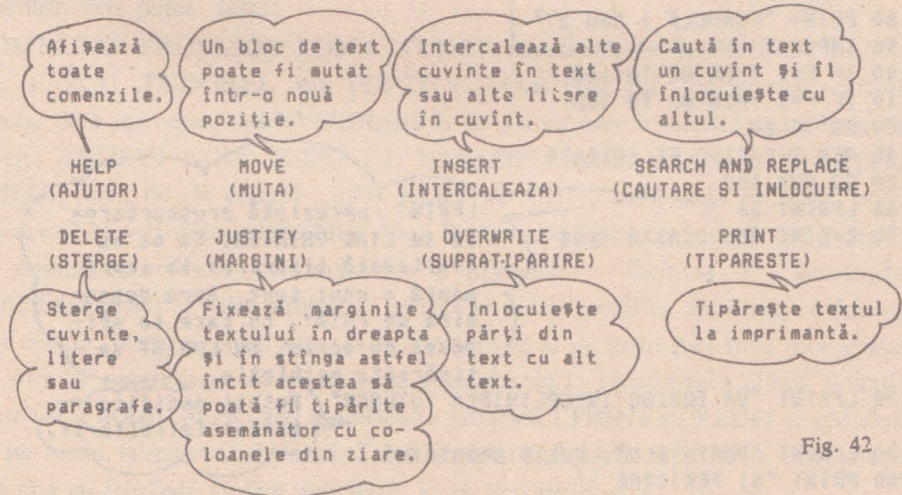


Fig. 42

țite la sfârșitul rîndului. Cursorul îl puteți așeza oriunde în text, mișcîndu-l la dreapta, stînga, sus sau jos cu ajutorul săgeților.

Puteți indica în orice moment calculatorului cum să aranjeze sau să pună textul în pagină, sau cît de lungi trebuie să fie rîndurile. De obicei, puteți 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 puteți observa din figura nr. 42, care redă ecranul TV după introducerea unui text pentru o scrisoare. Pentru comenzi nu apar (de obicei) explicații pe ecran; pentru ele se poate consulta un manual de utilizare.

19. *V-au necăjit cele cîteva greșeli pe care le-ați 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, Sînteți ajutați (de data aceasta nu de către calculator) prin marcarea cu semnul * rîndurilor în care se găsesc greșeli. Puteți indica, la fiecare, ce opțiuni (comenzi) trebuie acționate?*

Răspunsul la pagina 140.

Iată un program care vă va ajuta să scrieți felicitări în diverse ocazii, deși, 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 inițiale.

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

Puteti modifica textul felicitării.

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

```
*****
*** ** *  *  *** * ***** *  ***  ***
*  *  *  *  *  *  *  *  *  *  *  *
*** ** *  *  *  *  *  * ***** **  ***
*  *  *  *  *  *  *  *  *  *  *  *
*  *** ** *  *** *  *  *  *  *  *  *  ***
*****
```

PROFESOR TUDOR TOMA

CU OCAZIA ANULUI NOU,
VA DORESC INDEPLINIREA TUTUROR
DORINTELOR, MULTA SANATATE SI
FERICIRE !

RADU STEFAN

```
*****
*** ** *  *  *** * ***** *  ***  ***
*  *  *  *  *  *  *  *  *  *  *  *
*** ** *  *  *  *  *  * ***** **  ***
*  *  *  *  *  *  *  *  *  *  *  *
*  *** ** *  *** *  *  *  *  *  *  *  ***
*****
```

LUI PETRICA SASU

CU OCAZIA ZILEI DE NASTERE
ITI DORESC NUMAI NOTE BUNE,
SANATATE SI VOIE BUNA !

PRIETENUL TAU BOGDAN

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,
Îți mulțumesc mult pentru cadoul
pe care mi l-ai făcut de ziua
mea. Trenulețul este foarte frumos
și mă joc deseori cu el.

Nepotelul tău,
Dorel

* INVITATIE *

Draga Doina,
Pe data de 23 mai împlinesc
vîrsta de 14 ani. Cu această
ocazie organizez o mică pe-
trecere duminică 28 mai la
ora 17, la care mi-ar face
plăcere să vii.

Ne vom amuza cu cîteva
jocuri noi și sper că vom
dansa. Adresa mea este:
Bd. A.I. Cuza nr.53, et.2,
ap.3.

Prietena ta,
Maria

Fig. 44

EXPERIMENTE ȘI MODELE CU AJUTORUL CALCULATORULUI

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

Programele utilizate de oamenii de știință sau alți cercetători sînt deseori foarte complexe și funcționează pe calculatoare foarte puternice. Calculatoarele sînt ideale pentru realizarea de modele care utilizează ecuații matematice, deoarece ele pot face calcule foarte rapid. Iată cîteva exemple în care se arată cum sînt utilizate aceste programe:

MODELAREA APARIȚIEI 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 nașterea Universului. Modelul respectiv se bazează pe teoria conform căreia Universul s-a născut printr-o explozie uriașă numită big bang. Programul folosește ecuații matematice pentru a reprezenta ce s-a întîmplat începînd cu prima secundă după explozie, trasînd, apoi, structura Universului de atunci și pînă acum.

Prin compararea rezultatelor modelului privitoare la structura Universului cu măsurătorile și observațiile actuale, oamenii de știință pot desprinde concluzii referitoare la veridicitatea teoriei.

PROIECTAREA MAȘINILOR

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

De asemenea, forma autoturismului este proiectată în funcție de rezultatele simulărilor pe calculator a rezistenței pe care autoturismul o opune cînd este în mișcare, la diferite viteze, în condiții de vînt din diferite direcții, etc. Sînt proiectate piese noi sau componente din alte materiale (de obicei mai ușoare), în funcție de rezultatele modelelor.

Iată un program simplu care realizează modelul aruncării unei monede. Puteți experimenta cu ajutorul său și urmări rezultatele care se obțin atunci când 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 inițializează variabilele care reprezintă de câte ori s-a obținut stema și respectiv banul.

Calculatorul generează un număr cuprins între 0 și 1.

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

Atunci când rulați programul, veți introduce numărul de aruncări ale monedei. Programul va realiza modelul aruncării, rezultând numărul de apariții a stemei și a banului.

De ce am comparat numărul aleator generat cu 0,5? Deoarece atunci când se aruncă o monedă se poate obține în egală măsură stema sau banul, șansa (probabilitatea) de apariție a fiecărei fețe fiind, deci, de 1/2.

20. Realizați următorul experiment: încercați modelul (programul) pentru 10, 100, 1000 de aruncări ale monedei și înregistrați datele obținute într-un tabel, ca cel de mai jos:

Număr de aruncări	Rezultate			
	Obținerea stemei		Obținerea banului	
	Numărul (frecvența de apariție)	Procentul din numărul de aruncări (%)	Numărul (frecvența de apariție)	Procentul din numărul de aruncări (%)
10				
100				
1000				

Ce observați în legătură cu datele obținute?

Răspunsul la pagina 141.

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

Încercați să aruncați moneda de 100 de ori și să scrieți rezultatele obținute.

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

Experiment	Calculator
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	
63	
64	
65	
66	
67	
68	
69	
70	
71	
72	
73	
74	
75	
76	
77	
78	
79	
80	
81	
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	
92	
93	
94	
95	
96	
97	
98	
99	
100	

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 și pentru colorarea caracterului propriu-zis.

Următorul tabel arată culorile și codurile utilizate de calculator în acest scop. Nu este necesar să memorați 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	roșu	RED
3	purpuriu	MAGENTA
4	verde	GREEN
5	bleu	CYAN
6	galben	YELLOW
7	alb	WHITE

Dacă nu aveți la dispoziție un TV sau un monitor color, culorile vor fi reprezentate prin diverse nuanțe de gri (ordonate de la închis spre deschis, corespunzător codurilor 0 - 7).

Cind 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 acționind 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 acționind 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 acționind 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

Introduceți următorul program:

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

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

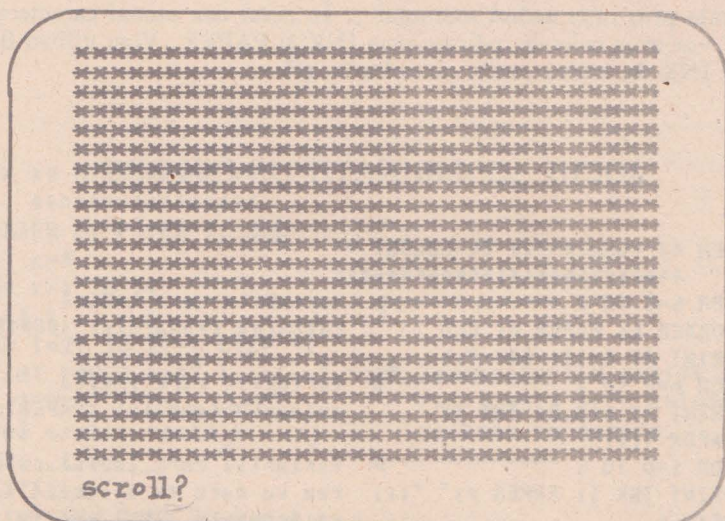


Fig. 46

Opriti programul cu **BREAK** și introduceți citeva comenzi pentru culori.
De exemplu:

```
BORDER 4      Nu uitați să introduceți  
PAPER 2      CR după fiecare comandă.  
INK 7
```

Rulați din nou programul "Testarea culorilor". Observați acum culorile (dacă folosiți un TV color): verde pentru chenar, albă pentru stelute și roșie pentru fond?

Opriti din nou programul și rulați-l încă o dată după ce introduceți alte comenzi pentru culori, de exemplu: **BORDER 3, PAPER 5 și 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, rulind 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 câteva 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).

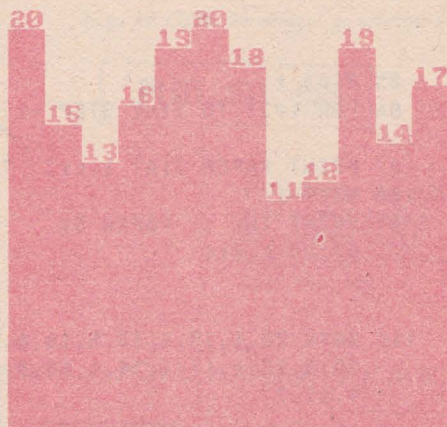


Fig. 47

```

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 1,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 1,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 atât temperatura maximă, cât și cea minimă dintr-o zi.

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

DESENE PE ECRAN

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

A. **utilizând 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. **utilizând ecranul grafic** de 256×176 de puncte, deci puncte, linii, cercuri, etc. Amândouă 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 având ca unitate de reprezentare caracterul (se pot afișa 32 de caractere pe rând 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, având 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 afișare a ecranului.

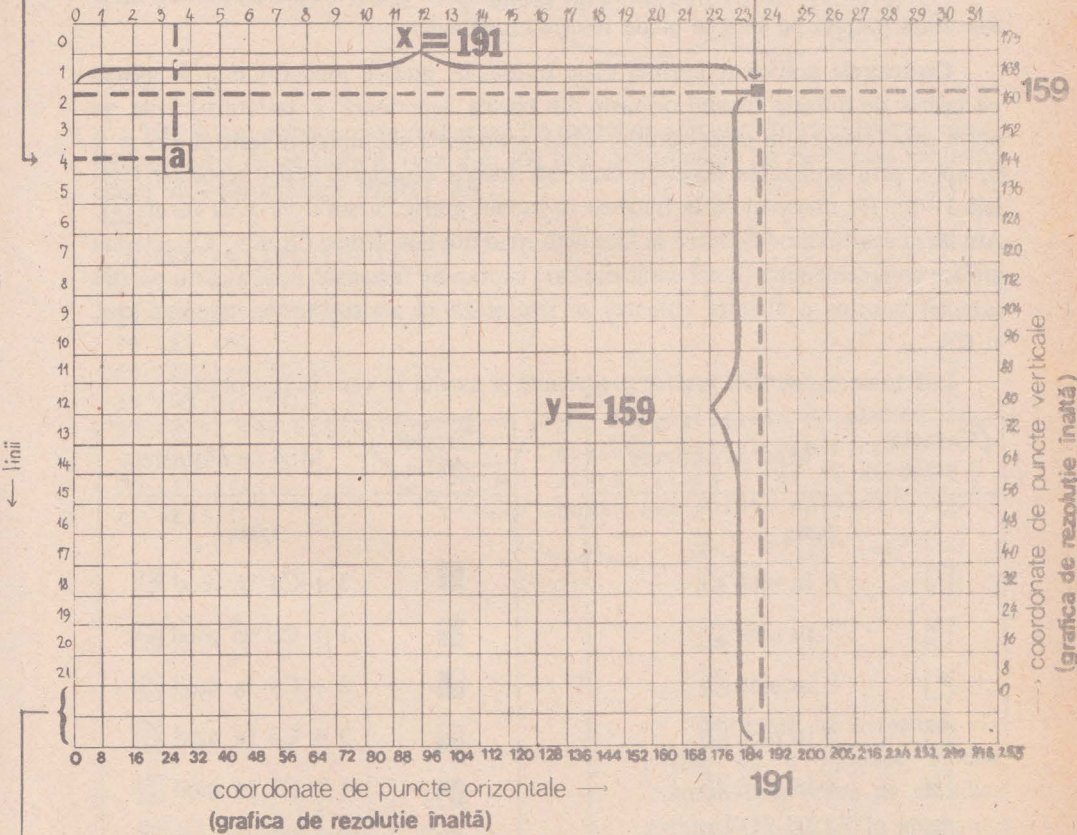
Un exemplu de utilizare pentru grafica de rezoluție scăzută: un caracter pe linia 4 și coloana 3.

PLOT 191, 159

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

PRINT AT 4, 3; "a"

→ coloane (grafica de rezoluție scăzută)



cele două linii din partea de jos a ecranului nu se pot utiliza în mod normal cu PRINT și 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, linia 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 dungii 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 . 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 
	1 în mod 
	2 în mod 
	3 în mod 
	4 în mod 
	5 în mod 
	6 în mod 
	7 în mod 

Caracter grafic	Mod de obținere
	taste
	8 + CS în mod 
	1 + CS în mod 
	2 + CS în mod 
	3 + CS în mod 
	4 + CS în mod 
	5 + CS în mod 
	6 + CS în mod 
	7 + CS în mod 

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

Iată un program care va realiza umplerea tuturor pozițiilor de caractere de pe ecran cu culori alese la întâmplare, 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 obține acest caracter, se acționează tasta CS și 9 și, apoi, se apasă tasta 8 (tot cu CS). Nu uitați, apoi, să ieșiți din modul grafic tot cu CS și 9.

Iată și un program care realizează un model de curcubeu tot cu ajutorul caracterului grafic de pe tasta 8, (vezi fig. 49) prin poziționarea lui în diverse locuri pe ecran cu instrucțiunea **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 1,c+x;"█"
60 NEXT c
70 LET x=x+1
80 NEXT l
```

În grafica de rezoluție scăzută, primul număr reprezintă linia, iar al doilea număr reprezintă coloana.

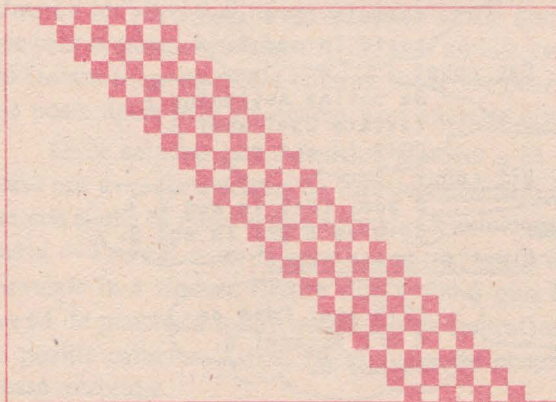


Fig. 49

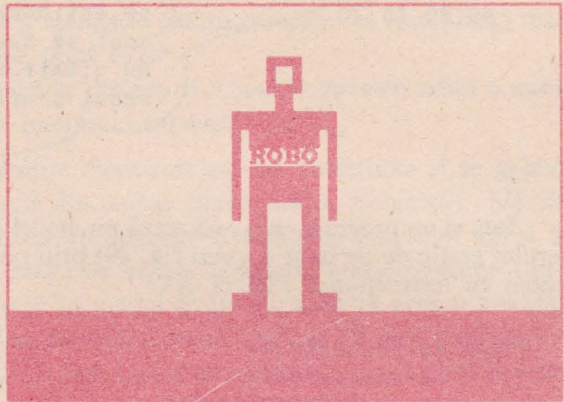
Observați că o metodă eficientă de a produce modele colorate este aceea de a utiliza cicluri **FOR-NEXT** în programe. În exemplul nostru, ciclurile **FOR-NEXT** au fost utilizate pentru a modifica culorile și pozițiile produse de **INK** și, respectiv, **AT**.

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 rânduri. 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 1=7 TO 10: PRINT INK 5;
  AT 1,13; "███████":NEXT 1
    
```

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.

Șit 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.

```

55 PRINT INK 6; PAPER 0; AT 8,14;
   "ROBO"
60 PRINT INK 2; AT 11,13; "█"
70 FOR l=11 TO 15: PRINT
   INK 6; AT 1,14; "█ █":
   NEXT l
80 PRINT INK 3; AT 16,13; "█";
   TAB 17; "█"
90 FOR l=17 TO 21: FOR c=0 TO 31
100 PRINT INK 4; AT l,c; "█"

110 NEXT c: NEXT l

```

Nume robot

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

Picioare robot

CS+8 apoi 8 de 2 ori
și apoi iarăși CS+8,
toate în mod 8.

CS+8 de 2 ori în mod 8.

CS+8 în mod 8.

TAB este utilizat pentru a poziționa un caracter în lungul liniei curente. Numărul care urmează după TAB specifică coloana unde se va afișa următorul caracter.

21. Scrieți un program care să realizeze cu ajutorul caracterelor grafice afișarea pe ecran a cuvântului *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 \times 8 puncte fiecare = 64 de puncte în total). Acest tablou (pătrat) de 8 \times 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 întâi desenați o grilă de 8×8 pătrățele, ca în fig. 51. Apoi, innegriți câteva din pătrățele, astfel încît întreaga grilă să reprezinte ceva, de exemplu: un păianjen (vezi fig. 52).

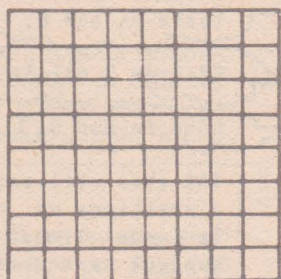


Fig. 51 Grila 8×8 .

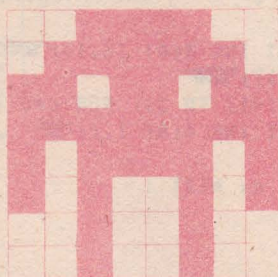


Fig. 52 Păianjen în grilă.

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:

```

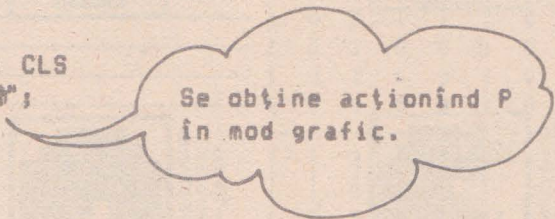
5 REM ** CARACTER PAIANJEN **
  *****
10 POKE USR "p",BIN 00111100 — linia 0 —> POKE modifică
11 POKE USR "p"+1,BIN 01111110 — linia 1 —> conținutul unei
12 POKE USR "p"+2,BIN 11011011 — linia 2 —> adrese de memo-
13 POKE USR "p"+3,BIN 11111111 . —> rie, în cazul
14 POKE USR "p"+4,BIN 10111101 . —> nostru 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.
    
```

Linile (rîndurile)
cite 8 de 0 și 1
pentru caracterul "p".

Rulați acum acest program și, apoi, acționați tasta P în modul grafic \square (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; " ";  
40 GO TO 30
```



Se obține acționând P
în mod grafic.

Acum apar păianjeni de diverse culori pe întregul ecran.

Atenție! Cind vă proiectați propriile caractere grafice, nu uitați că nu le veți putea vedea pe ecran decât după ce rulați programul care le definește. Până 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 utilizând numere zecimale care vor fi citite cu READ și DATA. Mai întâi, trebuie transformate cele 8 numere binare, formate din 0 și 1, în numere zecimale. Acest lucru se poate face introducând 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 rând 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.

2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰		Total rînd:
0	0	1	1	1	1	0	0	→	128×0+64×0+32×1+16×1+8×1+4×1+2×0+1×0= 60
0	1	1	1	1	1	1	0	→	126
1	1	0	1	1	0	1	1	→	219
1	1	1	1	1	1	1	1	→	255
1	0	1	1	1	1	0	1	→	189
1	0	1	0	0	1	0	1	→	165
1	0	1	0	0	1	0	1	→	165
0	0	1	0	0	1	0	0	→	36
↑	↑	↑	↑	↑	↑	↑	↑		
128	64	32	16	8	4	2	1		

Se obțin, bineînțeles, aceleași valori ca și la transformarea cu BIN.

Avînd acum la dispoziție aceste numere, putem să le introducem prin intermediul instrucțiunilor READ și 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

```

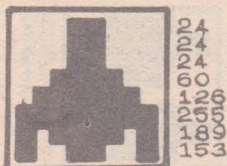
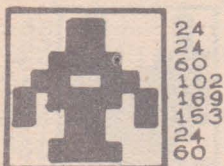
Programul va memora oricare 8 numere zecimale (între 0 și 255), care vor crea un caracter. În acest scop se modifică p din linia 30 cu orice literă doriți și se vor introduce numerele pe linia 50, în ordine, separate prin virgule.

Pentru a obține caracterul dorit, se va acționa tasta, corespunzătoare literei, în modul grafic.

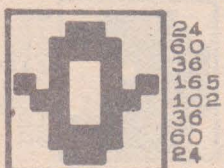
La ce folosește crearea de noi caractere grafice în jocuri?

Bineînțeles, pentru a crea diverse personaje (să le numim spiriduși) și obiecte (asteroizi, rachete etc.) care apoi vor fi deplasate pe ecran. În figura 54 aveți la dispoziție o întreagă colecție de spiriduși împreună cu valorile asociate pentru crearea lor. De asemenea, se mai pot folosi pentru desenarea unor figuri (piese), de exemplu: piesele de șah, sau pentru crearea literelor specifice limbii române (care ați observat că nu se pot obține în mod obișnuit de la tastatură): ă, â, î, ș, ț.

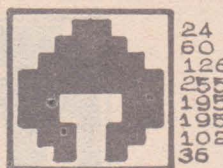
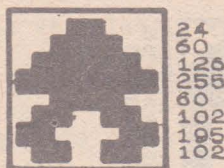
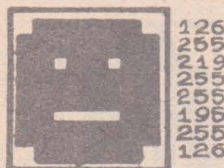
Rachete



Canoe



Extraterestri



Morișcă

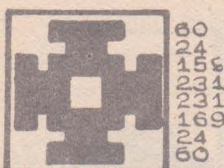


Fig. 54 b.

Iată și un program care desenează pe ecran tabla de șah, împreună cu piesele așezate ca pentru începutul unei partide:

```

5 REM ** TABLA DE ȘAH **
  *****
10 FOR x=1 TO 6
20 READ a$
30 GO SUB 500

```

→ Sînt 6 feluri de piese: pioni, ture, cai, nebuni, regi și regine.
 → Apelul subrutinei pentru definiția 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;"X1QX3Q1X" → Punerea pieselor pe tablă.

```

Aici se poate insera codul unei culori pentru a se deosebi piesele pentru un 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.

Tastele T,C,N,Q,K,N,C,T toate în mod 8.

```

100 PRINT AT 8,12;"11111111" → De 8 ori tasta P în mod 8.
110 PRINT AT 13,12;"11111111" → De 8 ori tasta P în mod 8.
120 PRINT AT 14,12;"X1QX3Q1X" → Tastele T,C,N,K,Q,N,C,T
    toate în mod 8.

```

```

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 8.
560 DATA "t",0,84,124,56,56,124, → Valorile pentru definirea turei
    124,0 → Ea se obține cu T în mod 8.
570 DATA "c",0,16,56,120,24,56, → Valorile pentru definirea calu-
    124,0 → lui. El se obține cu C în mod 8.
580 DATA "n",0,16,40,68,108,56, → Valorile pentru definirea nebu-
    124,0 → nului. El se obține cu
    N în mod 8.
590 DATA "k",0,16,56,16,56,68,56,0 → Valorile pentru definirea
    regelui. Se va obține cu
    K în mod 8.
600 DATA "q",0,84,40,16,108,124, → Valorile pentru definirea
    124,0 → reginei. Se va obține cu
    Q în mod 8.

```

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

- un omuleț
- o mașinuță

- c) litera ă
- d) litera î
- e) litera ș
- f) litera ț

B. Indicați secvența de taste ce trebuie acționate după ce ați definit caracterele grafice de la exercițiul A, pentru a obține pe ecran afișarea următorului text:

Un omuleț ✖ într-o mașinuță 🚗

Presupunem că am ales pentru omuleț litera o; pentru mașinuță litera m; pentru ă, litera a; pentru î litera i; pentru ș litera s; pentru ț litera t.

Răspunsurile la pagina 142.

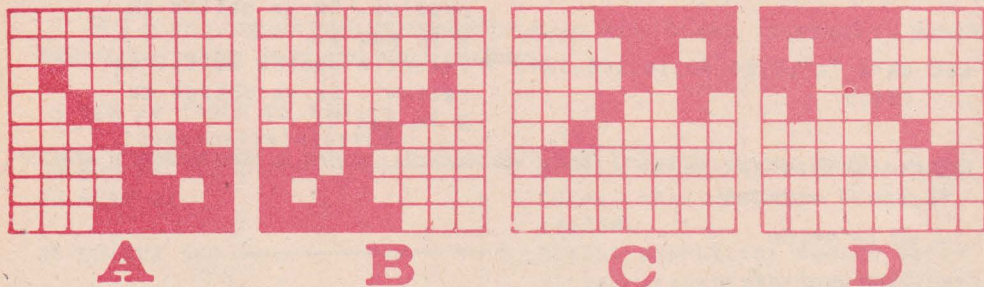
CREAREA DE SPIRIDUȘI DE DIMENSIUNI MAI MARI

Sigur se poate ca într-un joc să vă nemulțumească dimensiunea mică a unui spiriduș, desenul acestuia încadrându-se într-o poziție caracter. Aveți vreo idee pentru a crea în același mod spiriduși de dimensiuni mai mari?

Sigur că da! Puteți defini părți ale spiridușilor pe câte o poziție caracter și, apoi, prin îmbinarea acestor părți, să formați un spiriduș de dimensiuni mai mari

Iată cum arată grilele pentru patru caractere (să zicem A, B, C și D) care, imbinate, 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, obținându-se prin citirea în ordine inversă a listei pentru A, iar lista valorilor pentru D obținându-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).

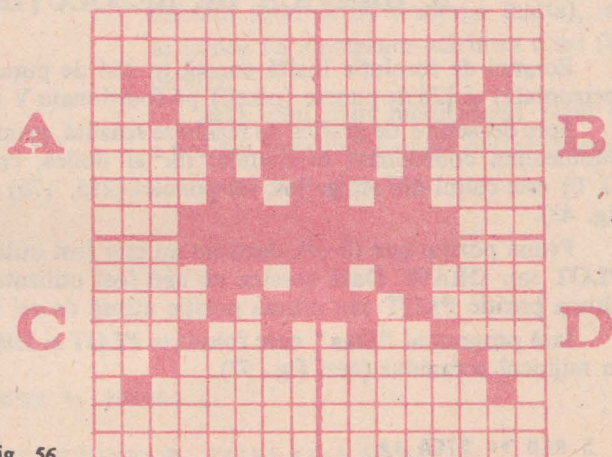


Fig. 56

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).

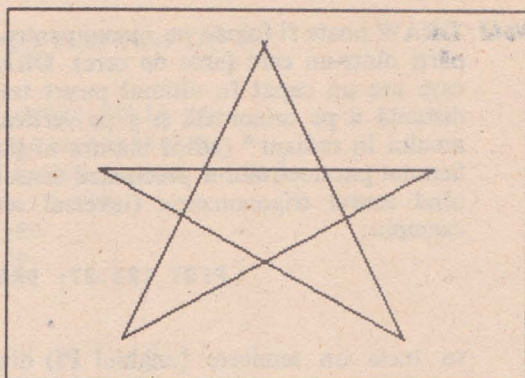
Spre deosebire de grafica de rezoluție scăzută, primul număr folosit îl reprezintă întotdeauna coordonata orizontală, iar al doilea, coordonata verticală. Punctul (0, 0) este colțul din stînga jos, iar punctul (255, 175) este cel din dreapta sus (vezi fig. 48).

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). :

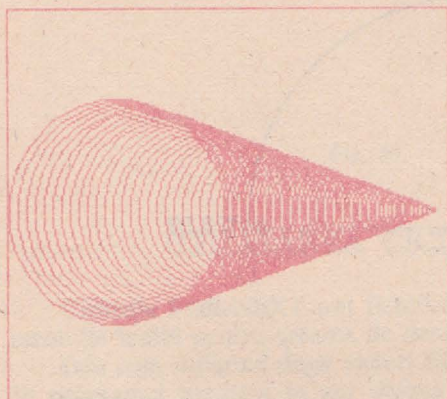
```
5 REM ** STEA **
*****
10 INK 2
20 PLOT 128,174 → Mută poziția de start în partea
de sus a ecranului.
30 DRAW 70,-140
40 DRAW -152,80
50 DRAW 164,0
60 DRAW -150,-80
70 DRAW 70,140 → 5 instrucțiuni DRAW care trasează
5 linii roșii.
```

Fig. 57



Acum adăugați următoarele linii programului:

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



Cuvântul **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 radiani). Semnul parametrului **u** precizează sensul în care se trasează arc, + însemnând 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).

PI + sens trigonometric

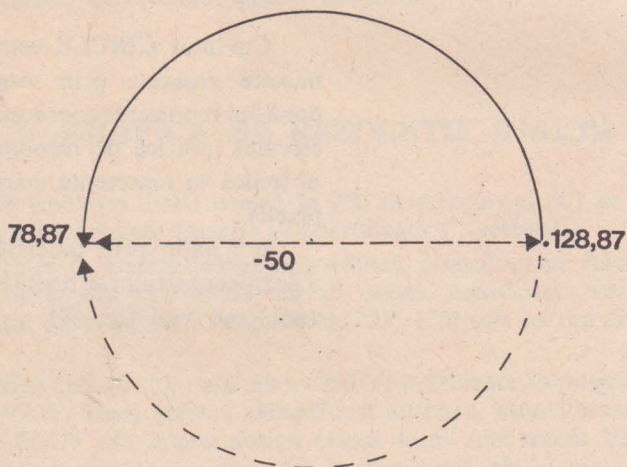


Fig. 59

PI - sens orar

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 ** TRIUNGHII 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 (utilizând STEP).

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

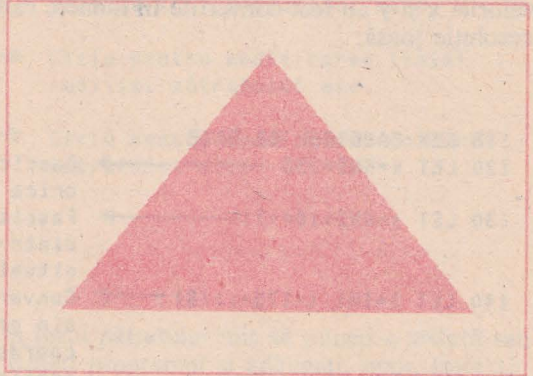


Fig. 60

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 utilizând 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
```

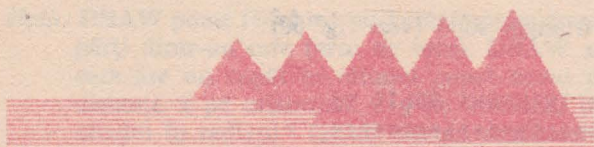


Fig. 6'

Acum adăugați și următoarele linii de program. Când veți rula din nou programul, veți vedea un fascicul de raze pe cerul nopții care produce mai multe stele. Fascicolul pornește din colțul ecranului înspre poziția (x, y), x și y fiind două variabile care vor lua valori aleatoare. Stelele vor fi desenate în diverse poziții, după ce valorile x și y au fost convertite în numere care indică poziția stelelor în grafica de rezoluție 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 rezoluție în coordonata pe verticală pentru grafica de rezoluție scăzută: când y=71, l=13 (linia 13) când y=174, l=0 (linia 0).
150 LET c=INT (x/8)	→	Conversia coordonatei pe orizontală: când x=0, c=0 (coloana 0) când 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 + SS în modul E.
190 PRINT AT l,c;"#"		
200 GO TO 120		Se vor desena liniile cu culoarea 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 restul desenului.

PROIECTAREA DESENELOR PE ECRAN

Puteți produce cu ajutorul calculatorului orice desen sau model utilizând fie grafica de rezoluție joasă, fie grafica de rezoluție înaltă, fie amândouă. Cea mai bună metodă de a proiecta desene este de a realiza mai întâi o schiță preliminară

pe o coală de hirtie, pe care în prealabil s-a trasat fin o grilă care să reprezinte atît grafica de înaltă rezoluție, cît și pe cea de rezoluție scăzută, și apoi, să se scrie programul care realizează desenul utilizînd instrucțiunile: **PRINT**, **PRINT AT**, **PLOT**, **DRAW**, **CIRCLE**, precum și 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
    și a mărimii pătratului mare.
30 INK x
40 FOR l=11-x TO 11+x → Ciclu pentru modificarea liniei
    poziției 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

```

Modificați linia 60, astfel încît în locul pătratului mic să puneți o steluță sau oricare alt caracter de pe tastatură. Rulați programul și adăugați, apoi, linia 15:

15 INVERSE 1

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

Modificați linia 15; adăugați linia 16 și rulați din nou programul:

15 BRIGHT 1	B+SS în mod E	BRIGHT realizează o afișare mai strălucitoare a caracterului.
16 FLASH 1	V+SS în mod E	FLASH realizează pulsarea poziției caracterului (caracter cli-pitor), făcînd culorile fondu-lui și caracterului să alterneze cu o cotă constantă.

Notă:

1. **FLASH**, **BRIGHT** și **INVERSE** se pot utiliza și în cadrul unor instrucțiuni **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 8.

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 întâmpla 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 drepte 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 coborârii acestuia. Pentru folosirea programului nu va fi necesară acționarea tastei CR atunci când 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 decât 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 → Poziția creionului (orizontală și
40 LET y=87 → 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$="" THEN 80 TO 60 → Dacă nu este apăsată nici o tastă
                          se așteaptă acționarea uneia.
80 IF c$="0" THEN LET a=0: → Dacă s-a acționat tasta 0, varia-
    80 TO 60                bila a devine 0 (creionul sus).
90 IF c$="1" THEN LET a=1: → Dacă s-a acționat tasta 1, varia-
    PLOT x,y: 80 TO 60      bila a devine 1 (creionul jos).
100 LET dx=0: LET dy=0 → Inițializarea creșterilor pe
                          orizontală și pe verticală.
110 IF c$="5" THEN LET dx=-1: → Dacă s-a acționat tasta 5,
    80 TO 200              coordonata pe orizontală se mic-
                          șorează cu o unitate ( ← ).
120 IF c$="6" THEN LET dy=-1: → Acționarea tastei 6 ( ↓ ).
    80 TO 200
130 IF c$="7" THEN LET dy=1: → Acționarea tastei 7 ( ↑ ).
    80 TO 200
140 IF c$="8" THEN LET dx=1: → Acționarea tastei 8 ( → ).
    80 TO 200
  
```

In locul combinației de taste C8+5 se folosește codul ASCII al caracterului respectiv.

```

150 IF c$=CHR$ 8 THEN LET dx=-1: —————> Acționarea tastelor
    LET dy=1: 80 TO 200                               5+CS (↖).
160 IF c$=CHR$ 10 THEN LET dx=-1: —————> Acționarea tastelor
    LET dy=-1: 80 TO 200                               6+CS (↙).
170 IF c$=CHR$ 11 THEN LET dx=1: —————> Acționarea tastelor
    LET dy=-1: 80 TO 200                               7+CS (↘).
180 IF c$=CHR$ 9 THEN LET dx=1 —————> Acționarea tastelor
    LET dy=1: 80 TO 200                               8+CS (↗).
190 80 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 80 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 (←)	5 + CS — deplasare stînga sus (↖)
6 — deplasare jos (↓)	6 + CS — deplasare stînga jos (↙)
7 — deplasare sus (↑)	7 + CS — deplasare dreapta jos (↘)
8 — deplasare dreapta (→)	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.

Ținând 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 decât 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 decât 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 funduri de culori diferite.

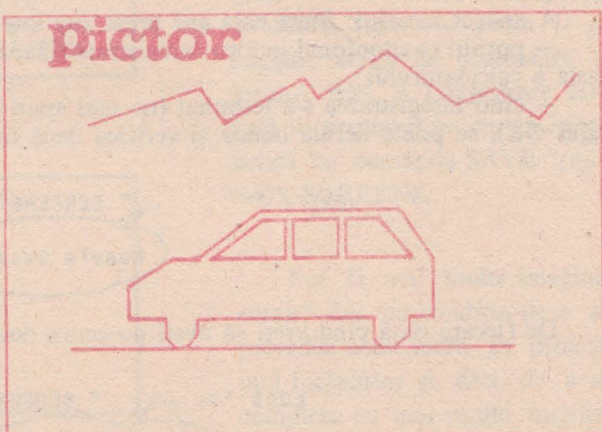


Fig. 62

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 veți proceda în felul următor:

- opriți rularea programului "Pictor". Pentru acest lucru veți acționa împreună tastele **SPACE** și **CS**, obținând **BREAK** (întreruperea programului);
- introduceți apoi următoarea comandă:

SAVE " " SCREEN\$

Aici veți introduce
numele desenului
(cel mult 9 caractere).

Se obține
acționând
K+SS în mod E.

și acționați **CR**.

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

- porniți casetofonul pentru înregistrare (clapa **RECORD**) și acționați orice tastă a calculatorului;
- când înregistrarea s-a terminat (nu mai apar dungi orizontale și apare mesajul **OK**), se poate derula banda și verifica dacă înregistrarea a fost corectă:

VERIFY "

" SCREEN\$

Numele desenului.

De fiecare dată când vreți să aveți pe ecran desenul dorit, veți da comanda:

LOAD "

" SCREEN\$

Numele desenului.

veți potrivi caseta pe care a fost salvat desenul respectiv și veți da drumul la case-
tofon (pe **PLAY**).

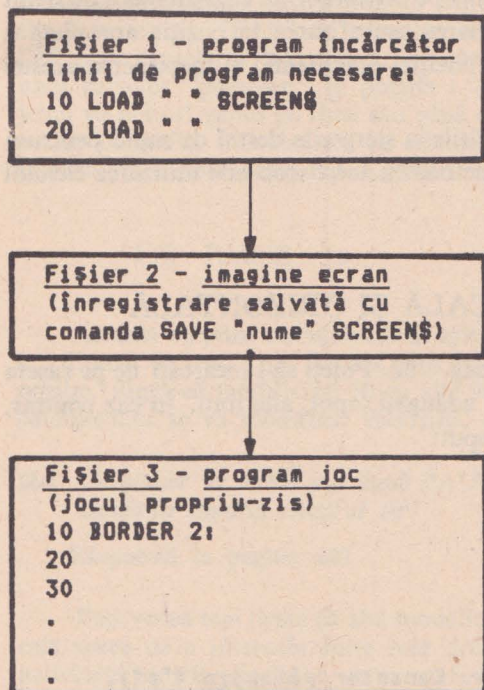
Desenul se conturează pe ecran, parte cu parte, pe măsură ce banda se derulează,
adică pe măsură ce informația este citită.

STRUCTURA UNUI JOC CU ECRAN GRAFIC DE PREZENTARE

Sintem convinși că v-ați gândit imediat cum puteți să integrați într-un joc o ima-
gine de acest fel, de exemplu: o imagine în care să prezentați grafic (cu desene și
text) jocul respectiv, după care să înceapă jocul propriu-zis. Acest lucru se poate

realiza asigurându-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.

Structura unui joc



- 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\$.

Fig. 63

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 instruc-

țiuni 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 câtorva imagini.

— următorul fișier este chiar **programul de joc**. În timpul în care are loc încărcarea acestui program, imaginea (de prezentare a jocului) rămâne pe ecran, fiind deci la dispoziția jucătorului, astfel încât 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.

ANIMAȚIA

Vă plac desenele animate?

Un joc pe calculator va fi cu atât mai plăcut, cu cât desenele realizate vor prinde viață pe ecran, prin mișcare imitând realitatea. Caracterele sau liniile se pot mișca pe ecran și astfel producerea animației pe calculator nu este un lucru deosebit de dificil. Tot ce trebuie făcut este desenarea sau afișarea în poziția apropiată a liniei sau caracterului (neuitând ștergerea poziției precedente) și menținerea acestei "mișcări" pe o traiectorie.

Calculatorul va realiza desenele, afișările și ștergerile destul de rapid pentru a da senzația unei mișcări. Cea mai bună metodă în acest scop este utilizarea ciclului FOR-NEXT.

MIȘCAREA PE VERTICALĂ ȘI ORIZONTALĂ

Reluați programul "Păianjen" de la pag. 92. Puteți să-l încărcați de pe caseta magnetică dacă l-ați salvat anterior și să adăugați, apoi, alte linii. În caz contrar, tastați 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 60,126,219,220,189,165,
   1,36
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 l=0 TO 20
120 PRINT AT 1,3; INK 0;"|"

130 PRINT AT 1+1,3; INK 2;"P" → Tasta P în mod 8.
```

Tasta F în mod 8.
Caracterul va fi afișat așa după ce ați rulat o dată programul.

Animatia: cu un ciclu FOR-NEXT caracterul "păianjen" se înlocuiește cu caracterul "fir", iar pe poziția actuală se înscrie caracterul "păianjen".

Prin intermediul instrucțiunilor dintre liniile 110 și 140, de fiecare dată cind ciclul se repetă, "firul" de păianjen se va lungi, adăugîndu-se un nou caracter "l" 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 păianjenului 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 "Animatie" 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; "l" → 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 întii 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 ștergă 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 I=0 TO 20	
130 PRINT AT I,c:" "	→ Ștergerea păianjenului de pe vechea poziție.
140 LET c=c+1	
150 PRINT AT I+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 I	

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) —————> Se stabilește coloana pe care
                                     se va deplasa gândacul.
200 FOR I=0 TO 19
210 PRINT AT I,c;" * " —————> Șterge partea de sus a gândacului.

220 PRINT INK 6;AT I+1,c;" * " —————> Mută partea de sus a gândacului
                                     pe linia de mai jos (de aceea
                                     nici nu mai este necesară șter-
                                     gerea părții de jos a gândacu-
                                     lui, întrucît în locul acesteia
                                     se afișează partea de sus).

```

Se obține cu A și B în mod 6.

Se obține cu C și D în mod 6

```

230 PRINT INK 6;AT I+2,c;" * "
240 NEXT I
250 GO TO 190

```

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 rămină 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.

$$\text{LET } X=X+P*(\text{INKEY}\$="D")-(\text{INKEY}\$="S")$$

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 mișcarea 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 mișcării 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.

Animatia se poate realiza nu numai cu caractere grafice, ci și cu oricare alte caractere (deci și litere). La ce poate folosi acest lucru în jocuri? Bineînțeles, la prezentarea mai frumoasă a titlului unui joc, a textului (instrucțiunile de utilizare) sau a diverselor mesaje care se afișează în timpul jocului. Iată un exemplu de subrutină care se poate apela din programul principal, în scopul afișării unui text (în cazul de față titlul jocului și tastele de folosit), astfel încât litera care urmează să fie afișată să se plimbe pe ecran și, 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 afișat.
   5 - STINGA, 6 - SUS, 7 - JOS,
   8 - DREAPTA, CS - DIAGONALE"
16 GO SUB 1000 → Apelarea subrutinei de afișare.
17 REM PROGRAM PRINCIPAL
999 STOP
1000 FOR x=1 TO LEN a$
1010 PRINT AT 0,0;a$( TO x) → Afișarea cîte unei litere.
1020 NEXT x
1030 RETURN
```

Încercați să introduceți această subrutină în jocul „Pictor” de la pagina 105

FOLOSIREA ȚINTEI

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

Detectarea unei coliziuni nu este dificilă. Dacă două caractere sînt afișate în poziția *l*, *c* (pentru linie și coloană) și respectiv *v*, *h* (pentru vertical și orizontal) și dacă $l = v$ și $c = h$, atunci caracterele vor fi în aceeași poziție. 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 decât 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;"▲"
```

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

- Apare fasciculus
- de raze în timp ce
- păianjenul se
- deplasează în jos.

```
200 NEXT v
```

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

Cînd ajunge pe ultima linie, păianjenul devine clipitor și își schimbă culoarea. Își începe mișcarea alt păianjen.

```
210 GO TO 114
```

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
```

Se obține cu L+85 în mod E

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

ATTR depistează "atributele" unei anumite poziții (linie, coloană) de pe ecran, adică culoarea (INK), fondul (PAPER), luminozitatea (BRIGHT) și clipirea (FLASH) poziției respective.

În 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.

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

Explozie.

Tasta E în mod 8.

510 PAUSE 100 → Explozia se menține circa 2
secunde, după care își începe
mișcarea alt păianjen.

520 60 TO 114 → Alt păianjen.

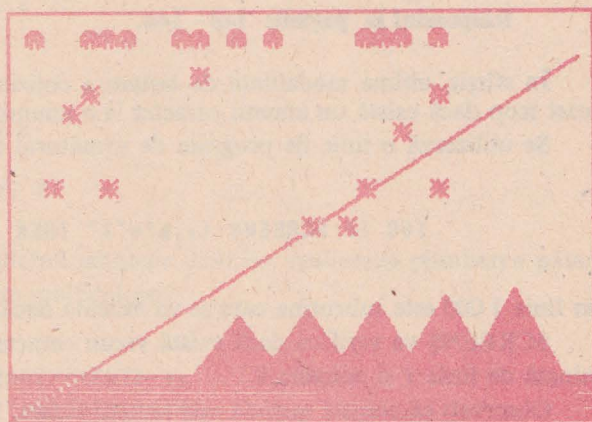


Fig. 64

Cînd fasciculul luminos lovește păianjenii, aceștia devin galbeni. Cînd linia realizată cu DRAW atinge poziția caracterului "păianjeni", acesta va "lua" aceeași culoare cu linia (galben) (vezi fig. 64). În linia 190, se testează care sînt atributele caracterului de pe poziția determinată de linia $v + 1$ și coloana h .

Dacă acestea sînt:	Cod	Valoarea cu care se înmulțește	Valoarea obținută
— 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 înmulțite cu valorile 1, 8, 64 și, respectiv, 128) și se va executa linia 500 care va simula explozia păianjenului.

Pentru orice combinație 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 "spiridușii" pe ecran afișindu-i pe noua poziție și ștergîndu-i de pe poziția precedentă punînd în locul lor un spațiu gol. Dar dacă "spiridușul" 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 "spiridușilor" 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.

Noță. Dacă veți testa acum din nou PRINT AT 10,10; OVER 1; "*" steluța va dispărea, pătrățelul negru devenînd intact. Deci, pentru a realiza o deplasare "transparentă", în loc să ștergeți "spiridușul", 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, utilizând tehnicile descrise, realizează plimbarea păianjenului într-un decor de pătrate multicolore:

```

5 REM ** PLIMBARE TRANSPARENȚA **
  *****
10 BORDER 0: PAPER 0: INK 4
20 CLS
30 LET X=0: LET Y=0: GO SUB 1000
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$="5")
80 LET Y=Y+(a$="6")-(a$="7")
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

```

→ Se stabilește locul de pornire a păianjenului.

→ Tastele cu care se va mișca păianjenul: 8 la dreapta, 5 la stânga, 6 în jos și 7 în sus.

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

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

Calculatorul posedă un sintetizator de sunete cu ajutorul căruia jocurile pot deveni mult mai antrenante, prin intercalarea în desfășurarea lor a diverselor efecte sonore și secvențe muzicale. Vă puteți imagina la ce se pot folosi efectele sonore în jocuri? La simularea sonoră a funcționării unor mecanisme (tren, avion, elicopter, diverse tipuri de mașini), decolarea navelor spațiale, coliziunea unor obiecte (păcănituri, explozii), vîjiitul proiectilelor, zgomotul de pași, etc. Secvențele muzicale se pot folosi la prezentarea jocurilor și în perioadele în care este indicat să se facă scurte pauze.

În jocurile pe bază de întrebare și răspuns este bine ca un răspuns corect să fie însoțit de o scurtă (dar plăcută) secvență muzicală (care poate fi oferită și drept recompensă), în timp ce, un răspuns incorect să fie însoțit de o scurtă secvență care să atragă atenția asupra greșelii.

Deși calculatorul nu poate realiza într-un anumit moment decît un singur lucru, faptul că posedă o viteză deosebit de mare va da senzația producerii efectelor sonore și a secvențelor muzicale în același timp cu secvențele grafice care se desfășoară pe ecran.

Producerea muzicii cu ajutorul calculatorului nu este complicată necesitînd numai cunoștințe 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 acționarea tastelor **Z** și **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. Ordonind 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 atât pe claviatura unui pian, cât 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 aparținând 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.

	1	3	6	8	10	13
notație calculator	do' 0	re' 0	fa' 0	sol' 0	si' b	
notație uzuală	C0	D0	F0	G0	A0	
notație literară	C0	D0	F0	G0	A0	

pt. clapele negre

pt. clapele albe

	C	D	E	F	G	A	B	C	D
notație literală	C	D	E	F	G	A	B	C	D
notație uzuală	do'	re'	mi'	fa'	sol'	la'	si'		
notație calculator	0	2	4	5	7	9	11	12	14

Correspondența pe portativi

A musical staff with a treble clef. Six notes are placed on the staff: do¹ (first line), do¹# (first space), re¹ (second line), re¹# (second space), mi¹ (third line), and fa¹ (third space). Below the staff, a table shows the correspondence between these notes and their notations.

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, obținindu-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

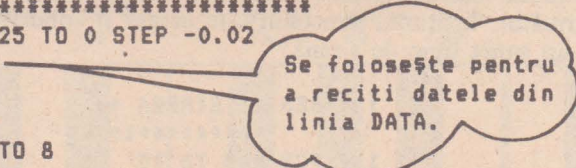
Deoarece poate produce o mare varietate de sunete, calculatorul poate fi folosit la interpretarea unor partituri muzicale. În programarea secvențelor muzicale, se poate urmări o linie melodică, precum și un anumit ritm. Din păcate, însă, cu calculatorul avuț la dispoziție nu se pot produce decît secvențe 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 și o scurtă melodie realizată pe calculator: se observă preferința de a grupa toate înălțimile 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 acționarea 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 și ceva). Putem obține notele respective prin acționarea unor taste ale calculatorului. În general, tastele de pe rîndul de sus (tastele numerice 1, 2, 4, 5, 6, 8 și 9) reprezintă clapele negre ale pianului, iar cele de pe rîndul al 2-lea reprezintă pe cele albe. Pentru a obține gama DO, de exemplu, se vor acționa, succesiv, tastele W, E, R, T, Y, U, I și O.

Durata fiecărei note este aceeași. Dacă doriți ca melodia cîntată să fie mai vioaie (sau mai lentă) puteți 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".

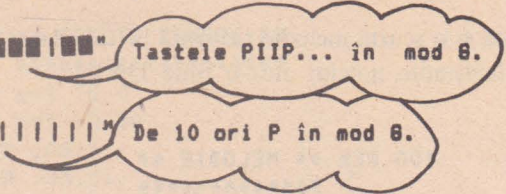
50 NEXT n
60 LET p$=""
70 LET n$=""
80 LET p=0
90 LET x=0
100 PAPER 0: INK 7: BORDER 0: CLS
110 PRINT

```

```

120 FOR m=1 TO 5
130 PRINT TAB 10;"■■■■■■■■■■" Tastele PIIP... în mod 8.
140 NEXT m
150 FOR m=1 TO 4
160 PRINT TAB 10;"||||||||||" De 10 ori P în mod 8.
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

```



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

```

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;"||||||||||"

```

19 spații libere.

```

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

```

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

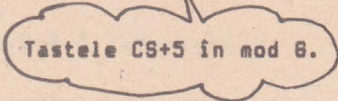
Tasta 5 în mod 8.

```

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

```

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



Acum puteți cânta 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 același 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 și editare de texte). Vor trebui desenate pe ecran portativul și notele de pe portativ pentru o anumită secvență melodică. De asemenea, va trebui să existe posibilitatea ca o notă să fie ștearsă, punându-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 și sunetele care sînt emise de calculator. În acest scop valorile pentru duratele și înălțimile sunetelor vor trebui memorate în variabile.

Toate aceste probleme le puteți rezolva singuri pe baza cunoștințelor acumulate pînă în prezent.

28. *Încercați să realizați un program care să întrunească condițiile descrise mai sus, adică să poată memora melodii și să le poată reda, atît sonor, cît și 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 pîriu.

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 pîriului); 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 (tehnic) folosite de programator pentru realizarea jocului:

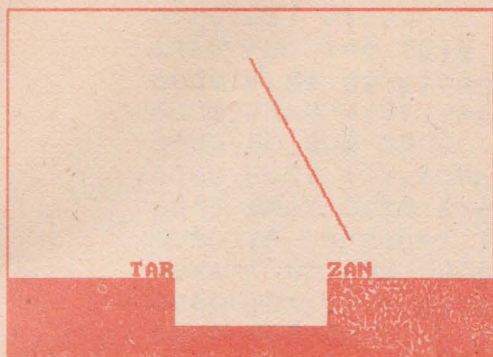
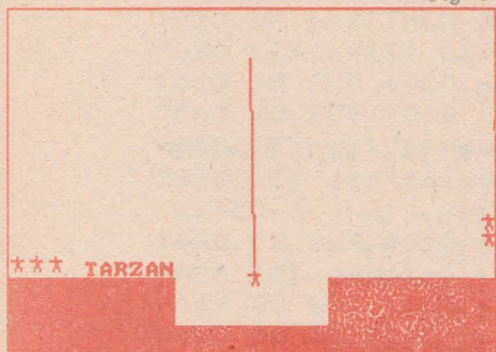


Fig. 66

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

Fig. 67



Iată jocul realizat (de data aceasta programul se prezintă așa 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 i;"*";: 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;"x"
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

```

*Tasta A
in mod G.*

```

9050 LET x(i)=-105*SIN j
9060 LET y(i)=-105*COS j
9070 NEXT j
9080 LET a$="██████████"
9090 LET b$="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 ȘI VARIABLELE IMPORTANTE ALE PROGRAMULUI

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

● **Subrutina de "inițializare" a unui explorator** prin care se pune în mișcare un nou explorator de fiecare dată când predecesorul a ajuns pe malul celălalt șan 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 mișcare (linia 1 020), oprirea jocului când nu mai sînt exploratori (linia 1 050) — singura situație de ieșire către programul principal, apelarea subrutinei pentru afișarea unui explorator (linia 1 060).

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

● **Subrutina de afișare a unui explorator** (liniile 3000—3090): permite afișarea caracterului explorator în orice loc pe ecran și 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 mișcare (linia 4000), afișarea exploratorului (liniile 4010 și 4020), deplasarea și afișarea exploratorului (liniile 4030 și 4040), test care verifică dacă exploratorul a reușit să se agațe de liană (linia 4050: o distanță mică față de extremitatea lianei și același 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 acționează o tastă sau un explorator este pe punctul de a se deplasa (linia 5020).

● **Subrutina de deplasare a lianei** (liniile 6000—6070): inițializarea variabilei *steag* pentru a arăta dacă liana poate fi apucată de explorator (linia 6000), parametrizarea unui ciclu pentru utilizarea aceleiași subrutine, oricare ar fi sensul de deplasare a lianei (linia 6010), afișarea lianei (linia 6020), apelarea subrutinei de dispecerat (linia 6030), ștergerea lianei (linia 6040), cazul în care un explorator este în mișcare (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 (liniile 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 curvintului **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 exercițiile și 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 ceașcă de ceai **

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

1.1.1. Uplete ceainicul cu apă.

1.1.2. Pune ceainicul pe plită.

1.1.3. Așteaptă pînă cînd fierbe apa.

1.1.4. Ia ceainicul de pe plită.

Comentariu * Sfirșitul sarcinii 1.1.*

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

1.2.1. Ia o ceașcă de ceai.

1.2.2. Pune un pîchet de ceai în ceașcă.

Comentariu * Sfirșitul sarcinii 1.2. *

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

1.3.1. Adu ceașca la ceainic.

1.3.2. Pune apă în ceașcă.

1.3.3. Amestecă ceaiul cu lingurița.

Comentariu * Sfirșitul sarcinii 1.3. *

Comentariu ** Sfirșitul algoritmului — ceașca 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. Răsuțește robinetul spre dreapta.

1.1.1.3. Așteaptă pînă cînd ceainicul e plin.

1.1.1.4. Răsuțește 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.

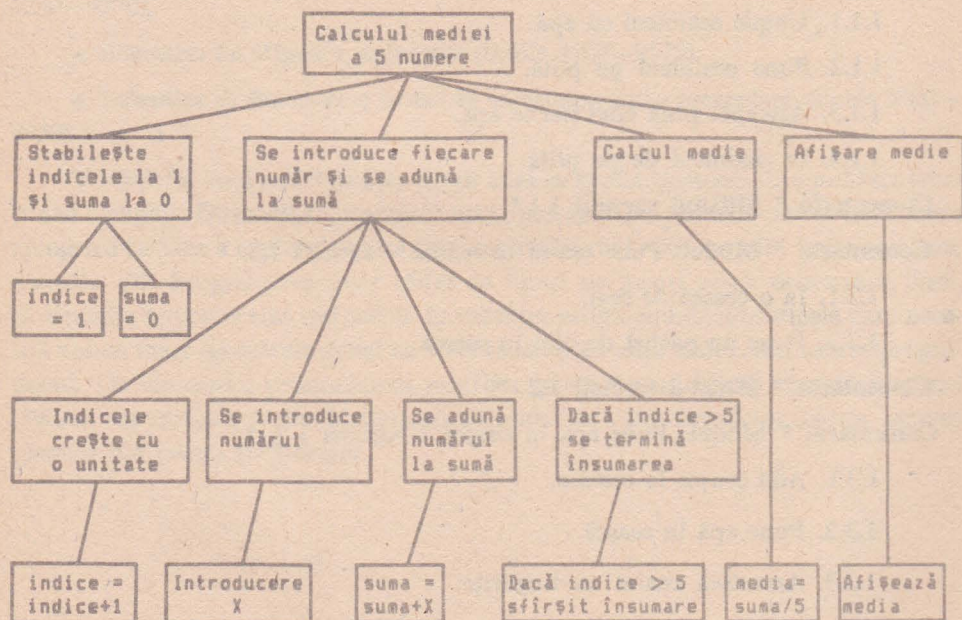
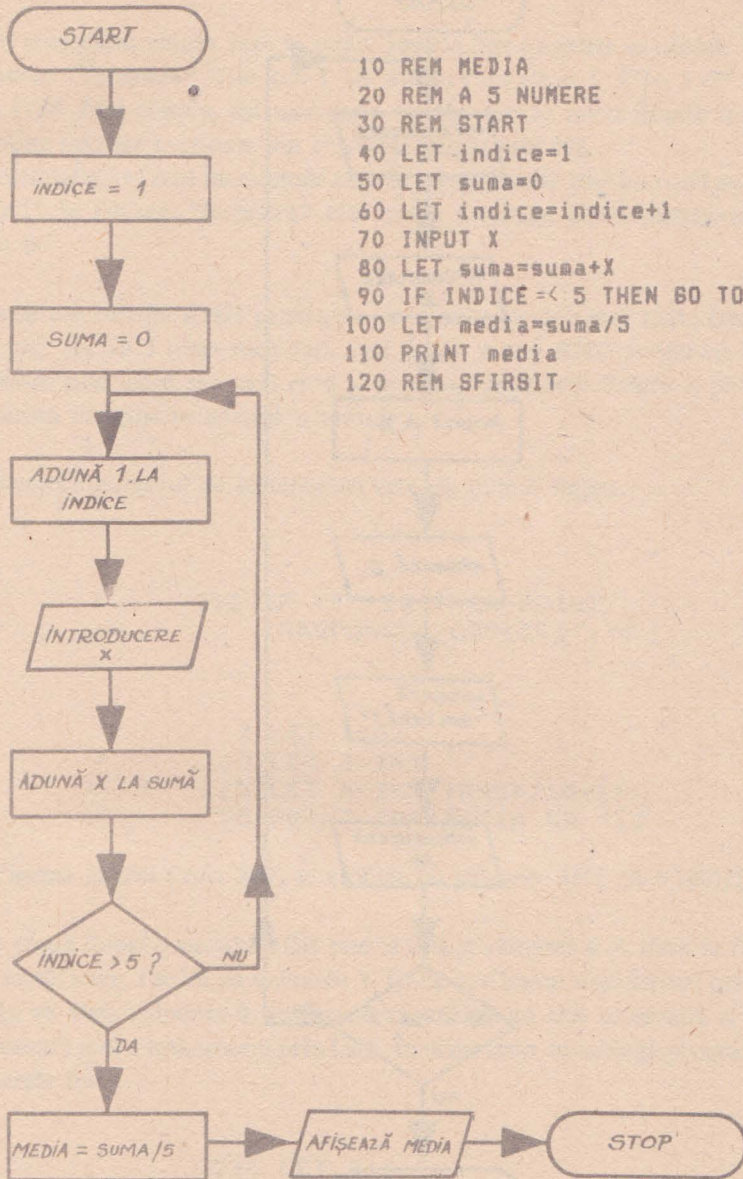


Fig. 68 Diagrama de tip arbore pentru calculul mediei a 5 numere.

b) și c) vezi fig. 69.

Schema logică



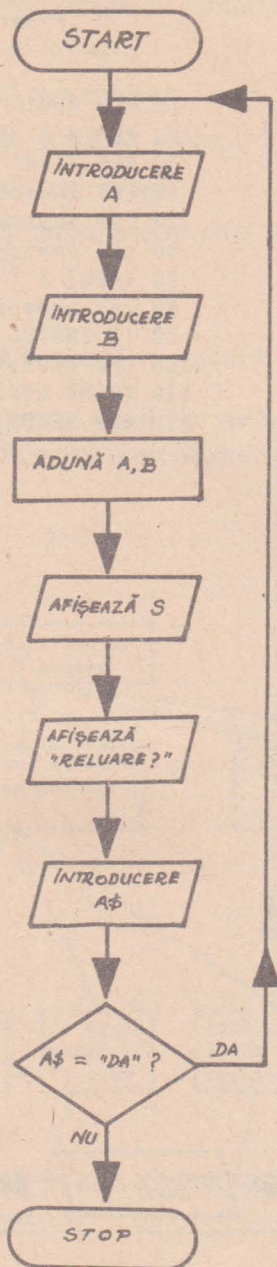
Programul

```
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) și b) Vezi fig. 70.

Schema logică



$$6. a) (101101)_2 = 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 32 + 8 + 4 + 1 = 45$$

$$b) (3A7)H = 3 \cdot 16^2 + 10 \cdot 16^1 + 7 \cdot 16^0 = 768 + 160 + 7 = 935$$

7. Cel mai mare număr care se poate înscrie într-un octet se obține "punind" în fiecare "fereastră" câte un 1 și, astfel, vom avea: $1 \cdot 2^7 + 1 \cdot 2^6 + 1 \cdot 2^5 + \dots + 1 \cdot 2^0$. Mai simplu, cel mai mare număr care se poate înscrie într-un octet se obține scăzând o unitate din 2^8 . Acest număr este **255**.

Bineînțeles, cel mai mic număr care se poate înscrie într-un octet se obține "punind" în fiecare "fereastră" câte un 0, rezultând 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 $RND \times 100$ va fi un număr aleator mai mare sau egal cu 0 și mai mic decât 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 sfârșitul 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 ȘTEFAN STOP AFLĂ DACĂ ARE UN JOC DE ȘAH ȘI UN CALCULATOR STOP.

16. Textul era scris deandoselea. Pentru decodificare, se vor modifica in program liniile 110 și 220 astfel:

```
FOR J=L TO 1 STEP -1
```

17. Codurile ASCII pentru literele mari se găsesse între 65 și 90. Trebuie modificată linia 90.

18. Recunoașterea 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 DARBA
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 e; 4-DELETE: se șterge litera t; 5-DELETE: se șterge cuvîntul și.

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.

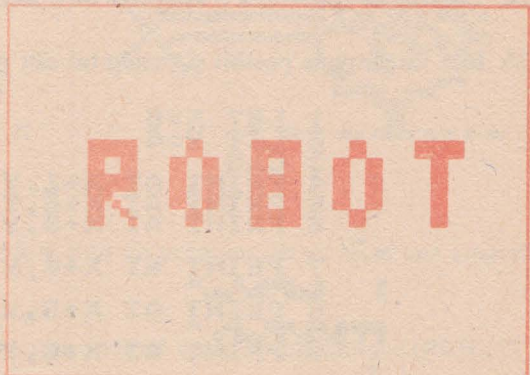


Fig. 71

22. Iată unele seturi posibile (ale voastre pot arăta și 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
ț      ă      i      n      t      r      -
O SPACE M A CS+9 S CS+9 I N U
o      m      a      ș      i      n      u
CS+9 T A SPACE M CS+9 SS+P CR
      ț      ă      :ș      "
  
```

23. Aveți toată libertatea de a realiza practic această temă.

24. PAUSE 50 provoacă o așteptare de o secundă din partea calculatorului.

25. Iată un răspuns posibil și rezultatul său pe ecran (fig. 72):

```

1 LET X=6
2 LET Y=4
3 CLS
4 PRINT AT X+1, Y+13; " "
5 PRINT AT X+2, Y+11; " "
6 PRINT AT X+3, Y+10; " "
7 PRINT AT X+4, Y+1; " "
8 PRINT AT X+5, Y+1; " "
9 PRINT AT X+6, Y+10; " "
  
```



```

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 27
22 IF A$="D" THEN GO TO 28
23 IF A$="S" THEN GO TO 29
24 IF A$="B" THEN GO TO 30
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=8 THEN GO TO 3
36 LET Y=Y-1
37 GO TO 3
38> PRINT AT 20,12;"SFIRSIT"

```

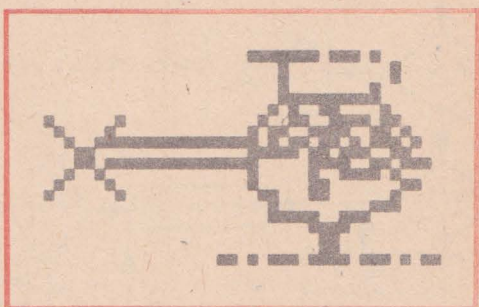


Fig. 72

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 roșie pe un fond verde și este luminos (**BRIGHT**);

- b) Caracterul are culoarea neagră pe un fond bleu și este clipitor (**FLASH**);
- c) Caracterul are culoarea albastră pe un fond alb, este luminos (**BRIGHT**) și clipitor (**FLASH**).

B. Metoda de determinare a atributelor când se cunoaște valoarea lui **ATTR** este următoarea:

- se testează dacă valoarea lui **ATTR** este mai mare decât numărul **128**; dacă da, atunci caracterul este clipitor (**FLASH = 1**), iar valoarea care se va testa în continuare (noul **ATTR**) va fi diferența care rămâne prin scăderea lui **128**; dacă nu, atunci:
- se testează dacă valoarea lui **ATTR** este mai mare decât numărul **64**; dacă da, atunci caracterul este luminos (**BRIGHT = 1**), iar valoarea care se va testa în continuare (noul **ATTR**) va fi diferența care rămâne prin scăderea lui **64**; dacă nu, atunci:
- codul culorii fondului (**PAPER**) va fi egal cu partea întreagă a împărțirii 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 puteți urmări în fig. 73.

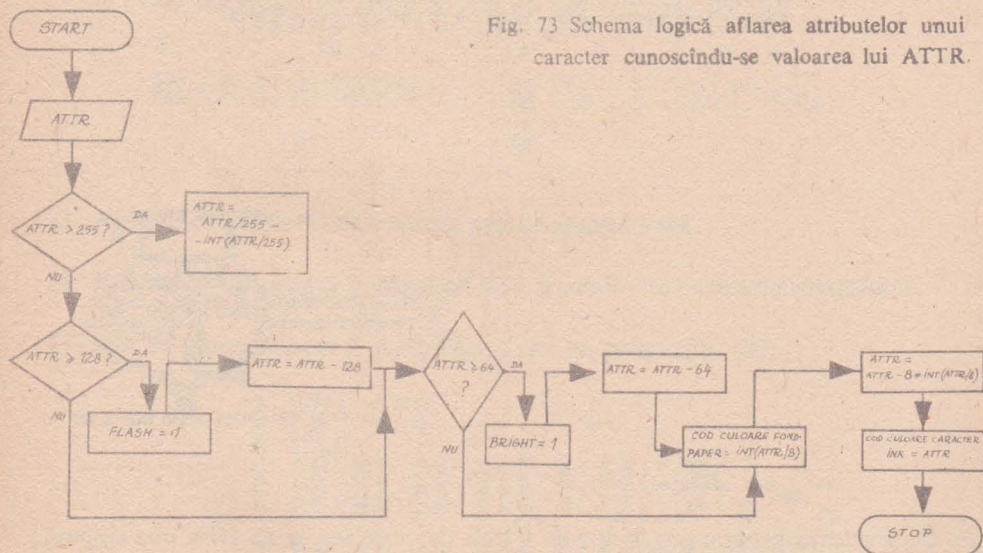


Fig. 73 Schema logică aflarea atributelor unui caracter cunoscându-se valoarea lui **ATTR**.

28. Prezentăm programul "Muzicianul", care rezolvă problemele expuse (programul vostru poate arăta altfel):

"Muzicianul" folosește tastatura, astfel încât se pot compune melodii (maximum 5 a câte 32 de note, pe două octave). Pentru obținerea 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 obținerea notelor pe ecran s-a proiectat un caracter grafic corespunzător tastei O. Toate notele vor avea aceeași durată, dar aceasta se poate modifica la o nouă audiere a melodiei. Se pot înscrie pauze (care au durată egală cu a notei) cu ajutorul tastei BREAK.

Ultima notă se poate șterge și, apoi, înlocui cu alta prin intermediul tastei O

```

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$="" THEN GO TO 120
140 IF t$="O" 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 O in mod G

Tasta O in mod G

Acum puteți compune melodii pe care calculatorul le va memora și reda la cerere.

ANEXĂ

INDEXUL CUVINTELOR CHEIE

Prin cuvinte **cheie** se înțeleg cuvintele introduse în memoria calculatorului și care prezintă o anumită semnificație pentru acesta. La calculatoarele la care se referă lucrarea de față, orice cuvânt cheie (pentru limbajul BASIC) se introduce prin acționarea unei anumite taste sau a unei combinații de taste. Orice cuvânt cheie poate face parte din următoarele categorii:

- **comandă** - cuvânt cheie a cărei introducere are ca urmare execuția imediată de către calculator a unei acțiuni. *Exemple:* RUN, LIST, LOAD etc.

- **instrucțiune** - cuvânt cheie care are ca urmare o acțiune executată de calculator și care poate fi utilizat într-o linie de program. Acțiunea este îndeplinită numai atunci când programul va fi executat. *Exemple:* INPUT, READ, PRINT etc.

Notă. Un cuvânt cheie care reprezintă o comandă va funcționa ca o instrucțiune atunci când este folosit într-o linie de program. *De exemplu:* 1000 LIST, cuvântul cheie LIST reprezintă o instrucțiune, acțiunea de listare a programului realizându-se când execuția programului va ajunge la linia 1000.

Invers, o instrucțiune poate fi considerată comandă când va fi folosită ca atare. *De exemplu:* PRINT 3+4, cuvântul cheie PRINT reprezintă o comandă deoarece acțiunea de afișare a rezultatelor se realizează imediat după introducere.

În indexul cuvintelor cheie se va indica cu prioritate categoria cea mai reprezentativă în care se poate încadra cuvântul cheie respectiv. De exemplu, RUN se va trece în categoria comandă sau instrucțiune, deoarece deși se poate utiliza și ca instrucțiune, se utilizează în primul rând ca și comandă. Invers, INPUT se va trece la categoria instrucțiune/comandă, deoarece deși se poate folosi și în mod comandă, se utilizează în primul rând în linii de program, deci ca instrucțiune.

- **funcție** - cuvânt cheie a cărui folosire produce o valoare. Acest cuvânt se utilizează în combinație cu o comandă sau o instrucțiune.

Exemple: INT, RND, ABS, SIN, SQR.

Exemplu de utilizare: PRINT INT 8.1

- **operator logic** - cuvânt cheie care este utilizat într-o comandă sau instrucțiune și care determină dacă o condiție este adevărată sau falsă.

La calculatoarele la care se referă lucrarea de față se utilizează doar trei operatori logici, și anume, **AND**, **OR** și **NOT**.

- **alte cuvinte cheie** - cuvinte cheie care sînt utilizate în cadrul unor comenzi sau instrucțiuni și care reprezintă particule de propoziții făcînd o legătură între cuvinte.

Exemple: THEN, AT, STEP etc.

În scopul cunoașterii și folosirii cuvintelor cheie, adică a realizării de programe, vor trebui cunoscute:

- localizarea cuvintelor cheie pe tastatură (pe ce tastă se găsește și în ce mod se obține);

- ce fel de cuvînt cheie este (categoria);

- scopul utilizării cuvîntului cheie (în ce caz se folosește);

- formatul cuvîntului cheie (sintaxa).

Deoarece în lucrare sînt oferite numeroase exemple de folosire a cuvintelor cheie în programe, deseori fiind indicată și modalitatea de obținere s-a considerat util să se facă doar o trecere în revistă (memento) în ordine alfabetică a tuturor cuvintelor cheie folosite, indicîndu-se doar semnificația (din care rezultă și scopul principal de folosire), categoria în care se încadrează cuvîntul precum și paginile din lucrare în care puteți regăsi cuvîntul cheie respectiv într-un anumit program. Indexul nu conține toate cuvintele cheie din limbajul BASIC, ci doar marea majoritate. Restul de cuvinte le puteți găsi pe tastatură, iar pentru utilizarea lor puteți consulta un manual de referință.

ABS (valoare ABSsolută). Funcție. Produce valoarea absolută a unei valori numerice, adică valoarea fără semn pozitiv sau negativ. *Pag:* 37, 130.

AND (ȘI). Operator logic. Leagă două sau mai multe condiții în cadrul unei instrucțiuni. Dacă toate condițiile legate prin AND sînt adevărate, atunci combinația va fi adevărată. *Pag:* 37, 48, 115, 120.

AT (afișează LA sau scrie LA). Particulă de legătură folosită în cadrul instrucțiunilor PRINT, LPRINT sau INPUT pentru a localiza o afișare 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 (ATTRibute). Funcție. Depistează atributele (culoarea cu care se afișează culoarea fondului, luminozitatea, intermitența afișării) unui caracter a cărei poziție pe ecran este specificată. *Pag:* 116, 117.

BEEP (sunet). Instrucțiune/comandă. Produce o notă muzicală de o anumită durată și înălțime. *Pag:* 102, 106, 122, 123, 124, 125, 126.

BIN (număr BINar). Funcție. Transformă un număr binar într-unul zecimal. *Pag:* 90, 91, 98.

BORDER (BORDURA, marginea, chenarul ecranului). Instrucțiune/comandă. Determină culoarea culoaarea marginii ecranului. *Pag:* 80-83, 87,88,91,95,101,103, 105, 110, 119,131.

BRIGHT (luminozitatea). Instrucțiune/comandă. Determină afișarea caracterelor într-o culoare mai luminoasă decât normal. *Pag:* 103, 117.

CHR\$ (șir de CHaRactere) Funcție. Produce orice caracter sau cuvânt cheie care se găsește pe tastatură. *Pag:* 59,61, 106.

CIRCLE (CERC). Instrucțiune/comandă. Trasează un cerc pe ecran. *Pag:* 99.

CLS (CLear Screen, șterge ecranul). Instrucțiune/comandă. Șterge de pe ecran orice text sau grafică, lăsând întregul ecran cu culoarea fondului. *Pag:* 77, 99, 101, 104, 110, 119.

CODE (CODul). Funcție. Produce codul numeric (ASCII) al unui caracter sau cuvânt cheie care se găsește pe tastatura calculatorului. *Pag:* 59-62.

COS (COSinus). Funcție. Produce cosinusul unui unghi. *Pag:* 132.

DATA (DATE). Instrucțiune. Specifică o listă de date. Se folosește în program în conjuncție cu instrucțiunea READ. *Pag:* 44, 45, 47, 50, 69, 83, 84, 92, 95, 110, 115, 119, 124, 125, 131.

DIM(DIMensiune). Instrucțiune. Rezervă un spațiu de memorie. *Pag:* 40, 41, 44, 50, 51, 55, 57, 61, 62, 64, 67, 69, 131.

DRAW (desenează linie). Instrucțiune/comandă. Desenează pe ecran linii drepte sau curbe. *Pag:* 98, 100, 101, 130, 132.

FLASH (afișare intermitentă, clipitoare). Instrucțiune/comandă. Afișează intermitent o poziție caracter (culoarea caracterului alternează cu culoarea fondului cu o frecvență constantă). *Pag:* 103, 116, 117.

FOR (repetă). Instrucțiune/comandă. Se începe o structură ciclică (buclă). Se utilizează în conjuncție cu NEXT care încheie bucla. Secțiunea dintre FOR și 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). Instrucțiune/comandă. Salt necondiționat de linie indicat. *Pag:* 37, 38, 42, 43, 44, 45...

IF (dacă). Instrucțiune/comandă. Testarea unei condiții. *Pag:* 36, 37, 38, 42, 43,44, 46, 48, 49 etc.

INK (cerneala, culoarea caracterului). Instrucțiune/comandă. Specifică culoarea cu care se afișează caracterele, punctele și 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 șir de caractere). Funcție. Detectează ce tastă a fost acționată (citește tastatura). *Pag:* 105, 106, 113, 119, 126,130.

INPUT (INTRODUCE). Instrucțiune/comandă. Se introduc datele de la tastatură în timpul execuției programului. *Pag:* 36, 38, 40, 42, 43, 49, 52, 55, 57, 58, 60...etc.

INT (INTreg). Funcție. Produce numere întregi. *Pag:* 36, 49, 57, 69, 113, 119.

INVERSE (INVERS). Instrucțiune/comandă. Inversează culoarea fondului cu cea a caracterului în cadrul unei poziții caracter. *Pag:* 103, 106.

LEN (LENGht of string, lungimea unui șir de caractere). Funcție. Produce lungimea unui șir de caractere. *Pag:* 61, 66, 68, 114.

LET (fie, pune ceva într-o locație de memorie). Instrucțiune/comandă. Se atribuie o valoare unei variabile (se pune o valoare într-o locație 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/instrucțiune. Afișează lista instrucțiunilor programului din memoria calculatorului. *Pag:* 40.

LOAD (INCARCA). Comandă/instrucțiune. Încarcă un program de pe un suport extern în memoria calculatorului. *Pag:* 108, 109.

LPRINT (scrie la imprimantă). Instrucțiune/comandă. Afișează date pe hîrtia de la imprimantă. *Pag:* 72,73.

OR (SAU). Operator logic. Leagă două sau mai multe condiții în cadrul unei instrucțiuni. Dacă cel puțin o condiție este adevărată atunci combinația va fi adevărată. *Pag:* 110, 126, 129, 130.

OVER (peste). Instrucțiune/comandă. Afișează un caracter pe deasupra altuia prin suprapunere. *Pag:* 118, 119, 126, 129, 130, 131, 132.

PAPER (hîrtia, culoarea fondului). Instrucțiune/comandă. Selectează culoarea fondului pentru întregul ecran grafic sau pentru caracter, precum și pentru punctele și liniile desenate în cadrul unei anumite poziții caracter. *Pag:* 80, 81, 82, 83, 84, 87-89, 91, 95,99, 101, 103, 104, 105, 110, 116, 117, 119, 131.

PAUSE (PAUZĂ). Instrucțiune/comandă. Suspendă execuția unui program pe o perioadă de timp definită sau nedefinită. *Pag:* 111, 117, 122, 123, 124.

PI (π ; 3,14). Funcție. Introduce valoarea lui PI (π). *Pag:* 100, 131.

PLOT (punct). Instrucțiune/comandă. Desenează un punct într-o anumită poziție pe ecran. *Pag:* 98, 100, 102, 104, 105, 106, 129.

POKE. Instrucțiune/comandă. Modifică valoarea unui octet dintr-o locație de memorie. *Pag:* 90, 92, 95, 98, 110, 115, 119, 125, 131.

PRINT (AFISEAZĂ). Instrucțiune/comandă. Afișează date pe ecran. *Pag:* 41, 41, 42, 43, 44, 45... (practic în toate programele).

READ (CITESTE). Instrucțiune/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). Instrucțiune. Introduce comentarii în cadrul programului. *Pag:* 36, 38, 41, 42, 43, 44, 45, 47, 49, 50, 51, 52...etc.

RESTORE (RESTAUREAZĂ). Instrucțiune/comandă. Oferă posibilitatea de recitare a datelor dintr-o anumită linie DATA. *Pag:* 124.

RETURN(REÎNTOARCE-TE). Instrucțiune/comandă. Termină o subrutină și 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). Funcție. Generează un număr aleator mai mare sau egal cu 0 și mai mic ca 1. *Pag:* 36, 40, 49, 77, 91, 102, 119.

RUN (ALEARGĂ, dă-i drumul). Comandă/instrucțiune. Se începe execuția unui program. *Pag:* 36, 109.

SAVE (SALVEAZĂ). Comandă/instrucțiune. Memorează (salvează) fișierele (programe sau date) pe un suport extern. *Pag:* 108, 109.

SCREEN\$ (ECRAN). Se utilizează cu LOAD sau SAVE pentru a încărca și, respectiv, salva o imagine ecran. *Pag:* 108, 109.

SIN (SINus). Funcție. Produce sinusul unui unghi. *Pag:* 132.

STEP (pas). Se utilizează în cadrul unei linii de program FOR pentru a evidenția pasul cu care se modifică valoarea variabilei. *Pag:* 83, 84, 95, 98, 99, 101, 103, 123, 124, 131, 132.

STOP (STOP, oprește-te). Instrucțiune/comandă. Oprește programul într-un anumit punct. *Pag:* 36, 43, 45, 50, 55, 61, 65, 68, 114.

TAB (TABulare). Determină coloana pe care se va afișa într-o instrucțiune cu PRINT. *Pag:* 126.

THEN (atunci). Se utilizează într-o instrucțiune IF. Dacă este adevărată condiția ATUNCI instrucțiunea (instrucțiunile) care urmează după THEN se vor executa, iar dacă nu este adevărată se vor executa instrucțiunile începând cu linia următoare de program. *Pag:* 36, 37, 38, 42,43... (vezi IF).

TO (până la). Se folosește într-o instrucțiune FOR pentru a determina valorile indicelui din ciclul FOR-NEXT. *Pag:* 38, 40, 42... (vezi FOR).

USR (User SubRoutine). Funcție. Apelează o subrutină în cod sau produce adresa de început a uneia dintre cele 21 de secțiuni de memorie rezervate caracterelor grafice definit de utilizator (UDGuri). *Pag:* 90, 92, 95, 98, 115, 119, 125, 131.

VERIFY(VERIFICĂ). Comandă/instrucțiune. Verifică dacă un program a fost salvat corect, comparând programul din memorie cu cel citit de pe un suport extern. *Pag:* 108.

POSTFAȚĂ

Jocul - o cale spre libertate

"De-ar da Domnul să se nască un om care să-i învețe pe copiii noștri să trăiască și să gândească devreme, în schimb să citească târziu." (Goethe)

Criza profundă pe care o parcurge învățămîntul contemporan, cel puțin în civilizația occidentală, la limita căreia viețuim și spre care aspirăm, este o confruntare, cu efecte distructive, între *mijloace* și *scopuri*. Efortul pentru atingerea scopurilor educaționale este sistematic deturnat către dobîndirea mijloacelor prin care învățacelul poate fi educat. Copilul, epuizat prin efortul de asimilare a mijloacelor, scapă de fapt procesului educațional pentru a cărui împlinire nu mai rămîne disponibil.

Elevul învață să citească, dar nu mai are *cînd* să citească, învață diverse formalisme, dar nu întrevede la *ce* le-ar putea folosi, deprinde limbi străine, dar nu are *cînd* să le exerseze, învață istorie și geografie, dar nu-și poate face o reprezentare proprie despre ce ar putea fi un spațiu cultural.

Mijloacele consumă prea mult din disponibilitatea copilului către formare, în primul rînd datorită faptului că *nu corespund* psihologiei și mentalității vîrstelor la care se fac cei mai importanți pași ai procesului educațional

Orice exercițiu este un act gratuit, dar jocul, ca exercițiu, este unul *explicit gratuit*. Această evidență îl face acceptat, chiar dorit, de mentalitatea oricărui copil; și nu numai, deoarece conservă, chiar iluzoriu, sentimentul de libertate, de acțiune care nu constrînge din perspectiva unui scop. Cel care proiectează educația ca joc nu este obligat să inventeze și să pună în practică diverse tehnici adiționale care să suplinească lipsa de motivație a celui educat. Absența acestor procedee parazite scade riscul unor efecte nedorite, antieducaționale.

Rigiditatea sistemelor de învățămînt poate fi scuzată invocînd lipsa unor instrumente suficient de flexibile și de expresive, în absența cărora s-au dezvoltat uneori tehnici greoaie, inexpressive, neatractive și, în consecință, ineficiente.

Nu avem, se pare, nici posibilitatea de a ne întoarce la profunzimea actului educațional care, la vechii greci, punea pe același plan muzica, gimnastica și matematica într-o armonie pe care nu mai sîntem în stare să o declanșăm. Nu ne rămîne decît să ne punem speranța într-o înțeleaptă folosire a celui mai avansat obiect tehnic pe care l-am inventat - calculatorul. Dacă, folosindu-l, vom debloca învătămintul din impasul în care se află, vom avea un argument în plus să credem că nu am inventat degeaba această mașinărie.

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 virtuțile acestui instrument ca mijloc. Tot jocul este, în continuare, mijlocul cel mai eficient de atingere a unor scopuri educaționale. Putem ajunge în miezul unei problematici esențiale, stimulați de relaxarea oferită prin frumusețea gestului gratuit la care predispoziția îl face posibil.

Ideea de joc nu presupune obligatoriu calculatorul, dar se împlinește avîndu-l ca suport în procesul educațional, datorită flexibilității nelimitate de care dispune cel ce gîndește pedagogic într-un astfel de context.

Există și alte căi pe care ne putem apropia de calculator, căi pe care le și putem exploata pentru a ne atinge scopurile. Dar este foarte important să ținem cont de faptul că implicăm acest instrument în procesul de instruire a copilului, într-o perioadă în care evoluția personalității lui are o dinamică foarte mare. Riscurile la care-l expunem sînt maxime și sîntem obligați să gîndim acest proces cu cea mai mare grijă. Folosind jocul, ca anvelopă pentru atingerea scopului educațional, minimizăm riscurile la care expunem procesul de formare al copilului.

Un avantaj esențial este oferit și de capacitatea instrumentului folosit de a se adapta diversității comportamentale a copiilor. Iar jocul, ca interfață între educat și procesul de educare, sporește flexibilitatea întregului angrenaj. Capacitatea de adaptare la predispozițiile individuale este scopul fundamental al cercetărilor pedagogice moderne. Ne oferim o șansă în plus implicînd tehnica de calcul și maximizăm această șansă dacă folosim jocul ca principală formă de captare a interesului copilului în acest proces.

Ar fi bine să nu uităm că jocul este și unul din acele mijloace prin care omul poate supraviețui, pe tot parcursul vieții, tendințelor de înregimentare rigidă pe care comunitatea i le impune. Procesul educațional tradițional inhibă tocmai aceste predispoziții naturale salvatoare. Am putea atunci vedea în stimularea prin joc și o cale de a redobîndi o parte din libertatea pe care o credeam definitiv pierdută.

martie, 1992

Gh. Ștefan

CUPRINS

Cuvînt înainte	5	Calculatorul profesor de limbi străine	46
Cîteva precizări pentru profesori	6	Tabla, înmulțirii	49
Cum se utilizează GHIDUL	7	Calculatorul poet	49
Cîteva precizări privind modul de ținere a evidenței jocurilor și a infor- mațiilor referitoare la ele	9	Calculul mediilor și sortarea datelor ..	51
I. FUNDAMENTE ALE REALI- ZĂRII JOCURILOR PE CALCU- LATOR		Program de sortare	54
Rezolvarea problemei	12	Program de căutare	57
Un exemplu de realizare a unei dia- grame de tip arbore	14	Cifrarea mesajelor și dezlegarea enigmelor	59
Structuri de control	16	Descifrarea textelor codificate	61
Descrierea algoritmului	19	Clasament alfabetic	63
Scheme logice	20	Studiul frazei. Lingvistică cu calcula- torul	66
Blocurile utilizate pentru scheme logice	20	Frecvența literelor	67
Baze de numerație	28	Conversații cu calculatorul	69
Baza doi și octetul	32	Calculatorul vă ajută să scrieți scrisori și să trimiteți felicitări	70
II. ÎNCEPEM SĂ REALIZĂM JOCURI		Experimente și modele cu ajutorul calculatorului	76
Ghicește numărul	35	III. CUM SE POT REALIZA DESENE CU CALCULATORUL	
Cine este campionul?	39	Utilizarea culorilor	79
Calculatorul joacă numai cu prieteni	43	Cum se realizează programe cu culori	82
Calculatorul profesor	45	Program pentru realizarea de histo- grame	82
		Desene pe ecran	84

Grafica de rezoluție scăzută	86	Structura unui joc cu ecran grafic de prezentare	108
Utilizarea caracterelor grafice pentru desene	86	Animația	110
Programarea desenelor cu caractere grafice	88	Mișcarea pe verticală și pe orizontală	110
Cum se pot crea noi caractere grafice	89	Folosirea țintei	114
Cum se proiectează un caracter grafic	90	Cum se pot deplasa „spiridușii“ într-un decor	118
Generarea mai simplă a caracterelor grafice utilizator	91	IV. CUM SE POT REALIZA EFECTE SONORE ȘI MUZICA CU CALCULATORUL	
Crearea de „spiriduși“ de dimensiuni mai mari	96	Programarea sunetelor	121
Definiția de caractere grafice cu două culori	98	Efecte sonore	123
B. Grafica de rezoluție înaltă	98	Muzică cu calculatorul	124
Cum se pot colora figurile	100	Calculatorul-instrument muzical	125
Utilizarea ciclurilor în grafică	101	Calculatorul muzician	127
Proiectarea desenelor pe ecran	102	Să aplicăm tehnicile învățate: Jocul „Tarzan“	128
Calculatorul pictor	105	V. RĂSPUNSURI LA EXERCITIILE ȘI TEMELE PROPUSE	
Ecrane grafice	107		135

Editura *Agni*

vă propune pentru anul 1992:

Luminița State
Hello, BASIC
(Aplicații pe microcalculatoare)

o

Ion Diamandi
Calculatorul în bancă
(Matematica pentru gimnaziu cu calculatorul)

o

Ion Diamandi, Gheorghe Păun
40 de jocuri logice în BASIC

Vă puteți asigura din timp aceste cărți, precum și cartea de față, trimițând cuponul alăturat pe adresa:

Editura AGNI

CP: 30-107

BUCUREȘTI

Redactor: MIHAELA IONESCU

Grafica: EMIL BOJIN

Coperta: ANDREI TEODORESCU

Machetare și tehnoredactare: DAN ROMANESCU

Dat la cules: 06/IV/1992

B.T.: 20 /IV/1992

Coli tipar: 10



Tiparul executat sub comanda nr. 20239

Producător asociat: D.P.P.T.

Regia Autonomă a Imprimeriilor

Imprimeria „CORESI” — București

ROMANIA

NUMELE _____

PRENUMELE _____

LOCALITATEA _____

STR _____

NR _____ BL _____ ET _____ AP _____

JUDETUL _____ COD _____

Vă rog să-mi expediati prin colet poștal cu ramburs _____

_____ exemplare din cartea (cărțile) _____

DATA _____ SEMNATURA _____



CUM SĂ REALIZĂM JOCURI PE CALCULATOR

ISBN 973-95626-0-4

este un ghid practic pentru utilizarea calculatoarelor de tip **Sinclair Spectrum - HC, CIP, JET, COBRA, TIM** atât la școală, cât și în cadrul familial. Lucrarea se adresează profesorilor și părinților, dar în special copiilor care se inițiază în informatică.

Pentru învățarea folosirii calculatorului s-a ales un subiect extrem de iubit de copii: jocurile. Cartea propune activități atât distractive, cât și instructive cu calculatorul. Astfel, calculatorul devine pe rînd pictor, poet, compozitor, profesor de limbi străine. Toate acestea se obțin prin intermediul celui mai comod limbaj de programare pentru tipurile de calculatoare considerate și, în plus, recomandat începătorilor - **limbajul BASIC**.

De același autor:

Dialog cu viitorul (1988); Partenerul meu de joc, calculatorul (1988, 1989, 1990); LOGO - o nouă metodă de a învăța cu ajutorul calculatorului (1991); Din spectacolul informaticii-calculatorul personal (1992).

Editura *Agni*

Preț: 198 Lei