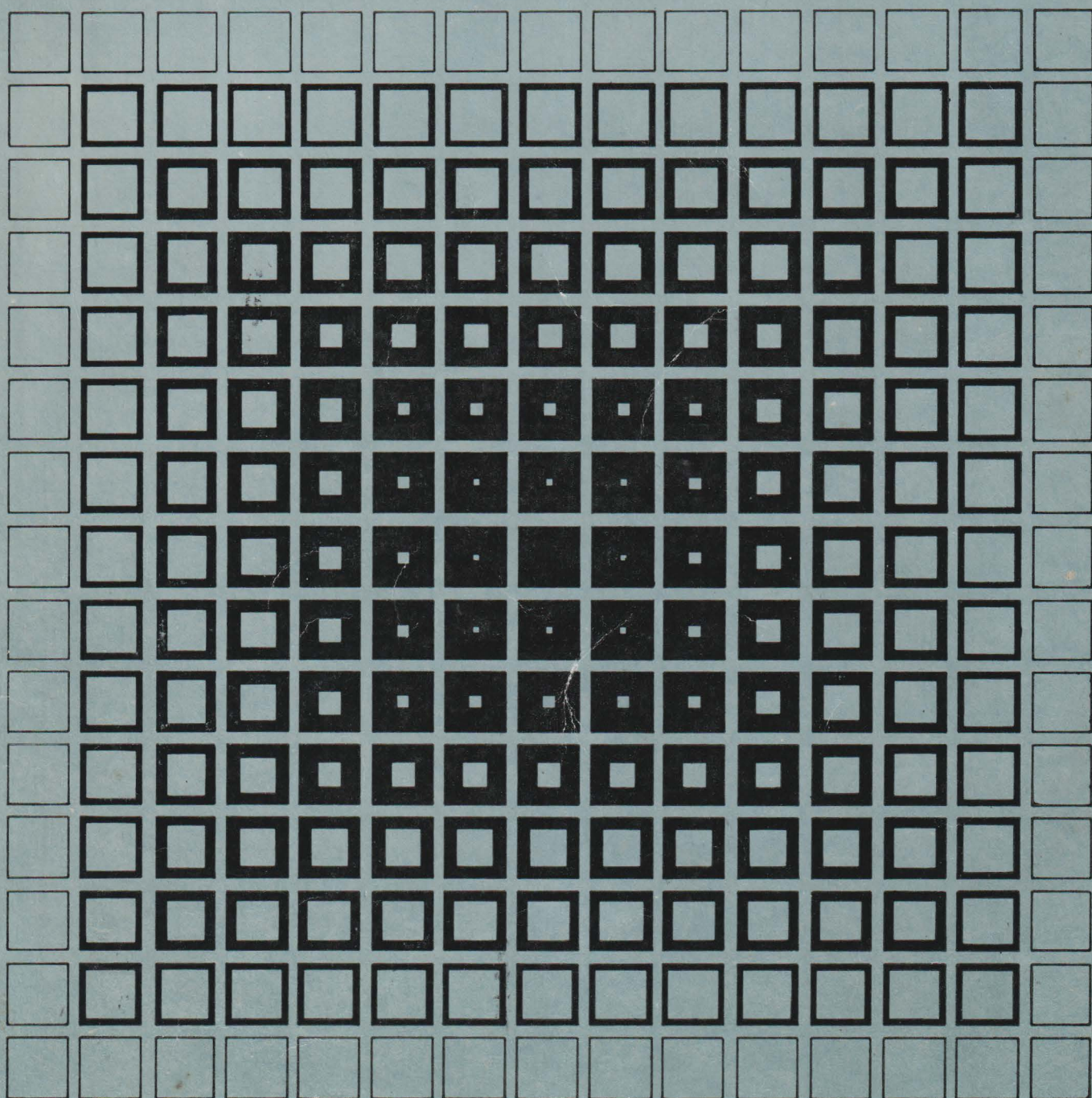


UNIUNEA TINERETULUI COMUNIST
COMITETUL CENTRAL

ELEMENTE DE INFORMATICĂ PENTRU CERCURILE TEHNICO — ȘTIINȚIFICE ALE ELEVILOR



UNIUNEA TINERETULUI COMUNIST
COMITETUL CENTRAL

ELEMENTE DE INFORMATICĂ PENTRU
CERCURILE TEHNICO—ȘTIINȚIFICE
ALE ELEVILOR

Lucrarea a fost coordonată de un colectiv format din
ing. STOICA ADRIAN, cerc. şt. DIAMANDI ION, st. CONSTANTINESCU CRISTIAN,
st. STĂNESCU NICOLAE şi st. LUTIC MIRCEA

Tiparul executat la I.P. ARTA GRAFICĂ
Bucureşti 1988



PREFAȚĂ

Dinamica dezvoltării economico-sociale a țării noastre, în conformitate cu hotărârile Congresului al XIII-lea al Partidului Comunist Român, în spiritul orientărilor și indicațiilor tovarășului Nicolae Ceaușescu, secretar general al partidului, impune formarea unor cadre cu înalte cunoștințe profesionale, capabile să soluționeze problemele complexe ale progresului tehnic, de a introduce rapid în producție cele mai noi cuceriri ale științei și tehnicii.

Pornind de la teza de largă perspectivă privind integrarea învățământului cu cercetarea științifică și producția, Uniunea Tineretului Comunist acționează, prin forme și acțiuni specifice, în strînsă colaborare cu ceilalți factori cu responsabilități, pentru continua perfecționare a pregătirii profesionale a tinerilor, pentru dobîndirea, încă de pe băncile școlii, a cunoștințelor teoretice și practice necesare utilizării celor mai noi tehnologii în diferite domenii ale economiei naționale.

Pe linia acestor preocupări se înscriu și taberele naționale de informatică și calculatoare, organizate de CC al UTC, reunind anual elevi cu rezultate deosebite în activitatea cercurilor de informatică din licee.

Ediția a III-a a taberei, desfășurată în 1987 la Tîrgu Mureș, a prilejuit instruirea elevilor participanți în domeniul utilizării calculatoarelor personale.

Beneficiind de prezența în cadrul taberei a unor prestigioși specialiști din învățămîntul superior, institute de cercetare, centre de calcul electronic și întreprinderi industriale, precum și de o bază materială deosebită, cuprinzînd cele mai performante calculatoare personale realizate de industria noastră: FELIX-PC, TPD JUNIOR, HC-85, TIM S, elevii au participat la un amplu program de informare și documentare științifică, aplicațiile realizate urmărind în principal următoarele direcții: grafică cu calculatorul, instruire asistată de calculator, metode numerice de calcul, proiectare asistată de calculator și elemente de gestiune.

Totodată, prin vizitele de documentare desfășurate în diverse unități social-economice, participanții la tabără au avut posibilitatea să cunoască unele realizări deosebite privind implementarea și utilizarea echipamentelor de tehnică de calcul în conducerea proceselor tehnologice și prelucrarea datelor.

Lucrarea de față prezintă în rezumat considerații teoretice asupra unor limbaje folosite în programarea calculatoarelor personale, precum și modul de utilizare a acestora în diferite aplicații, realizate de elevi la ediția a III-a a Taberei naționale de informatică și calculatoare.

Lucrarea se adresează în principal membrilor cercurilor de informatică din cadrul caselor de știință și tehnică pentru tineret și din unitățile de învățămînt.

Organizatorii Taberei naționale de informatică și calculatoare mulțumesc pe această cale celor care și-au adus o importantă contribuție la desfășurarea în cele mai bune condiții a acestei acțiuni: Comitetul Județean Mureș al PCR, Institutul de Tehnică de Calcul și Informatică București, Facultatea Automatică-IPB, Întreprinderea de Calculatoare Electronice București, Întreprinderea de Echipamente Periferice București, Centrul Teritorial de Calcul Electronic Tîrgu Mureș, Întreprinderea Electromureș și Inspectoratul Școlar al județului Mureș.

CUPRINS

Capitolul I	— Colocviu-dezbateri	5
Capitolul II	— Utilizarea avansată a calculatoarelor HC și TIM-S în limbaj BASIC	9
Capitolul III	— BETA BASIC — extensie a limbajului BASIC	21
Capitolul IV	— Limbajul PASCAL. Particularități de implementare pe calculatoarele HC 85 și TIM-S	27
Capitolul V	— Introducere în sistemul dBASE	43
Capitolul VI	— Programe aplicative	49
6.1.	Grafică	49
6.2.	Instruire asistată de calculator	64
6.3.	Metode numerice de calcul	70
6.4.	Elemente de proiectare asistată de calculator	76
6.5.	Elemente de gestiune	79

CAPITOLUL 1

COLOCVIU—DEZBATERE

Revista « Știință și tehnică » a organizat cu ocazia Taberei naționale de informatică pentru elevii de liceu, un colocviu-dezbateri cu invitați de prestigiu în domeniul informaticii: prof. univ. dr. docent ing. Edmond Nicolau, care a vorbit participanților despre viitorul calculatoarelor, structuri și componente noi; prof. univ. dr. docent Solomon Marcus, care s-a referit la bazele matematice ale informaticii, algoritmi și limbaje de programare; dr. ing. Dan Roman, director adj. științific ITCI, cu câteva spicuri din controversata și surprinzătoare istorie a calculatoarelor sau 40 de ani care au schimbat fața lumii; dr. mat. Petru Pepelea, directorul Centrului teritorial de calcul din Tg. Mureș, cu o prezentare a perspectivelor și direcțiilor de dezvoltare ale CTCE; cercet. șt. Iulian Cîmpeanu, ITCI, cu o expunere despre animația bi și tridimensională pe calculator.

În continuare invitații au răspuns la o serie de întrebări puse de elevi, întrebări care au demonstrat interesul acestora pentru problemele de actualitate din domeniul calculatoarelor, iar în încheiere s-a organizat un concurs fulger pe teme de cultură generală tehnică, concurs care s-a bucurat de multă popularitate, câștigătorii fiind răsplătiți cu frumoase premii.

Prezentăm în continuare câteva din ideile importante expuse de prestigioșii invitați.

Calculatorul, la limita inteligenței artificiale

*Prof. Univ. dr. docent, ing.
Edmond Nicolau*

Apariția calculatorului a adus după sine o adevărată explozie tehnologică în toate domeniile, în zilele noastre fiind greu de conceput unul măcar lipsit de intervenția — fie chiar și indirectă — a tehnicii de calcul. Dar calculatorul a adus și ceva în plus activității umane: i-a sporit eficiența constituindu-se într-o adevărată prelungire a inteligenței omului. Calculatorul a învățat să vorbească, să cînte, să deseneze și să proiecteze, a început să creeze roboți inteligenți înzestrați cu putere de decizie și cu simțuri localizate în traductoare specifice. Calculatorul poate să conțină astăzi zeci de mii de microprocesoare încorporate, să fie înzestrat cu limbaje complexe. În sprijinul acestor afirmații putem exemplifica cele mai recente realizări în domeniul calculatoarelor și anume:

calculatoarele sistolice — mai multe calculatoare unite între ele — și hipercubul boolean — o mașină cu 65536 microprocesoare de câte 4 Ko — care prezintă ca performanțe o viteză de calcul de peste un miliard de instrucțiuni pe secundă!

În sfîrșit, calculatorul poate fi la rîndu-i creator și proiectant de alte calculatoare. Putem afirma că el, calculatorul, dacă este programat corect, știe să facă aproape « orice », limita acestui « orice » fiind dată de contribuția strict umană la actul de creație. Culmile actuale ale calculatorului trebuie privite ca performanțe ale inteligenței umane, omul fiind în acest caz în dubla postură de creator și beneficiar utilizator al celei mai spectaculoase invenții a secolului nostru.

În ultimii ani, aplicațiile inteligenței artificiale (IA) devin tot mai numeroase, ele pătrunzînd în practica curentă din multe domenii. Cercetarea în domeniul IA se desfășoară pe mai multe direcții. Unele probleme, în aparență elementare, continuă să preocupe pe specialiști; aici se înscrie, de exemplu, problema recunoașterii formelor, fie ele scrise, vorbite sau de altă natură (electrice, acustice, etc). Dar fără îndoială că cel mai interesant aspect este astăzi constituit din sistemele robot cu braț-sistem expert. Sistemele expert pot fi definite drept sisteme cibernetice în care — într-o bancă de date — s-a acumulat o cantitate mare de date privind un anume domeniu. Pe lângă această bază de date, sistemul este prevăzut și cu un motor de inferențe, un sistem care plecînd de la anumite premise, trage concluziile logice. Se afirmă că aceste sisteme sînt atît de evolute încît cu ajutorul lor s-a putut descoperi un zăcămint de petrol pe care experții umani nu îl identificaseră. Desigur că utilizarea sistemelor expert necesită și limbaje de programare adecvate care par a fi LISP și PROLOG.

Problema centrală a IA rămîne utilizarea cît mai eficientă a resurselor disponibile: de la mașina considerată capabilă doar să răspundă la întrebări, s-a trecut azi la mașina care, plecînd de la axiomele aritmeticii, deduce singură — pe baza unei euristici date — că există numere prime, că există pătrate ale numerelor și, mai mult chiar, generează conjecturi în domeniul teoriei numerelor. Cu alte cuvinte, formulează întrebări la care omul sau mașina răspunde. Sîntem în prezența unui fenomen tipic cibernetic în cadrul căruia asistăm la o accelerare a ritmului, tocmai prin aceste mașini tot mai puternice, care fac posibilă înaintarea tot mai rapidă pe calea cunoașterii.

Calculatorul: O revoluție în modul de gândire

Prof. univ. dr. docent Solomon Marcus

Calculatorul înseamnă azi în primul rând programe inteligente. La baza acestor programe se află o gândire specifică, de natură algoritmică, un anumit mod de reprezentare a datelor și a cunoștințelor, pe care trebuie să-l ameliorăm mereu, deoarece algoritmi disponibili sînt, de cele mai multe ori, prea costisitori, în sensul că cer prea mult timp de calculator și/sau prea mult spațiu de memorie (a calculatorului). Așa se și explică faptul că informatica devine tot mai mult un capitol al unei științe mai vaste, inteligența artificială, iar limbajele de programare clasice sînt tot mai mult concurate de limbajele inteligenței artificiale, LISP și PROLOG.

Gîndirea algoritmică, prin ceea ce are ea mai bun, este gîndire creatoare deoarece a gîndi algoritmic înseamnă în primul rînd a fi capabil să descoperi singur algoritmul de reprezentare a unui proces sau de rezolvare a unei probleme, să ameliorezi unii dintre algoritmi existenți, să-i înlocuiești eventual cu alții, mai avantajoși. Aceasta este problema la ordinea zilei în studiul programării liniare, de exemplu, dar și în numeroase chestiuni teoretice, cum ar fi testarea primalității numerelor întregi. Această ultimă problemă și-a dovedit recent aplicații uimitoare în realizarea codurilor secrete, atît de importante în comerțul exterior, în diplomatie și în armată.

Rezolvarea problemelor globale ale omenirii nu este posibilă fără ajutorul informaticii. Hrana și materiile prime, energia și mediul nu pot face saltul așteptat fără realizarea unui progres științific substanțial în investigarea lor; acest progres include o capacitate superioară celei existente în prelucrarea datelor și în reprezentarea cunoștințelor. Pentru a da un singur exemplu, prognozelor meteorologice pretind tocmai o atare capacitate, noi modele matematice și algoritmice, mijloace de calcul cît mai perfecționate, pe baza cărora să se poată prelucra într-un timp cît mai scurt și cu rezultate cît mai eficiente o considerabilă cantitate de date.

Modelele matematice și computaționale au astăzi o miză socială deosebită. Pe baza lor se decide direcția pe care o vor lua uriașe mijloace financiare folosite în planificarea și organizarea economică, administrativă, sanitară. Comanda socială la adresa informaticii este atît de gravă, încît numeroase științe își reorientează unele preocupări în funcție de această situație. Matematica este în primul rînd vizată în această privință. Raza ei de acțiune cuprinde azi cvazi-totalitatea domeniilor de activitate, dar această performanță este realizată în mare măsură în colaborare cu calculatorul. Multe dintre premiile Nobel acordate pentru științe economice

s-au referit la cercetări puternic matematizate, în aceeași situație aflîndu-se și unele premii Nobel pentru medicină (de exemplu, tomografia computerizată are la bază modele matematice de înaltă complexitate). În cucerirea spațiului și în ameliorarea mediului, în industrie și comerț, în organizarea spitalelor și în procesul de învățămînt, în transporturi și comunicații, în realizarea programelor energetice calculatorul și gîndirea algoritmică asociate devin esențiale. Toate profesiile se restructurează în funcție de această nouă realitate, pe care o numim a doua revoluție industrială, o revoluție în care, alături de binomul *materie-energie* (caracteristic primei revoluții industriale) apare un al treilea element, *informația*.

Cele de mai sus ne îndreptătesc să considerăm că omenirea intră acum într-o nouă etapă, în care, alături de alfabetizarea tradițională, constînd în învățarea limbii materne, a scrisului și cititului, apare o a doua alfabetizare, de natură computațională, constînd în învățarea comunicării cu calculatorul. Pentru noile generații, succesul în profesie (indiferent care va fi aceasta) este esențial condiționat de ambele alfabetizări.

Tradiție, realizări și perspective în tehnica de calcul

Dr. ing. Dan Roman

Este un fapt evident că istoria calculatoarelor, deși foarte scurtă, numărînd numai patru decenii, este în același timp și deosebit de spectaculoasă. Ce distanță tehnologică imensă separă un calculator de dimensiuni impresionante din « preistoria » tehnicii de calcul și un flexibil calculator personal actual care încapă cu ușurință într-o servietă avînd performanțe de sute de ori mai mari!

Să punctăm cîteva aspecte relevante din istoria calculatoarelor considerînd momente separate de cîte un deceniu:

În 1947 existau în lume doar 6 calculatoare instalate, IBM, singurul producător de calculatoare hotărînd să nu mai investească în acest domeniu pe motivul că piața prezintă un nivel prea scăzut.

După cîteva luni se naște tranzistorul care provoacă un salt tehnologic spectaculos și apariția unei noi generații de calculatoare.

După 10 ani, în 1957, se naște firma DEC care va deveni lideră pe plan mondial în realizarea de minicalculatoare și a doua firmă de tehnică de calcul din lume după IBM.

În 1967 pe ușa magazinelor de calculatoare apare interdicția de intrare pentru tinerii sub 18 ani. Nu mai este însă mult timp pînă la apariția microprocesorului.

În 1976 este prezentat primul calculator personal: ALTAIR 8800. Este momentul în

care calculatorul devine bun al întregii omeniri nu numai al unui grup restrâns de specialiști, este momentul în care calculatorul își dezvăluie multiplele sale posibilități de aplicație care includ folosirea lui în familie și în școală.

Pentru 1987 nu se știe încă care este evenimentul care peste ani se va dovedi că a însemnat o revoluție: poate dezvoltarea calculatoarelor de generația a 5-a, poate apariția calculatoarelor personale pe 32 de biți sau poate apariția primelor forme de calculatoare biologice?

Tehnica de calcul românească se poate mândri cu un trecut bogat în realizări. Trebuie amintit că primul calculator românesc, realizat de un colectiv condus de Toma Victor, a fost printre primele calculatoare apărute în țările socialiste. O importantă contribuție la fundamentarea cercetărilor în informatică a adus-o școla românească de matematică, academicianul Grigore Moisil fiind în acest sens un reprezentant de seamă.

Tînăra industrie de tehnică de calcul românească (ea numără doar două decenii de la apariție) se mîndrește cu cîteva realizări importante: de la sisteme de calcul medii mari (FELIX C-256/512/1024) și minicalculatoare (familiile INDEPENDENT și CORAL), la microcalculatoare (familia FELIX M118, JUNIOR, CUB) și calculatoare personale (aMIC, PRAE, HC85, TIMS, FELIX PC) precum și echipamente periferice performante.

Inițierea institutului de cercetare de profil în anul 1968 a însemnat un salt calitativ în crearea condițiilor pentru o cercetare românească originală destinată producției autohtone de calculatoare și echipamente periferice. De

numele institutului se leagă de altfel dezvoltarea familiei de sisteme de calcul de capacitate medie mare FELIX, proiectarea minicalculatoarelor INDEPENDENT (I 100, I 102F, I 106), a unității de bandă magnetică cu transfer continuu, a sistemelor de proiectare asistată de calculator, pentru a enumera numai cîteva realizări. Printre ultimele realizări se poate sublinia proiectarea și fabricarea de calculatoare personale, echipamente dedicate în special utilizării de către tineri. Astfel au apărut: aMIC, primul calculator personal românesc, PRAE, CE 119, TIM-S și mai recent COBRA. De altfel pentru perioada următoare programele de dezvoltare prevăd îmbunătățirea și extinderea tipurilor existente: PRAE MAX cu compatibilitate CP/M, HC și TIMS cu disc flexibil, COBRA cu compatibilitate dublă, Sinclair Spectrum și CP/M, FELIX PC cu extensie de memorie (640 Ko), discuri fixe de capacitate mare, și dispozitiv de tip «șoarece», facilități pentru cuplarea calculatoarelor în rețele și, pentru toate tipurile amintite multe, multe programe de aplicație.

În același timp institutul se preocupă îndeaproape de pregătirea tinerei generații, capabilă să utilizeze noile și modernele mijloace. Dealtfel ITCI a organizat, în colaborare cu CNOP, prima tabără de calculatoare pentru copii chiar din ianuarie 1985, echipamentele utilizate fiind însăși calculatoarele personale din seria zero. Tradiția a fost continuată și în anii următori. În plus în institut funcționează permanent un cerc de calculatoare, în care copii de diferite vârste se familiarizează cu utilizarea celor mai noi tipuri de calculatoare.

CAPITOLUL 2

UTILIZAREA AVANSATĂ A CALCULATOARELOR HC SI TIM-S ÎN LIMBAJ BASIC

2.1. Organizarea memoriei interne

Cunoașterea organizării memoriei interne este utilă în perspectiva unei programări avansate, acest lucru permițând modificarea unor octeți ai memoriei a căror amplasare este cunoscută. Se ajunge astfel la rezultate mult mai greu de obținut utilizând tehnicile uzuale ale limbajului BASIC. Stăpînirea anumitor tehnici legate de organizarea memoriei interne oferă în plus posibilitatea intervenției directe în repartiția diferitelor zone ale memoriei.

Se cunoaște faptul că memoria internă se compune dintr-o parte care conține programe fixe și date (inscripționate în memorie de către fabricantul calculatorului) și alta la dispoziția utilizatorului. Din prima utilizatorul poate numai citi informațiile, neavînd posibilitatea și de a le modifica. De aceea această parte a memoriei se numește memorie ROM (Read Only Memory) iar pentru calculatoarele HC și TIM-S ea are o mărime echivalentă cu 16 KO (aproximativ 16000 octeți). Cu toate că această memorie poate fi numai citită (din această cauză ea se mai numește și memorie moartă) studiul ei poate fi deosebit de util prin identificarea unor anumite rutine care vor putea fi apoi încorporate în diferite programe personale ale utilizatorului ce vor deveni astfel mai performante. Din cealaltă parte a memoriei, RAM (Random Acces Memory), utilizatorul poate citi dar și scrie. Pentru calculatorul HC mărimea acestei memorii este de 48 KO iar pentru calculatorul TIM-S este de 64 KO. Cu toate că diferă astfel ca mărime, organizarea lor este similară, diferența de 16 KO fiind utilizată în scopul mutării conținutului memoriei ROM la conectarea calculatorului în această parte a memoriei. În acest fel cei 16 KO de memorie RAM nu mai sînt practic la dispoziția utilizatorului decît în scopul modificării unor părți ale sistemului BASIC. De aceea vom trata unitar problema organizării memoriei interne la calculatoarele HC și TIM-S făcînd abstracție de memoria suplimentară la cel de-al doilea calculator. Cu toate că se spune că memoria RAM este la dispoziția utilizatorului, acest lucru este adevărat numai în general, sistemul efectuînd în mod curent în această zonă alocări în vederea stocării unor date sau protecției unor zone în care vor figura rezultate provizorii sau variabile definite de utilizator,

2.2. Accesul asupra memoriei

PEEK este comanda care permite citirea conținutului unui octet de memorie iar POKE este comanda care permite modificarea acestui conținut. Conținutul octetului de modificat este o valoare cuprinsă între 0 și 255. Exemplificînd utilizarea comenzii de citire a memoriei prin introducerea programului P1 la momentul punerii sub tensiune a calculatorului, ca rezultat al rulării acestui program se va obține conținutul octeților care codifică caracterul "A". Acest lucru se întîmplă deoarece octetul de la adresa 65368 marchează începutul zonei rezervate caracterelor grafice utilizator (UDG) și în mod normal acest octet este încărcat cu caracterul "A", dacă utilizatorul nu l-a modificat în prealabil. Același rezultat l-am fi obținut și cu programul P2 citind tabela generală a caracterelor care este situată la începutul memoriei ROM.

Programul P1

```
10 FOR I=65368 TO 65375
20 PRINT PEEK I;" ";
30 NEXT I
```

Rezultatul rulării va fi:

```
0 60 66 66 126 66 66 0
```

Programul P2

```
10 FOR I=15880 TO 15887
20 PRINT PEEK I;" ";
30 NEXT I
0 60 66 66 126 66 66 0
```

Dacă facem POKE 65368,255 și comandăm apoi afișarea caracterului grafic corespunzător tastei "A" (în modul G) vom vedea afișat un caracter "A" modificat cu o bară deasupra: ansamblul de biți care compun primul octet au fost puși pe "1" înnegrind astfel partea superioară a zonei caracterului.

Deoarece nu putem să facem PEEK și POKE decît cu valori cuprinse între 0 și 255 (numărul maxim care se poate reprezenta cu 8 biți) se pune problema cum putem totuși reprezenta un număr mai mare decît 255, de exemplu una din adresele memoriei RAM. În acest caz va fi necesar să codificăm valoarea adresei pe doi octeți, cel de-al doilea avînd o pondere (semnificație) de 256 de ori superioară primului (invers ca ponderea cifrelor în reprezentarea

numerele* — vezi nota de subsol). De exemplu dacă vrem să știm care este RAMTOP-ul în cazul calculatorului HC vom citi conținutul adresei variabilei sistem RAMTOP (la 23730) care ne va indica adresa ultimului octet din zona sistemului BASIC și care este o valoare pe doi octeți.

Deci `PRINT PEEK 23730 + 256 * PEEK 23731` calculatorul răspunzându-ne cu valoarea 65367.

Comanda `PEEK` fiind destinată citirii unui octet de memorie utilizarea ei nu prezintă nici un pericol în derularea programului respectiv chiar dacă ea furnizează o informație eronată sau inutilă. Cu ajutorul ei se pot citi unul după altul octeții memoriei ROM pornind din oricare loc al acestei memorii. Comanda `POKE`, însă, trebuie utilizată cu mult mai multă precauție, în măsura în care ea modifică conținutul memoriei. Așa cum am văzut pe tot ansamblul octeților memoriei ROM comanda `POKE` este inoperantă (acești octeți neputând fi modificați), în schimb dacă comanda `POKE` va modifica abuziv una din variabilele care servesc gestiunii sistemului, funcționarea calculatorului va fi perturbată într-un mod în care nu totdeauna mai este controlabil.

2.3. Organizarea memoriei RAM

Memoria ROM ocupă adresele între 0 și 16383, prima adresă accesibilă a memoriei RAM fiind deci 16384. În fig. 1 se arată organizarea memoriei RAM, repartizarea ei în diverse zone dintre adresele 16384 și 65535.

După cum se poate observa din fig. 1, primele trei zone de memorie RAM, zona de afișaj, zona de atribute și bufferul de imprimantă, ocupă aproape 7500 octeți. Toți octeții care compun această parte a memoriei pot fi modificați cu ajutorul comenzii `POKE`, dar este imposibil de a se mări sau reduce numărul de octeți considerați. Acest lucru se întâmplă deoarece în cazul calculatoarelor avute în vedere nu este posibilă creșterea memoriei disponibile pentru variabile prin diminuarea rezoluției și nu se poate rezerva prin program într-un mod durabil o parte din memoria de afișaj pentru salvarea de valori sau date. Într-adevăr, va fi de ajuns un CLS sau o reeditare de liste și ansamblul acestor date va fi șters. După aceste trei zone urmează zona variabilelor de sistem (VS) cuprinsă între octeții 23552 și 23733. Variabilele sistem determină poziția fiecărei zone de lucru, fiind diferite față de variabilele utilizator care sînt plasate într-o zonă de memorie care le este rezervată. Următoarele zone ale memoriei RAM sînt: zona programului BASIC, zona variabilelor utilizator și stiva.

* Notă: în memorie adresa se păstrează cu octetul cel mai puțin semnificativ în stînga. De exemplu: $40000 = 113 \times 256 + 72$

Dacă aceasta este o adresă o vom găsi în memorie la adresa adr astfel: la adr 72
la adr+1 113

2.3.1. Zona de afișaj

Pe ecranul asociat calculatoarelor HC și TIM-S se pot afișa 24 de linii a câte 32 de caractere, cu restricții pentru ultimele două linii care servesc, în principiu, la afișarea rezultatelor rulării programelor. Un caracter este afișat în oricare din cele $24 \times 32 = 768$ zone caracter (căsuțe, celule) care acoperă ecranul iar fiecare zonă caracter este constituită din $8 \times 8 = 64$ de puncte elementare (pixeli). O zonă caracter este formată deci din 8 octeți, octetul de afișaj reprezentînd o linie de puncte elementare. În memorie acestor octeți le corespunde zona de afișaj care ocupă memoria RAM de la adresa 16384 la adresa 22527 fiind constituită deci din 6144 octeți (768 de zone caracter $\times 8$ octeți pentru fiecare zonă caracter = 6144). Dacă în cadrul unui octet, care reprezintă o parte a zonei caracter un bit este pus pe 1 atunci acel punct va apărea pe ecran în culoarea inversă față de culoarea fondului (PAPER) zonei respective iar dacă bitul este pus pe 0 atunci acel punct va apărea pe ecran în aceeași culoare ca și a fondului zonei (deci nu se va vedea).

Pentru înțelegerea organizării zonei de afișaj se poate încerca următorul exemplu instructiv: se va umple întreg ecranul cu orice caracter, apoi se va salva ecranul sub forma de fișier pe caseta magnetică (`SAVE "<nume>" SCREEN$`) apoi se va proceda la încărcarea fișierului de pe caseta magnetică. Se va observa că reconstituirea ecranului nu se face într-o singură direcție (începînd de sus în jos, dreapta sau stînga) ceea ce înseamnă că octeții nu se succed în mod regulat în memorie. Structura generală a zonei de afișaj este, într-adevăr, următoarea: ecranul este divizat în trei subzone de câte 8 linii de caractere (zona 1 conține liniile de la 0 la 7, a doua liniile de 8 la 15 iar a treia liniile de la 16 la 24), iar reconstituirea ecranului se face pe zone, mai întîi zona 1, apoi după terminarea ei zona 2 și în sfîrșit zona a treia. În fiecare din aceste subzone, baleiajul se efectuează nu prin epuizarea liniilor caracter, una după alta, ci după a opta linie octet din cadrul fiecărei linii caracter, sau altfel spus: la al 6-lea baleiaj în cadrul unei zone, de exemplu, nu primele 6 linii caracter vor fi reconstituite integral ci fiecare din cele 8 linii caracter va fi reconstituită într-o proporție de trei sferturi ($6/8 = 3/4$). Astfel pentru a trasa o linie orizontală în partea superioară a ecranului este suficient să se încarce primii 32 octeți de afișaj cu valoarea 255 (adică toți biții din cadrul fiecărui octet sînt puși pe 1) așa cum este făcut în programul P3.

P3

```
10 FOR I=16384 TO 16384+31
20 POKE I,255
30 NEXT I
```

Și la fel, pentru a trasa o linie verticală în partea stîngă a ecranului, este suficient să

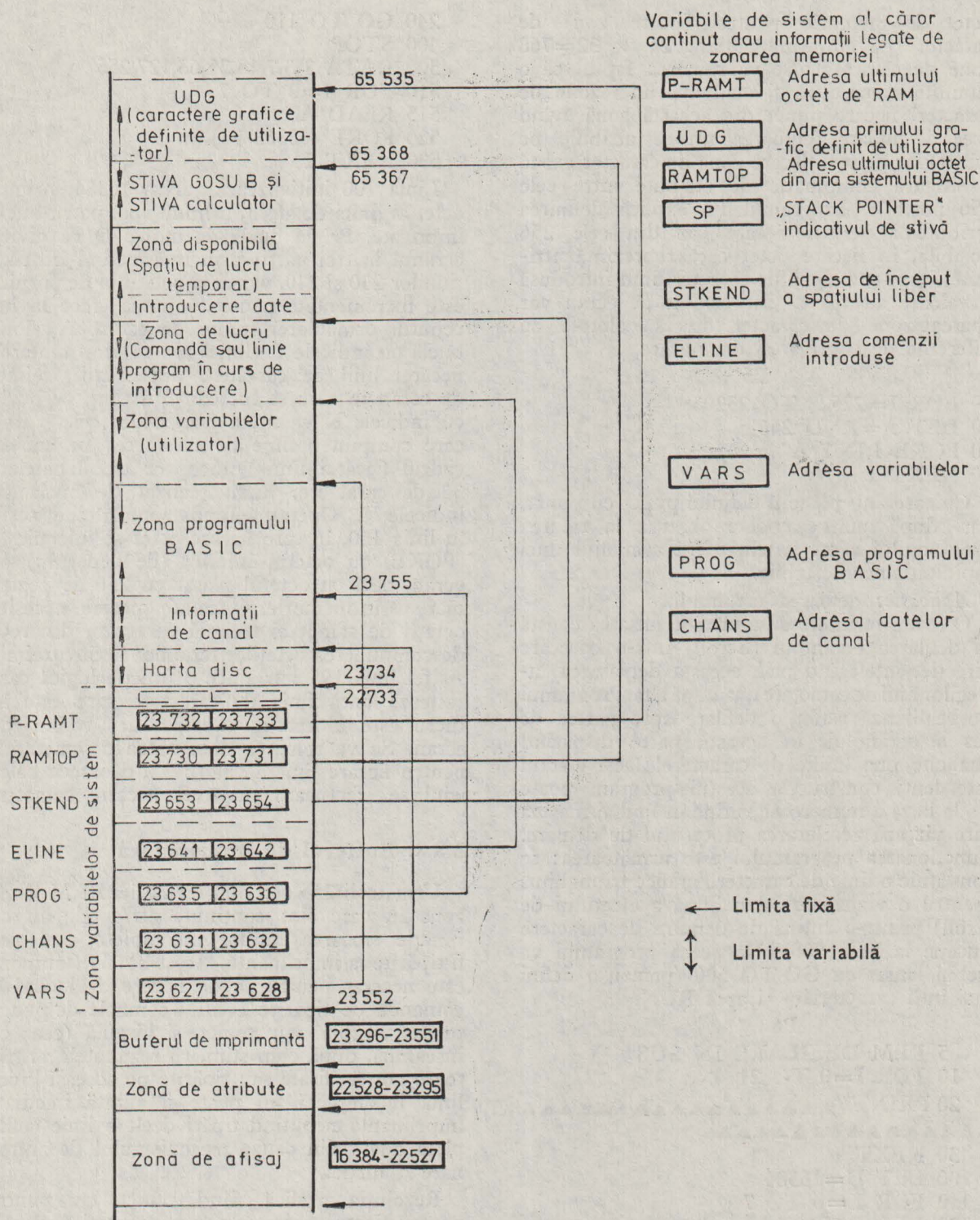


Fig. 1 — Organizarea memoriei RAM

introducem programul P4, prin care calculatorul va sări câte 32 de caractere de fiecare dată, fiind de ajuns ca un singur bit (punct) din cadrul octetului să fie pus pe 1 pentru a se trasa linia.

```
P4
10 FOR I=16384 TO 22400 STEP 32
20 POKE I,1
30 NEXT I
```

2.3.2. Zona de atribute

Atributele definesc modul în care va fi afișată fiecare zonă de caracter, în speță care va fi culoarea fondului afișării (PAPER), care va fi culoarea de afișare a caracterului (INK), contrastul (BRIGHT) și dacă va pulsa sau nu (FLASH). Zona de memorie RAM corespunzătoare atributelor este cuprinsă între octeții 22528 și 23295, fiind deci atribuit câte un

P5

P6

Deasemenea cunoașterea octeților din bufferul de imprimantă poate prezenta interes și prin faptul că astfel se poate realiza o imprimare cu caracteristici superioare care ar atenua diferența calitativă între afișarea pe ecran (care poate fi și în culori) și tipărirea la imprimantă. Astfel încărcându-se în mod adecvat octeții din zona bufferului de imprimantă va fi posibilă realizarea unei corespondențe între o culoare de pe ecran și o nuanță gri pe hîrtie de densitate echivalentă cu culoarea (mai puternică pentru culori închise și mai puțin puternică pentru culori mai deschise). Pentru realizarea acestui lucru este suficientă determinarea atributului zonei caracter de pe ecran care se dorește a se reproduce pe hîrtie și apoi găsirea corespondenței între culoarea cernelii și nuanța de gri prin anumite valori cuprinse între 1 și 255.

Pentru a ilustra aceasta tehnică vom da ca exemplu programul P7 care are ca efect încărcarea secvențială a celor 256 octeți ai zonei bufferului de imprimantă cu valori alternative. Grație acestui program, fie modificînd pasul de incrementare a octeților, fie modificînd valorile cu care se încarcă octeții se vor obține mai multe varietăți de gri, prin a căror suprapunere vor rezulta diverse efecte grafice pe hîrtia de imprimantă.

P7

```
10 LET A=1
20 LET S=0
30 FOR I=0 TO 255
40 IF S=32 THEN LET A=-A: LET
S=0
50 POKE (23297+I),136*(A>0)+34
*(A<0)
60 LET S=S+1
70 NEXT I
80 LPRINT
90 GO TO 30
```

2.3.4. Zona programului Basic

Spre deosebire de zonele de memorie precedente care au o adresă de început fixă, începutul zonei program este condiționată de existența extensiei de disc flexibil. În absența acesteia programul Basic începe la octetul 23755. Oricum adresa de început a zonei program o putem afla interogînd variabila de sistem situată la adresele 23635 și 23636. În vederea unui lucru eficient cu octeții zonei program trebuie cunoscută codificarea liniilor de program în calculator. Cu ajutorul programului P8 care comandă afișarea simultană a fiecărui octet cu conținutul său se va înțelege mult mai bine structura acestora.

P8

```
5 REM LINIA1
10 FOR I=23755 TO 23830
20 PRINT I;" ";PEEK I;TAB 15;
CHR$ PEEK I
30 NEXT I
```

Rezultatul rulării acestui program va fi:

23755	0	?
23756	5	?
23757	8	
23758	0	?
23759	234	REM
23760	76	L
23761	73	I
23762	78	N
23763	73	I
23764	65	A
23765	49	1
23766	13	
23767	0	?
23768	10	?
23769	27	?
23770	0	?
23771	235	FOR
23772	73	I
23773	61	=
23774	50	2
23775	51	3
23776	55	7
23777	53	5
23778	53	5
23779	14	?
23780	0	?
23781	0	?
23782	203	THEN
23783	92	/
23784	0	?
23785	204	TO
23786	50	2
23787	51	3
23788	56	8
23789	51	3
23790	48	0
23791	14	?
23792	0	?
23793	0	?
23794	22	

După cum se poate observa atît din rezultatele rulării programului P8 cît și din fig. 2 fiecare linie de program Basic are următoarea formă: pe primii doi octeți se reprezintă numărul de linie respectiv (primul octet este cel mai semnificativ), următorii doi octeți codifică lungimea liniei respective (numărul de caractere) astfel: un caracter este reprezentat pe un octet la fel ca și un cuvînt cheie care se reprezintă tot pe un octet (linia 5 are 7 caractere + 1 pentru CR). De remarcat că ponderea celor doi octeți nu este aceeași în grupa care indică numărul de linie și în grupa care indică lungimea sa. Constantele numerice și variabilele se codifică mai complicat. În program o constantă numerică este urmată de forma sa binară (de exemplu $23755=203+256 \times 92$), utilizîndu-se caracterul "number" (codul ASCII 14, vezi octeții 23779 și 23791) urmat de 5 octeți ai numărului însuși (vezi octeții 23780—23784).

O variabilă are un format diferit în funcție de natura ei. Pentru variabile numerice, al căror nume este o singură literă, codificarea se

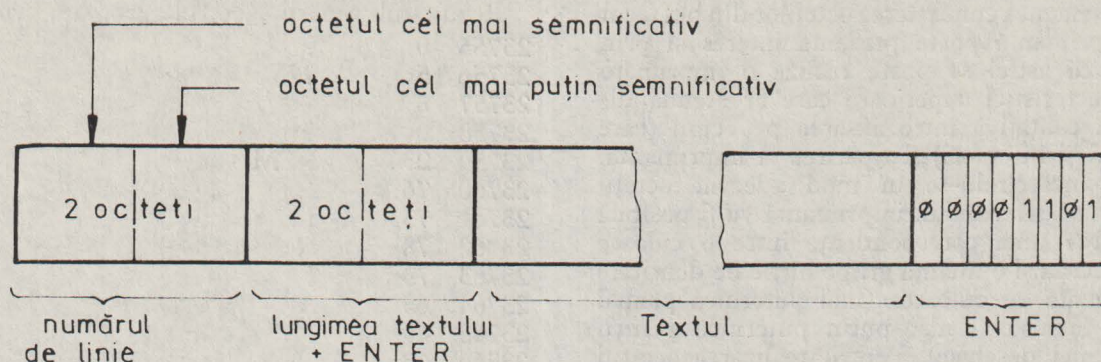


Fig. 2. — Forma liniei de program BASIC

realizează, de exemplu, pe 6 octeți: primul pentru literă, al doilea pentru exponent și următorii patru pentru mantisă. În codificarea liniilor programului Basic valoarea cheie este 13 (codul ASCII pentru caracterul ENTER, vezi octetul 23766) care indică sistemului că octetul următor reprezintă începutul unei noi linii, avînd rolul deci de a separa liniile.

Semnele de întrebare apărute la afișare (de exemplu corespunzător octeților 23755, 23756 etc) se datoresc faptului că pentru codurile ale căror caractere corespunzătoare nu sînt utilizate (codurile 0—5) sau pentru codurile corespunzătoare mișcării cursorului (codurile 8—11) etc apare de obicei afișat un semn de întrebare. Deasemenea a mai apărut cuvîntul cheie THEN pentru octetul 23782 și caracterul/ pentru octetul 23783 corespunzătoare codurilor 203 și respectiv 92 care fac parte însă din reprezentarea constantei numerice 23755. De altfel dacă rezultatele se obțineau la imprimanta (prin modificarea lui PRINT cu LPRINT) nu ar mai fi fost tipărite nici semnele de întrebare și nici celelalte caractere sau cuvinte cheie care nu au legătură cu programul Basic decît prin simpla coincidență a reprezentării unor constante numerice cu codurile unor caractere sau cuvinte cheie.

Vom dezvolta în continuare cîteva aplicații care vor pune în valoare importanța cunoașterii codificării liniilor de program Basic și a structurii memoriei corespunzătoare zonei programului.

Aplicația 1: Renumerotarea liniilor unui program

În sistemul Basic al calculatoarelor studiate este imposibilă renumerotarea liniilor unui program acest lucru fiind totuși foarte util în vederea inserării unor noi linii de program, a operației de fuzionare între două sau mai multe programe care au numere de linie identice sau pur și simplu în vederea organizării estetice a programelor. Problema renumerotării liniilor unui program Basic se poate rezolva însă cu ajutorul unor rutine. Programul P9 intitulat "Renumerotare" poate fi plasat în continuarea unui program în curs de elaborare pentru modificarea numerotării între mai multe linii sau pentru a șterge o porțiune de program care ocupă prea mult loc pentru numerele de

linie (se știe că nici acest lucru nu este realizabil în sistemul Basic decît prin ștergerea linie cu linie ceea ce poate fi destul de anevoios dacă este vorba de un număr mai mare de linii care trebuie șterse).

P9

```

9000 REM RENUMEROTARE
9005 INPUT "linia de inceput ";D
9010 INPUT "pasul ";P
9015 LET Y=23755
9020 POKE Y,INT (D/256)
9030 POKE (Y+1),INT ((D/256-INT
(D/256))*256)
9040 LET D=D+P
9050 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4
9060 IF PEEK (Y)=35 AND PEEK (Y+
1)=40 THEN GO TO 9080
9070 GO TO 9020
9080 LIST

```

Această rutină renumerotează liniile programului situat înaintea ei. Cînd este întîlnită linia 9000 rutina se oprește pentru a nu perturba execuția în curs. Programul se situează pe octetul de la începutul zonei program (linia 9015) și apoi introduce în adresele primilor doi octeți ai zonei program numărul liniei de pornire descompus pentru a fi reprezentat pe doi octeți (liniile 9020 și 9030). Prin adăugarea pasului P la numărul liniei de început (linia 9040) se va găsi numărul următoarei linii de program. Linia 9050 utilizează octeții care indică lungimea liniei în curs de renumerotare pentru a calcula saltul care trebuie făcut pentru poziționarea pe octetul care descrie parametrii următoarei linii de program. Linia 9060 testează dacă renumerotarea a ajuns la numărul de linie 9000 ($9000 = 40 + 256 \times 35$) caz în care procesul încetează. Dacă nu s-a ajuns însă la linia 9000 procedura de memorare a primilor doi octeți ai liniei următoare este apelată din nou prin linia 9070 iar în final listarea întregului program (renumerotat) este cerută de linia 9080.

Programul P10 aduce o facilitate suplimentară față de programul precedent și anume se poate renumerota numai o porțiune de program (situată între o limită inferioară și alta superioară).

P10

```

9000 REM RENUMEROTARE BLOC
9005 INPUT "incepind de la linia ?";N
9006 INPUT "pina la linia?";M
9010 INPUT "pasul? ";P
9015 LET Y=23755
9020 LET A=INT (M/256): LET B=(M
/256-INT (M/256))*256
9030 IF 256*PEEK Y+PEEK (Y+1)=N
THEN GO TO 9100
9050 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4
9060 GO TO 9030
9100 LET D=N
9120 POKE Y,INT (D/256)
9130 POKE (Y+1),INT ((D/256-INT
(D/256))*256)
9140 LET D=D+P
9150 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4
9160 IF PEEK Y=A AND PEEK (Y+1) =
B THEN GO TO 9180

```

9170 GO TO 9120

9180 LIST

Programul se poziționează pe primul octet al zonei program și începînd de aici citește ansamblul de numere de linii pînă cînd întîlnește numărul de linie de la care s-a cerut executarea renumerotării (linia 9030). Următoarea porțiune de program este practic identică cu programul precedent, renumerotarea încheindu-se în momentul în care se ajunge la linia pînă la care s-a dorit renumerotarea.

Procedura de renumerotare, foarte utilă, prezintă totuși un inconvenient: numerele de transfer care însoțesc instrucțiunile de salt ca GO TO sau GO SUB nu sînt modificate. Programul P11 prezintă în plus față de precedentele ștergerea zonelor caracter care urmează după instrucțiunile GO TO și GO SUB ale programului renumerotat, cu scopul de a atrage atenția utilizatorului asupra modificărilor pe care trebuie să le opereze în aceste instrucțiuni prin introducerea noilor numere de linie (vezi fig. 3). Se va observa cu această ocazie și modul în care sînt codificate numerele în memorie.

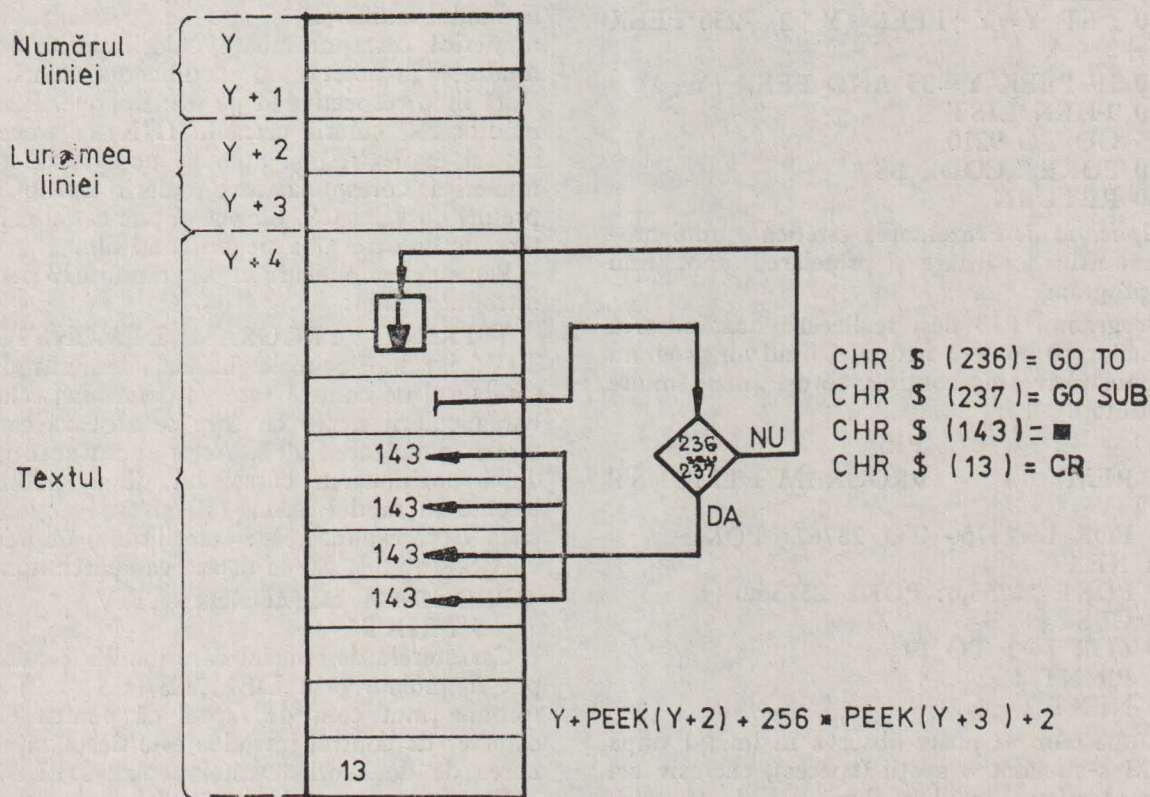


Fig. 3. — Introducerea noilor numere de linie (programul P 11)

P11

```

9200 LET Y=23755
9210 FOR I=Y+4 TO Y+PEEK (Y+2)+2
56*PEEK (Y+3)+2
9220 IF PEEK (I)=236 OR PEEK (I)
=237 THEN GO SUB 9250
9230 LET Y=Y+PEEK (Y+2)+256*PEEK
(Y+3)+4

```

```

9240 IF PEEK Y=35 AND PEEK (Y+1)
=40 THEN LIST
9245 GO TO 9210
9250 POKE (I+1),143: POKE (I+2),
143: POKE (I+3),143: POKE (I+4),
143
9260 RETURN

```


Aplicația 2: Modificarea automată a numelor variabilelor

Dacă un program devine mai lung și deci complex posibilitatea modificării numelor variabilelor în vederea grupării lor (A, B și F în X, Y, Z) devine un lucru util. Structura programului care realizează modificarea numelor variabilelor (P12) este asemănătoare cu cea a programului de renumerotare. În momentul în care se întâlnește codul literei (linia 9220) acesta este substituit prin intermediul rutinei 9250 cu codul noii litere. Se crează impresia că programul se scrie singur.

P12

```
9200 REM MODIFICAREA NUMELUI VAR  
IABILELOR  
9204 INPUT "litera de modificat? ";A$  
9206 INPUT "cu care litera?";B$  
9208 LET Y=23755  
9210 FOR I=Y+4 TO Y+PEEK(Y+2)+2  
56*PEEK(Y+3)+2  
9220 IF PEEK(I)=CODE A$ THEN GO  
SUB 9250  
9225 NEXT I  
9230 LET Y=Y+PEEK(Y+2)+256*PEEK  
(Y+3)+4  
9240 IF PEEK Y=35 AND PEEK(Y+1)  
=40 THEN LIST  
9245 GO TO 9210  
9250 POKE I, CODE B$  
9260 RETURN
```

Aplicația 3: Prezentarea estetică a unui program: titlu la listare și protejarea unor linii de program

Programul P13 deși realizează doar afișarea primelor 10 numere naturale, fiind un program demonstrativ, el conține totuși mai multe subtilități.

P13

```
1 REM PROGRAM DEMONSTRATIV  
10 FOR I=23760 TO 23767: POKE  
I,8: NEXT I  
20 POKE 23755,0: POKE 23756,0  
100 CLS  
110 FOR I=1 TO 10  
120 PRINT I  
130 NEXT I
```

După cum se poate observa în linia 1 după REM s-au lăsat 8 spații (spaces), exclusiv cel natural plasat imediat după REM ce către sistem, începîndu-se comentariul de la poziția 9. În linia 10 s-a introdus în primii 8 (cite spații s-au lăsat în REM) octeți ai textului primei linii de program caracterul de control codificat cu 8 (el corespunde caracterului "cursor înapoi" — back space) astfel încît la listarea programului (după rularea sa) linia 1 nu va fi afișată normal, ci ca un titlu de program (fără număr de linie). Dacă s-ar fi lăsat mai multe spații libere după REM, s-ar fi obținut și o deplasare corespunzătoare a titlului spre

dreapta, adică o încercare de "centrare" a titlului. În linia 20 s-a introdus în locațiile de memorie 23755 și 23756 (în care se reprezintă numărul liniei respective) valoarea 0 astfel încît după rularea programului linia 1 se va transforma în linia 0. În plus acesta linie nu va mai putea fi nici editată și nici ștearsă. Procedeu general pentru a readuce linia 0 la numărul de linie inițial NNNN (care poate fi un număr mai mic ca 10000) este următorul:

POKE 23755, INT (NNNN/256): POKE 23756, NNNN-256*INT (NNNN/256)
Bineînțeles că liniile 10 și 20 le putem introduce și în mod comandă, în acest caz nemaifiind necesară rularea programului pentru protejarea primei linii și pentru obținerea unei prezentări estetice.

Uneori o protejare eficientă a listării liniilor de program poate fi și numai facila utilizare a caracterelor de control în cadrul unei instrucțiuni PRINT prin modificarea culorii cernelii (INK) sau a fondului (PAPER) astfel încît culoarea cernelii să fie aceeași cu culoarea fondului. Amintim cu această ocazie că pentru modificarea culorii fondului (PAPER) se trece în modul extins (E) și apoi se acționează tasta numerică corespunzătoare culorii dorite obținîndu-se în acest mod modificarea culorii pe toată linia respectivă și pe următoarele. Pentru modificarea culorii cernelii (INK) se trece în modul extins (E) și apoi se acționează tasta numerică corespunzătoare culorii dorite împreună cu tasta CS. Se pot obține astfel caractere de culoare albă pe un fond alb.

Pentru exemplificare în programul P13 putem introduce linia:

```
5 PRINT "PROGRAM DEMONSTRATIV", iar după primele ghilimele vom introduce caracterul de control care va determina schimbarea culorii negre cu care se afișează caracterele în culoarea albă. Acest lucru se obține după poziționarea cursorului după ghilimele trecînd în modul extins (E) și apoi acționînd tasta 7 (corespunzătoare culorii albe) împreună cu CS. În acest caz la listare va apărea numai:
```

```
PROGRAM DEMONSTRATIV  
5 PRINT "
```

Caracterele de control care modifică culorile pot fi anulate prin DELETE (CS și 0) dar trebuie ținut cont de faptul că pentru orice caracter de control introdus este necesară acționarea de două ori a tastei pentru DELETE.

Folosind caracterele de control pentru deplasarea cursorului se poate realiza și modificarea numelui unui program salvat (heder). Se știe ca numele unui program Basic este o expresie șir de caractere care se referă la numele sub care a fost salvat programul pe casetă magnetică. Această expresie constă în maximum 10 caractere ASCII. De asemenea se știe că atunci cînd la o operație de încărcare a unui program Basic se identifică un anume program apare pe ecran mesajul:

Program: <nume program>

De exemplu dacă vom introduce comanda de salvare pentru un program Basic:

SAVE CHR\$ 8+CHR\$ 8+"el: hop"
atunci la încercarea de a încărca alt program sau chiar acesta va apărea mesajul:

Programel: hop
deoarece cele două caractere de control introduse în numele programului va determina deplasarea cursorului de două ori înspre stînga: o dată trecînd peste spațiul dintre cele două puncte și numele programului și a doua oară peste cele două puncte introducînd din această poziție restul denumirii programului. De altfel dacă se va introduce o linie, să zicem,

1000 PRINT AT 10,10;"Program: " +
CHR\$ 8+CHR\$ 8+"el: hop"

și apoi comandăm GO TO 1000 vom vedea cum cele două caractere de după m și anume e și l pîlpie, fiind adăugate permanent.

2.4. Organizarea memoriei ROM

Memoria ROM este situată în primii 16 Kocteti ai memoriei fiind constituită în principal din sistemul de operare, interpretorul Basic și o tabelă de 96 de caractere. În această memorie rezidă astfel un program foarte mare în limbaj mașina—programul monitor—al cărui studiu este important deoarece cunoscîndu-se amplasarea unor rutine utile este suficient să se învețe tehnica de apelare a acestora din programele Basic ale utilizatorilor pentru a se dispune în acest mod de programe foarte eficiente.

2.4.1. Sistemul de operare

Această parte se compune din rutine de inițializare care ocupa octeții de la 0 la 7 și de la 4555 la 4769. Rutinele de inițializare au ca scop cunoașterea de către sistem a volumului memoriei disponibile pentru executarea operațiilor următoare. Rutinele de afișare de mare utilitate pentru utilizatori sînt conținute între adresele 2548 și 3404. Alte rutine importante sînt rutinele de editare, adică tot ce reprezintă funcție de editare și verificare de sintaxă. Știm că dacă o linie de program a fost introdusă greșit, atunci se va da un mesaj (raport) de eroare corespunzător. Tocmai această parte a memoriei ROM care se întinde de la adresa 3884 la 5550, conținînd deci o parte din rutinele de inițializare, asigură funcționarea respectivă. O subrutina interesantă a acestui ansamblu este rutina de decodificare a tastaturii care citește ultima tastă care a fost acționată. Ea se situează de la 4264 la 4380. Cu ajutorul ei se realizează, bineînțeles, funcția INKEY\$.

2.4.2. Interpretorul Basic

Această parte esențială a memoriei ROM are ca funcție traducerea liniilor Basic în rutine limbaj mașina care pot fi înțelese de microprocesor. Cele mai importante părți ale inter-

pretatorului Basic sînt rutinele de control care fac ca interpretarea să treacă de la o linie la alta; ele sînt situate între 6935 și 7168. De analiza parametrilor care acompaniază funcțiile și comenzile Basic se ocupă o rutină situată între 7176 și 7389.

Pînă la 9466 rutinele corespund comenzilor Basic. Astfel la fiecare din cele circa 50 de comenzi Basic de care dispun calculatoarele studiate, corespunde un subprogram în limbaj mașină. Acesta funcționează în legătură cu tabela de comenzi situată între 6728 și 6934.

Pașii 9467 la 10955 sînt ocupați de evaluatorul expresiilor și rutinele de manipulare a variabilelor. Urmează rutinele de calcul, plasate între 12187 și 14445. Aici sînt efectuate operațiile aritmetice, calculele funcțiilor trigonometrice etc.

2.5. Utilizarea rutinelor programului monitor în programe Basic

În acest capitol vom exemplifica cum anumite rutine eficiente ale monitorului pot fi recuperate și utilizate cu ușurință în programe Basic.

Aplicația 1: ștergerea unui bloc de linii afișate pe ecran.

La adresa 3652 începe o rutină care șterge un anumit număr de linii de pe ecran, începînd cu partea de jos a ecranului. Programul P14 conține la linia 100, date reprezentînd 6 valori zecimale care vor fi mai întîi citite și apoi introduse în adresele de memorie începînd cu 65368 (memoria rezervată UDG-urilor).

P14

```
5 REM APEL RUTINA MEMORIE
10 FOR I=1 TO 20
20 PRINT "*****"
*****"
30 NEXT I
40 INPUT "numar linii de sters? "; L
50 POKE 65369,L
60 LET K=USR 65368: STOP
100 DATA 6,10,205,68,14,201
110 FOR I=0 TO 5
120 READ A: POKE (65368+I),A
130 NEXT I
```

Primul număr reprezintă codul instrucțiunii de încărcare a unui număr într-un registru al microprocesorului (în limbaj mașină aceasta se scrie LD B,N ceea ce înseamnă că se încarcă registrul B cu valoarea N). Al doilea număr reprezintă chiar numărul de încărcat în registru (deci N=10). Codul 205 (al treilea număr citit) reprezintă codificarea în limbaj mașină a instrucțiunii de apel a unei rutine (în limbaj se scrie CALL NN unde NN este adresa de memorie a rutinei care se apelează). După codul 205 urmează adresa rutinei pentru ștergerea liniilor de pe ecran, codificată pe doi octeți, fiind un număr mai mare decît 256 (codificarea se face deci pe doi octeți dintre care cel mai semnificativ este primul din dreapta).

Valorile se pot verifica prin operația: $14 \times 256 + 68 = 3652$. Ultima dată (201) reprezintă codul instrucțiunii de reîntoarcere în Basic (RET).

Punerea în lucru a acestei rutine se efectuează prin GO TO 100 înainte de rularea efectivă, deoarece datele trebuie mai întâi citite și introduse în zona UDG-urilor și apoi apelată rutina de la adresa 3652 prin linia de program 60.

Primele linii de program realizează umplerea a 20 de linii ecran cu caractere asterisc. Apoi se întreabă câte linii de ecran se dorește a se șterge, iar numărul cu care se răspunde se va încărca în adresa 65369, adică în al doilea octet al rutinei în cod mașina. Ștergerea numărului de linii afișate pe ecran începând din partea de jos a ecranului se va realiza imediat. Cu această procedură inserată într-un program Basic se va putea realiza menținerea unor titluri explicative în partea de sus a ecranului cu ștergerea unui text. Fără această procedură acest lucru se putea realiza mai puțin eficient, în partea de jos a ecranului pe liniile 23 și 24.

Aplicația 2: Vizualizarea locului memoriei disponibile

Pentru utilizator este important să cunoască în orice moment al realizării unor programe care sînt părțile principale de memorie ocupată mai ales dacă programele sînt lungi și în ele intervin multe variabile. Astfel se poate identifica locul de memorie care rămîne încă disponibil dar, în plus, se poate realiza un echilibru mai bun între lungimea programului și numărul de variabile care intervin. Aceasta deoarece în mod normal aceeași aplicație poate fi realizată prin intermediul unui program mai lung care conține mai puține variabile sau unui program mai scurt care conține însă mai multe variabile. Deseori codificarea unei variabile numerice într-o variabilă alfanumerică în tablouri mari permite un important cîștig de spațiu de memorie cu prețul cîtorva linii suplimentare de instrucțiuni. În schimb, invers, utilizarea cîtorva variabile de lucru intermediare permite cîteodată reducerea pașilor de program care se repetă deseori și a căror scriere nu este simplă.

Programul P15, ușor de incorporat ca subrutină în alt program oferă informații utile și prezintă avantajul unui acces permanent, pentru ca odată testată amplasarea zonelor de memorie mai întinse, rezultatul este transcris în interiorul liniei REM sub o formă ușor vizualizabilă.

P15

```
1 REM MARCAREA SPATIULUI
MEMORIEI PRIN RUTINA UTILIZATOR.
9000 FOR I=23760 TO 65367 STEP 770
9010 IF I<(PEEK 23627+256*PEEK
23628) THEN LET A=80: GO TO 9050
9020 IF I<(PEEK 23641+256*PEEK
23642) THEN LET A=86: GO TO 9050
9030 IF I<(PEEK 23653+256*PEEK
23654) THEN LET A=76: GO TO 9050
9040 LET A=32
```

9050 POKE (23760+(I-23760)/770), A:
NEXT I: LIST

Dacă rutina 9000 este apelată după un program plasat între liniile 10 și 90000, cele 53 de caractere care urmează liniei REM vor fi modificate reprezentînd proporțional ansamblul memoriei cuprinse între 23760, zona de început a programului și 65368, zona rezervată UDG-urilor. Zona ocupată proporțional de program va fi umplută cu caractere P, cea ocupată de variabile va fi umplută cu caractere V, cea ocupată de zona de lucru a calculatorului va fi umplută cu caractere L iar în zonă disponibilă integral vor figura spații goale (blancuri). Zona de lucru a calculatorului este cuprinsă între începutul zonei de editare și sfîrșitul stivei * calculatorului. Reprezentarea nu este foarte precisă deoarece fiecare caracter corespunde unei zone de 770 octeți, ceea ce are ca urmare nereprezentarea uneori a zonei de lucru, aceasta conținînd prea puțini octeți. Au fost necesare 55 de caractere conform calculului:

INTREG [(65367-23760)/770]=53

Limitele testate succesiv sînt determinate de conținutul variabilelor sistem VARS și RAMTOP.

Programul P16, mai lung în execuție utilizează ecranul și culorile în scopul unei reprezentări mai precise a repartizării memoriei. Cele 704 zone caracter corespunzătoare primelor 22 linii ecran sînt modificate astfel: pentru octeții ocupați de program zonele caracter vor apărea cu albastru închis, pentru octeții ocupați de zona de variabile cu albastru deschis, cu roșu pentru zona de lucru și cu verde pentru zona rămasă la dispoziție.

P16

```
1 REM MARCAREA SPATIULUI MEMO
9000 FOR I=23760 TO 65367 STEP 12.56
9010 IF I<(PEEK 23627+256*PEEK
23628) THEN LET A=80: LET P=1: GO
TO 9050
9020 IF I<(PEEK 23641+256*PEEK
23642) THEN LET A=86: LET P=5: GO
TO 9050
9030 IF I<(PEEK 23653+256*PEEK
23654) THEN LET A=76: LET P=2: GO
TO 9050
9040 LET A=32: LET P=4
9050 PRINT PAPER P; INK 9; CHR$ A;;
NEXT I
```

De fapt zona cu adevărat disponibilă, care începe la sfîrșitul stivei calculatorului nu se întinde pînă la cel mai sus punct al memoriei. Trebuie scăzut un oarecare număr de octeți care reprezintă stiva calculatorului precum și stiva adreselor de întoarcere a transferului instrucțiunilor GO SUB (3 octeți pentru un transfer). Din păcate nu există în locul variabilelor sistem o

* Notă: putem imagina stiva ca un tablou la care nu avem acces decît la ultimul element introdus. Există un indicator (pointer) care ne spune unde ne aflăm la un moment dat. Extragerea elementelor din stivă se face în ordinea inversă introducerii elementelor în stivă.

variabilă reactualizată constant care să indice începutul stivei microprocesorului. Pentru aceasta este nevoie de un program în cod mașina.

2.6. Variabile de sistem

Octeții de memorie de la adresa 23552 la adresa 23733 sînt rezervați pentru operații specifice ale sistemului. Ei pot fi citați pentru a afla diferite lucruri despre sistem, iar unii dintre ei pot fi și modificați. Acești octeți se numesc variabile de sistem și au câte un nume (cu toate acestea nu trebuie confundați cu variabilele utilizate de BASIC). Unele variabile de sistem sînt localizate într-un singur octet de memorie, altele pe doi octeți de memorie și mai puține pe 3 sau chiar mai mulți octeți. Fiind vorba de adrese de memorie, în cazul variabilelor formate din mai mulți octeți, primul va fi octetul cel mai puțin semnificativ.

Lucrul cu variabilele de sistem este poate cel mai spectaculos legat de utilizarea memoriei, prin modificarea unor variabile de sistem putîndu-se realiza artificii interesante și eficiente cu un efort minim. Dar atenție! modificarea unor variabile de sistem va conduce la o funcționare eronată a sistemului, de aceea acest lucru nu este în general recomandat. De obicei în tabelele în care se regăsesc aceste variabile se indică asupra cărora nu se poate acționa precum și cele a căror modificare nu are un efect asupra funcționării normale a sistemului. Deasemenea se specifică și numărul de octeți din variabilă. (Vezi, de exemplu HC 85 Manual tehnic, ICE). Vom da câteva exemple de utilizare a variabilelor de sistem prin care se obțin lucruri interesante prin simpla modificare a conținutului acestor variabile, dar exemplele pot fi, bineînțeles, mult mai numeroase.

Aplicația 1: Protecția programelor la întreruperi utilizînd variabilele de sistem.

De obicei pentru astfel de protecții se folosește variabila de sistem ERR SP, aceasta specificînd adresa de întoarcere în caz de eroare. Este o variabilă care indicînd o adresă, va avea nevoie de doi octeți, aceștia fiind 23613 și 23614. Tocmai pentru că modificarea ei are ca repercusiuni funcționarea anormală a sistemului este utilizată la protecția programelor la întreruperi. Programul P17 desenează puncte pe ecran la distanțe foarte mici (pasul 0,1) pe o linie diagonală. Datorită modificării lui ERR SP din linia 1 sistemul se va bloca la BREAK în timpul rulării programului.

P17

```
1 POKE 23613,0: POKE 23614,0
10 LET A=1
20 PLOT A,A
30 LET A=A+1
40 GO TO 20
```

La alte valori ale variabilei, de exemplu pentru:

```
1 POKE 23614,244
```

programul se va distruge la o încercare de întrerupere (BREAK). Aceleași efecte s-ar produce și la o încercare de oprire a programului cu STOP (dacă în program ar exista o instrucțiune INPUT) sau la o eroare (de exemplu așteptînd ca programul să se termine natural, caz în care A va ajunge la o valoare care va depăși limitele ecranului).

Tot în scopul protejării programelor la întreruperi se poate utiliza variabila de sistem E LINE care conține adresa comenzii introduse. Aceasta este o variabilă de sistem pe doi octeți (adresa 23641/23642) care dacă ia anumite valori poate bloca tastatura la comenzi sau distruge programul, așa cum se va întîmpla în cazul introducerii la rularea programului P18 a unei valori pentru A mai mare decît 100.

P18

```
10 INPUT A
20 IF A > 100 THEN POKE 23642,0
30 PLOT A,A
40 GO TO 10
```

Dacă vom introduce o valoare pentru A mai mare decît 100 și se oprește programul cu STOP sau BREAK, la listare programul se autodistruge.

Altă variabilă de sistem care poate fi folosită în protejarea programelor la întreruperi este cea care conține numărul liniilor din partea de jos a ecranului (DF SZ). Este o variabilă care se găsește la adresa 23659 fiind pe un singur octet. În programul P19 deoarece s-au declarat zero linii în partea de jos a ecranului (linia 10) la o întrerupere a programului din rulare prin BREAK sistemul nu va mai avea unde să scrie mesajul de întrerupere și programul se va distruge.

P19

```
10 POKE 23659,0
20 LET I=1
30 PRINT AT 1,1:I
40 LET I=I+1
50 GO TO 30
```

Procedul se poate utiliza pentru programe scurte (de exemplu la programe de încărcare) nefuncționînd dacă există instrucțiuni INPUT în program.

Aplicația 2: Protecția programelor la încărcare.

Această protecție este făcută în ideea ca programele să nu poată fi citite (încărcate) de alți utilizatori decît realizatorul programului respectiv și se realizează prin modificarea variabilei de sistem PROG care indică adresa programului BASIC. Adresa variabilei PROG este 23635/23636.

Știm că adresa programului BASIC este 23755; acesta fiind un număr care ocupă doi octeți ($23755 = 203 + 256 \cdot 92$) în octetul cel mai puțin semnificativ vom găsi valoarea 203.

Pentru salvarea unei versiuni care va deveni protejată la citire se va proceda astfel:
POKE 23635,n: SAVE "<nume>"
unde n poate fi orice număr natural mai mic ca 203.

Pentru încărcarea programului respectiv realizatorul său va introduce:

POKE 23635,n: LOAD " "
neuitînd că după încărcare să restabilească valoarea variabilei PROG prin:

POKE 23635,203

Dacă nu se va ști valoarea lui n în momentul salvării programului, acesta nu se va putea încărca.

Aplicația 3: Alte artificii realizate prin utilizarea variabilelor de sistem.

Variabila de sistem PIP (pe un singur octet, adresa 23609) conține durata sunetului la apăsarea unei taste. Folosindu-se în programe, de exemplu, POKE 23609,5 sau POKE 23609,10 se va obține un sunet caracteristic (clic) la apăsarea unei taste. Cu cît numărul introdus în adresa 23609 va fi mai mic cu atît sunetul va fi mai scurt. Pentru 255 se obține sunetul cu cea mai lungă durată.

Variabila de sistem MODE (pe un singur octet, adresa 23617) specifică cursorul (K,L,C,E,G). Modificînd această variabilă se pot obține diverse caractere pentru cursor, inclusiv un caracter grafic.

Variabila de sistem FRAMES (este pe trei octeți) se găsește la adresa 23672/23673/23674 și indică contorul de cadre. Cu ajutorul acestei variabile de sistem se poate obține o m surare mai precisă a timpului cu ajutorul calculu orului

față de instrucțiunea PAUSE cu care se poate obține doar o pauză aproximativă a calculatorului. De exemplu PAUSE 42 marchează trecerea aproximativ a unei secunde. Citind valorile din locațiile variabilei de sistem FRAMES și anume 23674, 23673 și 23672 se pot număra incrementele mai precise de 20 ms. Fiecare locație variază de la 0 la 255, după care se reîncepe. Cel mai rapid se incrementează locația 23672 (cu 1 la fiecare 20 ms); cînd se trece de la 255 la 0, locația 23673 se incrementează cu 1; analog pentru 23674. De exemplu pentru a se poziționa ceasul pe ora 10 se procedează astfel:

POKE 23674,27: POKE 23673,119: POKE 23672,64

conform calculului: $10 \cdot 60 \cdot 60 \cdot 50 = 1800000 = 65536 \cdot 27 + 256 \cdot 19 + 64$

Variabila de sistem SCR CT (un octet la adresa 23692) numără scroll-urile. Valoarea ei este totdeauna mai mare cu o unitate decît numărul de scroll-uri care vor fi făcute înaintea opririi prin mesajul *scroll*?. În acest caz dacă se face POKE 23692,255 se va obține un scroll permanent fără a mai apare mesajul.

Variabila de sistem REPPER (un octet la adresa 23562) determină timpul necesar acționării unei taste astfel încît aceasta să se repete. Inițial valoarea variabilei REPPER este 35, dar dacă ea se modifică, de exemplu cu POKE 23562,1 atunci tastatura va fi citită mult mai repede, ceea ce determină o editare mult mai eficientă a liniilor de program mai lungi care necesită modificări (cursorul se deplasează mult mai rapid).

CAPITOLUL 3

BETA BASIC — EXTENSIE A LIMBAJULUI BASIC

Beta Basic este o puternică extensie a interpretorului de Basic Spectrum care oferă utilizatorului posibilități largi de lucru cu un limbaj de nivel înalt, capabil să utilizeze optim posibilitățile mașinii.

Extensia limbajului este însoțită de un editor mult mai evoluat care permite o adevărată editare ecran.

Accesul la comenzile suplimentare se face trecând tastatura în modul G (Graphics), apăsând CAPS SHIFT și 9 simultan, după care se apasă tasta corespunzătoare cuvântului cheie din setul extins. Din motive mnemotehnice acestea sînt tastele literale care reprezintă prima literă din numele comenzii.

Beta Basic extinde limbajul de programare Basic cu:

- 1) instrucțiuni pentru programarea structurată
 - . decizie IF THEN ... ELSE
 - . cicluri DO LOOP
 - . cu pretest (WHILE)
 - . cu posttest (UNTIL)
 - . cicluri cu mai multe ieșiri (EXITIF)
- 2) salturi și apeluri de subrutine calculate cu ON... GO TO... și ON... GOSUB...
- 3) subrutinele pot fi apelate după *nume* și cu o listă de *parametri formali*, care poate să nu fie complet specificată, caz în care se pot specifica valori prin lipsă pentru parametri formali.
- 4) la fel ca în limbajul de nivel înalt Pascal și C în Beta Basic apare conceptul de "încapsulare". Se pot declara în subrutine variabile locale, care nu sînt vizibile în afara acelor subrutine. Aceste variabile sînt alocate dinamic astfel încît mecanismul de apeluri recursive este mult îmbunătățit. Acesta este completat de instrucțiunea POP care produce o revenire necondiționată dintr-o variabilă.
- 5) pentru a putea apela mai ușor asupra memoriei sînt disponibile funcții care manipulează datele în formatul adresă Intel, cu octeți, care tratează zone din memorie la fel ca niște șiruri de caractere.
- 6) Beta Basic oferă posibilități mai largi de manipulare a șirurilor de caractere.
- 7) funcțiile aritmetice suplimentare permit operații la nivel de bit, funcțiile SINE și COSE, care operează mult mai rapid, de cca. 6 ori, dar cu o precizie mai mică.
- 8) sînt prezente funcții de conversie și editare a datelor.
- 9) funcțiile grafice oferă poate cea mai spectaculoasă implementare. Se poate defini pe ecran o fereastră în numere reale, se poate

deplasa originea axelor de coordonate, se pot face deplasări de zone ecran, schimba atributele globale.

10) instrucțiuni pentru sortarea șirurilor și tablourilor numerice și de caractere.

De mare folos în dezvoltarea programelor sînt comenzile de editare suplimentare ce permit:

- 1) renumerotarea liniilor
- 2) ștergerea pe blocuri a liniilor
- 3) listarea liniilor
- 4) autonumerotarea liniilor la introducere
- 5) condensarea unor linii
- 6) spargerea unei linii în mai multe

Beta Basic are posibilități de tratare de către utilizator a tuturor erorilor produse de interpretorul Basic și Interfata 1, cu instrucțiunea ON ERROR.

Trasarea complexă a execuției programelor se face cu instrucțiunea TRACE.

Mai sînt posibile:

- 1) modificarea dimensiunilor caracterelor
- 2) indexarea automată
- 3) definirea de taste programate
- 4) utilizarea ceasului de timp real.

UTILIZARE

Pentru introducerea comenzilor se trece în modul Graphics (cs +9) și se apasă tasta corespunzătoare.

Comenzi:

1) ALTER < sir de atribute 1 > TO < sir de atribute 2 > (Gr. A)

Schimbă atributele ecran pentru toate caracterele ce au combinația < sir de atribute 1 > în combinația < sir de atribute 2 >.

Ex:

ALTER INK 7 TO INK 0

ALTER INK 3, BRIGHT 1, PAPER 7
TO INK 5, FLASH 1

2) AUTO < nr. linie > < ,valoare pas > (Gr.6)

3) BREAK se întrerupe programul revenindu-se în sistemul BASIC, dacă se apasă mai mult de 1 secundă CAPS SHIFT + Break. Este folositor dacă au fost dezarmate "STOP IN INPUT" sau "BREAK INTO PROGRAM" cu o instrucțiune ON ERROR.

4) CLOCK < nr > sau < sir > (Gr. C)

Această instrucțiune controlează lucrul unui driver de întreruperi cu ceas de timp real de 24 de ore, care poate afișa ora curentă în colțul din dreapta sus al ecranului. Cînd se ajunge la o oră fixată, o alarmă sonoră poate fi declanșată și/sau o anumită subrutină poate fi lansată în execuție (apel GOSUB). Pentru a

menționa care din facilități este operațională, CLOCK trebuie urmat de o expresie numerică cu valoarea între 0 și 7, care va poziționa ceasul în modul:

Mod	Rutina de tratare a întreruperii	Alarma sonora	Afișaj
0	nu	nu	nu
1	nu	nu	da
2	nu	da	nu
3	nu	da	da
4	da	nu	nu
5	da	nu	da
6	da	da	nu
7	da	da	da

Indicarea liniei rutinei de tratare a întreruperii de ceas se face cu CLOCK nr1, cu nr1 > 8.

Resetarea ceasului se face cu CLOCK urmat de un șir de caractere urmat de primele 3 cifre vor reprezenta ora. Orice alt caracter înafara lui ("A" "a") este ignorat.

Dacă șirul este de forma "A....." atunci alarma va fi poziționată la ora de după A.

Apelul la rutina de tratare a întreruperii de ceas are loc doar dacă în acel moment rulează un program.

5) Taster pentru controlul cursorului

CHR\$ 8 ← (CS+5) identic PRINT
CHR\$ 9 → (CS+8) CHR\$ 8 in Basic
CHR\$ 10 ↑ (CS+7)
CHR\$ 11 ↓ (CS+6)

6) DELETE <nr1> TO <nr2> (Gr. 7)

Șterge textul între limitele specificate.

DELETE 0 șterge întreg programul mai puțin linia zero. Diferă de NEW prin aceea că nu șterge și variabilele programului.

7) DO (Gr. D) WHILE (Gr. J)

DO WHILE condiție UNTIL (Gr. K)

DO UNTIL condiție LOOP (Gr. L)

DO și LOOP împreună cu calificatorii WHILE și UNTIL formează o structură de control.

WHILE: zona de program dintre DO și LOOP este executată atâta timp cât condiția este adevărată.

UNTIL: zona de program dintre DO și LOOP se execută pînă cînd condiția devine adevărată (deci atîta timp cît este falsă).

8) DPOKE adr.,valoare (Gr. P)

valoare = val. pe 2 octeti

Operează cu informații memorate pe 2 octeti în format de adresă.

9) EDIT (Gr. 5)

10) ELSE (Gr. E) structura este:

IF cond. THEN instr: instr.....:

:ELSE instr.....: instr:... <CR>

11) ENDPROC (Gr. 3)

marchează sfîrșitul unei proceduri.

12) EXIT IF (<conditie>) (Gr. I)

este o parte din structura DO-LOOP.

Permite ieșirea din ciclu de undeva din mijlocul unei bucle sau și nu pe la capete,

Dacă condiția e adevărată se face un salt la instrucțiunea imediat următoare lui LOOP, altfel nu se întîmplă nimic.

13) FILL X,Y (Gr. F)

FILL <INK c>; X,Y

FILL <PAPER c>; X,Y

Umple o zonă de culoare PAPER cu culoarea cernelii INK c, dacă se folosește FILL sau FILL INK sau o zonă de culoarea cernelii cu culoarea hîrtiei PAPER, dacă FILL PAPER este folosit.

14) GET (var. nume) sau (var. șir) (Gr. G)

Lucrează la fel ca INKEY\$, permițînd citirea tastaturii fără a folosi Enter.

Diferența față de INKEY\$ constă în aceea că GET așteaptă ca o tasta să fie apăsată înainte de a continua.

Dacă GET este folosit cu o variabilă numerică se va obține:

1 pentru "1" 10 pentru "A"

2 pentru "2" 11 pentru "B"

3 pentru "3" 214 pentru "CR"

4 pentru "4" 201 pentru "SPACE"

literele apăsate cu SS obțin numere de sute ex:

SS+A → 139

SS cu CS singure nu acționează

15) JOIN <nr1> (Gr. Shift 6)

Adaugă linia următoare lui nr1 la linia specifică nr1. Linia următoare lui nr1 își va pierde numărul de linie și va fi separată de prima printr-un caracter ":".

Ex:

10 PRINT "text1"

20 GET a

30 PRINT a

dacă facem JOIN 10 CR

și apoi introducem 20 PRINT "text2"

la list va apărea:

10 PRINT "text1"

GET a

PRINT a

20 PRINT "text2"

16) KEYIN <șir> (Gr. Shift 1)

Introduce șirul "șir" așa cum ar fi fost introdus direct de la tastatura. Aceasta permite programelor de a se autoscrie. Este posibilă automatizarea producerii unor programe sau părți de programe.

Cu KEYIN se pot introduce doar linii cu număr de linii mai mare decît oricare existent deja!

17) KEYWORDS {0} (Gr. 8)

{1}

Indică care set de caractere se va folosi pentru UDG, cele ale Beta Basic KEYWORDS1 sau cele utilizator (KEYWORDS0)

18) LIST <nr1> TO <nr1>

19) LOOP (Gr.L)

20) WHILE (Gr.J)

21) UNTIL (Gr.K)

22) ON (Gr.O)

se folosește în salturi și apeluri calculate ON permite selectarea unui număr de linie

din cele din listă, conform cu valoarea variabilei sau expresiei numerice ce urmează după ON.

23) ON ERROR <nr1> (Gr.N)

După execuția acestei instrucțiuni se va face un salt la procedura de la linia nr1. dacă apare o eroare. Această facilitate se inhibă cu ON ERROR0 (mai este inhibată pe timpul procedurii utilizator de tratare a erorii) și este reamorsată atunci când se execută RETURN în programul principal.

Rutina de eroare are acces la trei variabile rezervate, care nu sînt cuvinte cheie.

LINE nr1. cu eroare

STAT nr. prop. (instrucțiunii) cu eroare

ERROR codul erorii

Cod semnificație

1 NEXT without FOR

2 variable not found

3 subscript wrong

4 out of memory

5 out of screen

6 number too big

7 RETURN without GOSUB

8 end of file

9 STOP statement

10 invalid argument

11 integer out of range

12 nonsense in Basic

13 BREAK CONT repeat

14 out of Data

15 invalid file name

16 G no room for line

17 STOP in INPUT

18 FOR without NEXT

19 invalid I/O device

20 invalid colour

21 Break into program

22 RAMTOP no good

23 statement lost

24 invalid stream

25 FN without DEF

26 parameter error

27 tape loading error

28 missing LOOP

29 LOOP without DO

30 no such line

31 no POP data

32 missing DEF PROC

33 no END PROC

34 to hanrd?

24) PLOT X,Y <;string>

desenează șirurile la fel ca și pixelii.

Este folosită la etichetarea graficelor și diagramelor

25) POP <valoarea numerică> (Gr.Q)

Extrage o adresă din stiva GOSUB, DO LOOP și PROC.

POP utilizat singur face doar saltul la valoarea din stivă.

POP <locatie> face disponibilă valoarea din vârful stivei în variabila <locatie>.

Aceasta instrucțiune permite, dacă e posibil, saltul înapoi dintr-o subrutină sau un ciclu

DO-LOOP fără a încărca stiva cu valori neutilizate.

26) RENUM (început TO sfîrșit) LINE pasnou STEPLINE (Gr. 4)

Renumerotează liniile programului începînd cu linia "început" pînă la linia "sfîrșit" cu pasul "pasnou".

27) ROLL cod, pixeli <;X,Y; lățime, lungime> (Gr. R)

ROLL mută circular întreg ecranul sau doar o porțiune din ecran în sus, în jos, stînga sau dreapta.

a) codul dă direcția deplasării zonei de pe ecran:

Cod	Dir	Deplasează
1	←	atributele
2	↑	atributele
3	↓	atributele
4	→	atributele
5	←	datele
6	↑	datele
7	↓	datele
8	→	datele
9	←	atributele și datele
10	↑	atributele și datele
11	↓	atributele și datele
12	→	atributele și datele

b) indică numărul de pixeli cu care se face deplasarea atributelor, datelor sau a amîndurora.

c) această secțiune apare doar dacă o anumită fereastră este deplasată circular.

X,Y sînt coordonatele colțului din stînga sus lățime, lungime sînt dimensiunile în caractere și nu în pixeli.

28) CROLL, cod, pixeli [;X,Y;lungime, lățime] (Gr. S)

La fel ca ROLL dar deplasarea nu se face circular; ce se deplasează în afara ecranului se distruge.

29) SORT (Gr. M)

SORT INVERSE

Sortează tablouri de șiruri, numere și litere în ordine crescătoare sau descrescătoare. Utilizarea cu tablouri de șiruri va fi conformă cu:

SORT A\$ (1 TO 20)

A\$ variabila șir — sînt sortate primele 20 de șiruri din tablou crescător.

SORT A\$ () (2 TO) sortează șirurile din tablou conform cu caracterele de la al doilea încolo.

30) Split (Gr. SS W)

10 PRINT "salut" : GO TO 10: <> PRINT "la revedere"

se introduce în textul programului și

10 PRINT "la revedere" va rămîne în partea de jos a ecranului.

31) TRACE nr1. (Gr.T)

Subrutina nr1. este apelată imediat înaintea execuției oricărei instrucțiuni din program.

Sînt disponibile variabilele STAT și LINE

Ex. de rutina de trasare

Trace 9000

9000 LET col=PEEK 23688

LET rind=PEEK 23689

9010 PRINT AT 0,0;LINE;" ";STAT

9020 POKE 23688,col:
POKE 23689,rind
RETURN

Alte variabile rezervate:

XOS, YOS, ORG, YRG

acești 4 identificatori nu sînt cuvinte cheie, ci niște variabile rezervate care permit să fie schimbate scala și originea folosite de către instrucțiunile PLOT, DRAW, CIRCLE și FILL (nu și pentru POINT care adresează pixeli).

XOS, YOS setează originea
ORG, YRG setează scala

Funcții predefinite:

Se introduce	Se afișază
FN a(	AND
FN b\$(	BIN\$
FN c\$(	CHAR\$
FN c(	COSE
FN d(	DEC
FN p(	DPEEK
FN f(	FILLED
FN h\$	HEX\$
FN i(	INSTRING
FN m(	MEM
FN m\$	MEMORY\$
FN v(	MOD
FN n(	NUMBER
FN o(	OR
FN r(	RNDM
FN s\$(	SCRNS\$
FN s(	SINE
FN t\$(	TIMES\$
FN n\$(	USING\$
FN x(	XOR

1) AND (w1,w2) produce "și logic" între variabilele întregi w1 și w2

2) BIN\$(nr) dă echivalentul binar al lui nr. ca un șir de opt caractere pentru un nr < 255 sau ca un șir de 16 caractere pentru un număr între 256 și 65535.

Dacă se dorește înlocuirea lui "1" și a lui "0" în șirul generat, se vor înlocui cu caracterul dorit locațiile 62865 și 62869.

POKE 62865,"x"

POKE 62869,"y"

3) CHAR\$(nr)

această funcție convertește întregi (între 0 și 65535) în șirul de două caractere, permițînd memorarea unor date numerice cu economie de memorie.

4) COSE(nr)

returnează cosinusul valorii n, cu o precizie mai mică decît funcția standard COS (precizia 4 poziții zecimale) dar de cca. 6 ori mai rapid.

5) DEC(sir)

returnează valoarea zecimală a unui șir pe 1—4 caractere care reprezintă un număr hexazecimal corect.

6) DEEK(adr)

este un PEEK dublu ce furnizează adresa conținută la locația adr echivalent cu:

PEEK adr+256*PEEK (adr+1)

Procedura inversă este DPOKE

7) FILL()

reprezintă numărul de pixeli modificați de ultimul apel al procedurii FILL

8) HEX\$(nr)

argumentul numeric va fi convertit într-un șir de 2 sau 4 caractere.

2 caractere pentru 0,nr,255

4 caractere pentru 256, nr,65535

9) INSTRING (start, sir1, sir2)

caută în șirul sir1 secvența sir2, începînd cu caracterul nr. start din sir1.

Dacă șirul este găsit căutarea se oprește, rezultatul funcției este poziția din șirul sir1 a primului caracter din sir2.

Altfel rezultatul este zero.

E posibilă înlocuirea unora dintre caractere din șirul de căutat cu caracterul "#", care va însemna oarecare.

Singura oară cînd "#" este literalmente căutat este atunci cînd este primul caracter din sir2.

10) MEM()

returnează numărul de bytes ai spațiului de memorie disponibil.

Echivalent cu:

PRINT 65535 — USR 7962

11) MEMORY\$() (început TO sfîrșit)

returnează blocuri de memorie ca șiruri de caractere folosind TO.

În combinație cu posibilitățile Basic de a memora cu POKE șiruri, această funcție dă posibilitatea de a deplasa în memorie mari cantități de date rapid. O altă utilizare este aceea în conjunctură cu INSTRING, ceea ce permite căutări mai rapide în memorie.

12) MOD(nr1, nr2) = nr1 modular nr2

13) OR(nr1, nr2) reprezintă un "sau logic" bit cu bit

14) RNDM (nr)

Dacă nr = 0 reprezintă un număr aleator între 0 și 1

nr ≠ 0 reprezintă o valoare aleatoare între 0 și nr.

15) SCRNS\$ (line, col.)

Lucrează la fel ca SCREEN\$, doar că recunoaște și caracterele definite de utilizator (UDG).

16) SINE (nr)

reprezintă un șir ce cuprinde de n ori juxtapus șirul șir.

18) TIMES()

reprezintă ora ceasului de timp real ca un șir de 8 caractere.

19) XOR (nr1, nr2) "sau exclusiv" bit cu bit

Continuare comenzi:

32) DEF PROC nume (Gr. 1)

— trebuie să fie prima instrucțiune dintr-o linie

— nume = identificator legal

— corpul trebuie încheiat cu ENDPROC

— nu necesită RETURN înainte de

ENDPROC.

33) DEF KEY litera;sir;instr;instr... (Gr. Shift 1)

Beta Basic permite ca șiruri sau linii de program de orice lungime să fie asociate cu orice tastă numerică sau literară.

Exemplu: DEF KEY "1"; "SALUT"

tastând 1 vom avea la baza ecranului, SALUT.

Dacă lipsește ":" la sfârșit, odată șirul introdus, linia este trecută prin analizatorul sintactic și este introdusă în program.

O tastă poate fi asignată cu o valoare oarecare la orice moment de timp, vechea definiție fiind înlocuită. Dacă se folosește un șir vid, sau nici un șir nu urmează după definiția tastei, atunci tasta nu va mai avea nici o definiție asociată. DEF KEY ERASE va șterge toate definițiile pentru taste, care sînt memorate după RAMTOP și astfel sînt protejate, chiar și față de NEW.

Rutina SAVE din Basic va salva orice definiții de taste odată cu programul, salvînd de la RAMTOP pînă la sfârșitul programului Basic.

RAMTOP este coborît automat pentru a permite amplasarea definițiilor pentru taste.

Folosirea lui CLEAR nr va produce schimbarea RAMTOP și va produce probabil pierderea unor definiții.

Rutina următoare deplasează RAMTOP și orice definiții de taste în jos în memorie cu un număr de spații pentru a preveni această problemă. Spațiul liber este creat între sfârșitul

definițiilor de taste și RAMTOP-ul original de 55800, dar acesta se poate schimba.

```
10 INPUT "spatii?";spatiu
```

```
20 LET vechirt=DPEEK(23730)
```

```
30 DPOKE 23728, vechirt
```

```
40 CLEAR vechirt — spatiu
```

```
50 LET vechirt=DPEEK(23728)
```

```
60 LET a$=MEMORY( ) vechirt TO 55800)
```

```
70 LET vechirt=DPEEK 23730
```

```
80 LET spatiu=vechirt—nourt
```

```
90 POKE spatiu,a$
```

```
100 POKE 55801-spatiu,STRING$(spatiu,CHR$(0))
```

Întotdeauna lasă un octet NULL după ultima definiție, ca indicator de sfârșit.

34) POKE adr, a\$

permite scrierea șirului a\$ în memorie, de la adresa adr.

Dacă se dorește încărcarea unui program salvat cu Beta Basic, se va încărca în primul rînd Beta și apoi propriul program. Dacă s-a uitat încărcarea lui Beta Basic la rularea programului va rezulta mesajul: "Nonsense in Basic". Se va putea utiliza MERGE "Beta Basic" : GO TO 2.

Eventualele erori din timpul execuției programelor vor avea ca urmare raporturi specifice Beta Basic sau Basic Spectrum.

CAPITOLUL 4

LIMBAJUL PASCAL. PARTICULARITĂȚI DE IMPLEMENTARE PE CALCULATOARELE HC85 ȘI TIM-S

Definit la începutul anilor '70 de către Wirth pentru a înlesni învățarea programării, limbajul PASCAL a depășit destul de repede sfera universităților fiind adoptat pentru scrierea aplicațiilor în majoritatea domeniilor în care se utilizează calculatorul. Născut prin îmbunătățiri radicale aduse limbajului Algol, Pascalul oferă avantaje atât în ceea ce privește instrucțiunile de control — care sînt chiar cele impuse de tehnica programării structurate (IF—THEN—ELSE, REPEAT—UNTIL, CASE, FOR)—cît și facilități deosebite de puternice pentru reprezentarea datelor. Evitarea sistematică a utilizării instrucțiunii GOTO precum și scrierea modularizată oferă limbajului o inteligibilitate și o claritate foarte bună.

În ceea ce privește implementarea pe calculatoarele compatibile Sinclair Spectrum (HC85, TIM-S, COBRA, etc), avantajul major față de celelalte limbaje disponibile este viteza mare la rulare (datorată faptului că prin compilare se generează cod direct executabil).

4.1. Limbajul PASCAL

Să analizăm un exemplu de program (calculul factorialului):

```
10 PROGRAM EXEMPLU;  
20 VAR I,J,K:INTEGER;  
30 BEGIN  
40   READ(I);  
50   J:=1;  
60   FOR K:=1 TO I DO  
70     J:=J*K;  
80   WRITE('Factorial de ',I,' este ',J)  
90 END.
```

Orice program PASCAL trebuie să înceapă cu o declarație de program. Aceasta constă (în cazul implementării pe HC) din cuvîntul cheie PROGRAM (scris cu majuscule) urmat de un identificator (EXEMPLU în cazul nostru) care reprezintă numele programului. Caracterul ';' este un simbol standard și are semnificație de separator.

Liniile 20—90 (mai puțin punctul) reprezintă un bloc. Blocul este unitatea de bază a programelor PASCAL, el cuprinzînd descrierea datelor și a acțiunilor ce se efectuează asupra lor. Linia 20 specifică faptul că vom utiliza 3 variabile I, J, și K de tip întreg (care nu pot lua decît valori în mulțimea numerelor întregi). Dacă am fi avut nevoie și de valori reale putem scrie:

```
20 VAR I,J,K:INTEGER;  
25   VARIABILA:REAL;
```

Cuvintele cheie BEGIN și END delimitează corpul de instrucțiuni al blocului iar caracterul '.' specifică sfîrșitul programului. Semantica programului este ușor de urmărit. În linia 40 se citește un număr întreg apoi factoriala este inițializată cu 1 și se intră într-un ciclu ce cuprinde o singură instrucțiune în care se calculează produsul. În linia 80 se afișează rezultatul.

Pentru a descrie forma cea mai generală pe care o poate avea un program se utilizează diagrame de sintaxă. Acestea sînt grafuri orientate avînd o singură intrare și o singură ieșire; orice drum posibil de la intrare la ieșire, care urmează sensul de parcurgere indicat de săgeți, definește o construcție admisibilă sintactic.

Sintaxa unui program și cea a unui bloc sînt date în fig. 4 respectiv fig. 5.

4.1.1. Tipuri, constante, variabile

După cum am văzut toate variabilele folosite în interiorul unui bloc trebuie declarate. Mai mult, analizînd diagrama de sintaxă observăm că putem declara și etichete (LABEL), constante simbolice (CONST) și chiar tipuri noi (TYPE).

Prin tip se înțelege în Pascal o mulțime de valori căreia i se poate atașa un nume. Există patru tipuri standard (predefinite): BOOLEAN, CHAR, INTEGER și REAL.

4.1.1.1. Tipuri standard

Tipul BOOLEAN

Reprezintă mulțimea valorilor logice TRUE și FALSE. Există următoarea relație de ordine FALSE < TRUE precum și trei operatori logici NOT, OR și AND (cu semnificațiile cunoscute).

Tipul CHAR

Este format din mulțimea caracterelor ASCII pe care se definește o relație de ordine (după valoarea numerică a codului caracterului). Astfel 'a' < 'b' și '1' < 'A' (de remarcat că pentru a delimita șirurile de caractere se folosește simbolul ' (SS+7) și nu " (SS+p) ca în BASIC).

Tipul INTEGER

Cuprinde o submulțime a multimii numerelor întregi (între -32767 și 32767) pe care este definită relația naturală de ordine. Operatorii aritmetici sînt +, —, *, DIV, MOD. DIV reprezintă cîțul împărțirii întregi iar MOD restul împărțirii. De remarcat că 3/4 este diferit de 3 DIV 4 (prima operație are rezultat real

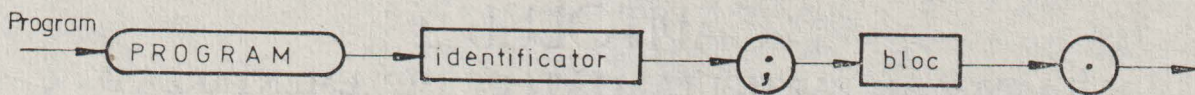


Fig. 4

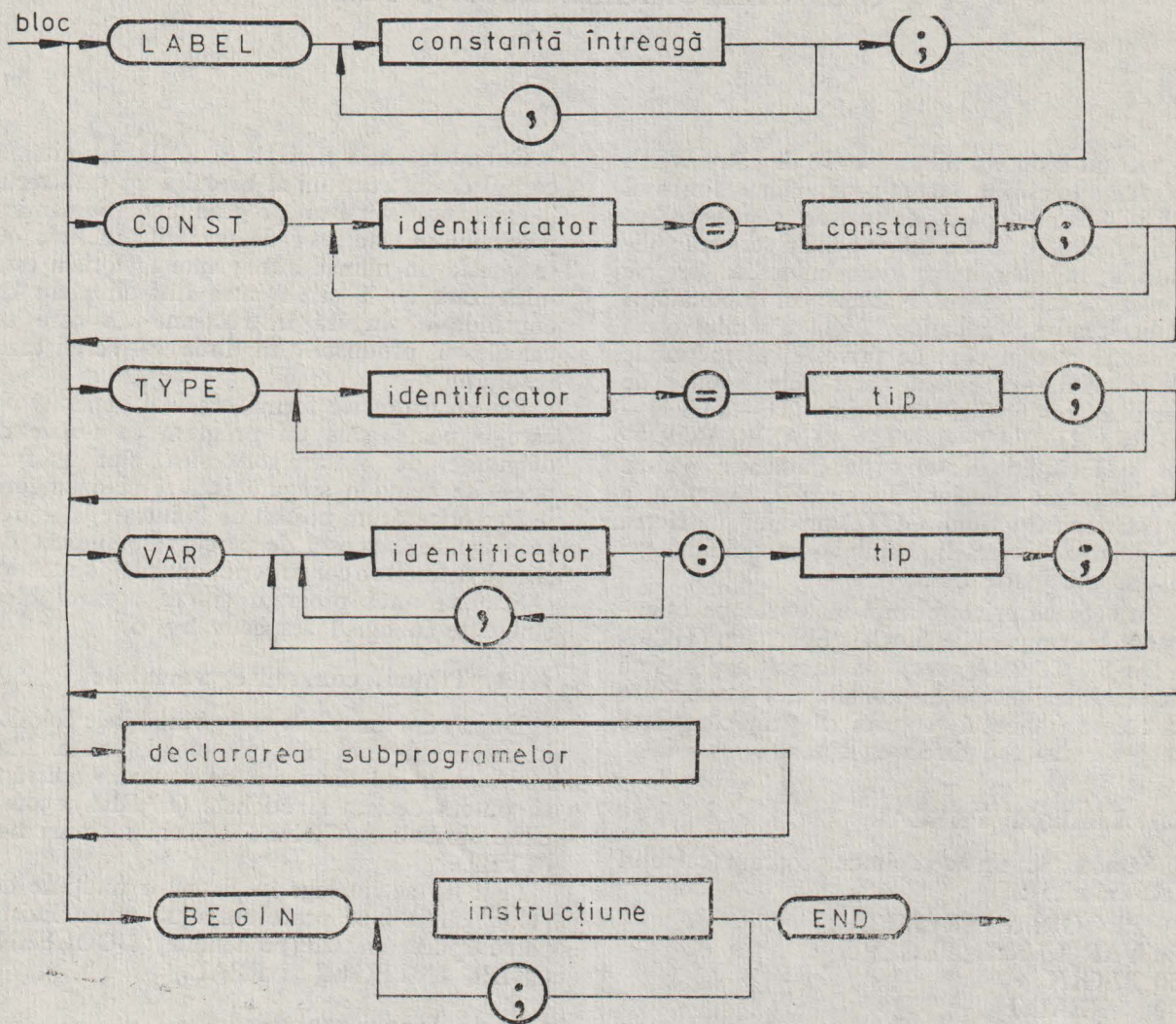


Fig. 5

0.75 iar a doua rezultat întreg 0), deci atenție la comparații.

Tipul REAL

Este o submulțime a numerelor raționale cuprinsă între -10E34 și 10E34 pe care se definesc operatorii matematici obișnuiți.

Sintaxa și semantica expresiilor Pascal este similară cu cea a expresiilor din BASIC sau FORTRAN motiv pentru care nu o mai analizăm. Este totuși utilă specificarea regulilor de prioritate:

- prioritate 0 operatorul NOT (cel mai prioritar)
- prioritate 1 operatorii multiplicativi *,/, DIV, MOD, AND
- prioritate 2 operatorii aditivi +, -, OR
- prioritate 3 operatorii relaționali =, <, >, <=, >=, IN

Dacă există dubii în privința ordinii de evaluare, se pot utiliza paranteze pentru a forța o anumită ordine.

4.1.1.2. Tipuri definite de utilizator

Una din facilitățile deosebite ale limbajului PASCAL este posibilitatea de a defini noi tipuri, putându-se crea astfel o structură de date potrivită descrierii algoritmului ce trebuie implementat. Noile tipuri definite pot fi: tipuri scalare, tipuri structurate sau tip pointer. Cu excepția ultimului tip, care va fi prezentat mai târziu, celelalte sînt prezentate în continuare.

a. Tipuri scalare

Tipul enumerare

Se definește prin enumerarea identificatoarelor ce reprezintă mulțimea valorilor posibile. De exemplu pentru a defini un tip care să descrie familia calculatoarelor compatibile SPECTRUM:

TYPE calculator=(SPECTRUM, HC85, TMS, COBRA);

Acest mod de definire introduce și o relație de ordine (ordinea în care au apărut constantele simbolice în definirea tipului) ce poate fi testată

în program folosind operatori relaționali obișnuiți. Există de asemenea trei funcții predefinite ce operează pe orice tip scalar (mai puțin tipul standard REAL care are o situație aparte). Acestea sînt:

- ORD — întoarce un întreg ce reprezintă numărul de ordine al argumentului (Ex. ORD(HC85)=2)
- PRED — întoarce predecesorul argumentului în lista de definiție (Ex. PRED(TIMES)=HC85, PRED(SINCLAIR) nu este definit !)
- SUCC — întoarce succesorul argumentului în lista de definire (Ex. SUCC(TIMES)=COBRA, SUCC(COBRA) nu este definit !)

Alte exemple de tipuri enumerare:

TYPE zi=(Luni, Marti, Miercuri, Joi, Vineri, Simbata, Duminica);

anotimp=(primavara, vara, toamna, iarna);

note=(DO, RE, MI, FA, SOL, LA, SI, DO);

Se observă apariția o singură dată a cuvîntului cheie TYPE pentru toate definițiile de tipuri care urmează și care sînt separate prin ';'. Ca o observație tipul standard BOOLEAN este un tip enumerare cu definiția

TYPE BOOLEAN=(FALSE, TRUE);

Tipul subdomeniu

Există cazuri în care se cunosc limitele între care poate varia o anumită variabilă de tip scalar. În aceste cazuri tipul variabilei se definește ca un subdomeniu de tip scalar. De exemplu putem defini un tip:

TYPE calcrom=HC85..COBRA;

Se pot defini de asemenea subdomenii ale tipurilor INTEGER și CHAR dar nu și ale tipului REAL:

TYPE indice=1..100;
litere='a'..'Z';
cifre='1'..'9';

Toți operatorii și funcțiile care se pot aplica pe tipul de bază pot lucra și cu variabile de tip subdomeniu. La fiecare atribuire însă se face verificarea încadrării mărimii respective în intervalul declarat.

b. Tipuri structurate

Tipul tablou

Ca și în celelalte limbaje tabloul este o structură formată dintr-un număr fix de componente. Pentru a-l defini trebuie precizat domeniul pe care iau valori indicii și tipul elementelor tabloului:

TYPE tablou=ARRAY[tip-index] OF tip-de-bază;

tip-index poate fi orice tip scalar mai puțin tipul REAL și INTEGER iar tip-de-bază poate fi orice tip.

Exemplu:

TYPE carti=(manuale, cursuri, romane, dictionare);
bilant=ARRAY[carti]OF INTEGER;
linie=ARRAY[1..32] OF CHAR;

total=ARRAY[1..10] OF bilant;
mat1=ARRAY[1..10] OF ARRAY
[1..10] OF REAL;
mat2=ARRAY[1..10, 1..10] OF
REAL;

Ultimele trei tipuri din exemplu evidențiază că tipul de bază poate fi și el un tip structurat. În acest caz adresarea unei variabile A de tip total se face:

A[2] are tipul bilant;
A[2][manuale] are tipul INTEGER

De remarcat ca tipurile mat1 și mat2 nu sînt echivalente. Dacă variabila X este de tipul mat 1 și Y de tipul mat 2 atunci:

X[3] este un vector cu 10 elemente

X[3][5] este un număr real

Y[3,5] este un număr real

Y[3] este o eroare de sintaxă la compilare!

Singurele constante de tip tablou ce pot apărea în program sînt șirurile de caractere ce se consideră ca fiind ARRAY[1..n] OF CHAR (n lungimea șirului).

Este posibil să facem atribuiri cu variabile structurate direct (fără a atribui fiecare element separat), de exemplu:

X[2]:=X[4] copiază toată linia 4 a matricii în linia 2

Tipul mulțime

Acest tip descrie ceea ce se înțelege prin noțiunea matematică de mulțime. Pentru a defini un tip mulțime este necesar să se precizeze tipul elementelor mulțimii (tipul de bază al mulțimii) care trebuie să fie scalar (nu REAL sau INTEGER, dar se admit subdomenii ale tipului INTEGER).

Exemplu:

TYPE fructe=(mar, par, pruna, caisa, gutuie);
fructiera=SET OF fructe;

O variabilă de tip "fructieră" poate lua ca valori orice submulțime a mulțimii valorilor tipului fructe inclusiv mulțimea vidă. Cu mulțimi compatibile (care au același tip de bază) se pot executa o serie de operații: OR (reuniune), AND (intersecție), — (scădere). De asemenea există operatorii relaționali = (egalitate), <> (diferit), <= (incluziune la dreapta), >= (incluziune la stînga), IN (verifică dacă un element se află în mulțime). Cu definițiile de mai sus se poate scrie:

VAR a,b,c:fructiera;

.....

a:=[];

b:=[mar, par];

c:=[caisa] OR b OR c;

IF gutuie IN c THEN...

În acest caz c va fi [mar, par, caisa] iar condiția gutuie IN c va fi falsă. Tipul operandului care apare înaintea lui IN trebuie să fie același cu tipul de bază al mulțimii altfel se semnalează eroare de sintaxă.

Tipul articol

Articolul reprezintă o reuniune de componente de tipuri în general diferite. El este cel

mai general mod de structurare a datelor în PASCAL. Fiecare componentă a unui articol (cîmp al articolului) poate fi diferită ca structură și tip. Accesul la cîmpuri se face prin intermediul unui identificator atașat la definirea articolului.

Exemplu:

```
TYPE caracteristica=RECORD
    nume: ARRAY[1..10] OF CHAR;
    preț: REAL;
    greutate: REAL;
    nr-bucăți: INTEGER
END;
```

O variabilă de tip caracteristică poate astfel grupa o serie de informații despre un obiect (numele, prețul...). Accesul la fiecare cîmp se face adăugînd la numele variabilei separat prin. (punct) numele cîmpului. Acest procedeu se numește calificare.

Exemplu:

```
VAR a,b: caracteristica;
bilant:real;
```

```
.....
a.nume:='calculator';
b.pret:=2000;
```

```
.....
bilant:=bilant+a.pret*a.nr-bucati;
```

Pentru a evita scrierea numelui variabilei de fiecare dată se folosește instrucțiunea WHIT. Astfel ultima instrucțiune devine:

```
WHIT a DO bilant:=bilant+pret*nr-bucati;
```

NOTA: În PASCALul standard există articolul cu variante. Deoarece implementarea nu oferă această posibilitate, nu o descriem. De asemenea nu este implementat tipul fișier.

4.1.2. Instrucțiunile PASCAL

Instrucțiunea de asigurare

După cum am văzut operația de asignare se specifică cu := și nu cu =. Acest lucru este necesar pentru a nu face confuzie între o comparație și o asignare (lucru evitat în BASIC prin folosirea prefixului LET).

Exemple:

```
a:=3;
b:=a*c+d;
logic:=a=b;
```

Instrucțiunea compusă

Majoritatea instrucțiunilor de control în Pascal operează pe o singură instrucțiune. Pentru a le putea folosi este necesar să grupăm mai multe instrucțiuni la un loc.

O instrucțiune compusă este o listă de instrucțiuni separate prin; și cuprinse între cuvintele cheie BEGIN și END.

Exemple:

```
BEGIN
    a:=b*b+c*c;
    a:=SQRT(a);
    b:=a+1;
END;
```

```
BEGIN
    a:=1;
    BEGIN
        b:=0;
        c:=0
    END
END;
```

După cum se vede o instrucțiune compusă poate cuprinde o altă instrucțiune compusă. Pentru a evita greșelile de închidere a grupurilor BEGIN END este recomandat ca alinierea instrucțiunilor să se facă ca în exemplele date. Este permis de asemenea ca separatorul; să apară înaintea lui END (avem o instrucțiune vidă între; și END).

Structuri de control

Instrucțiunea FOR

Sintaxa instrucțiunii este prezentată în fig. 6. Corpul ciclului se execută pentru toate valorile contorului între expresie1 și expresie2. Dacă expresie2 < expresie1 și ciclul se face ascendent atunci instrucțiunea nu se execută de loc (ATENȚIE! ciclul FOR diferă de ciclul DO din FORTRAN în această privință).

Exemplu:

```
10 PROGRAM NUMARA;
20 CONST N=20;
30 VAR I:INTEGER;
40     C:CHAR;
50 BEGIN
60     FOR I:=1 TO N DO
70         WRITELN(I);
80     FOR C:='A' TO 'Z' DO
90         BEGIN
100            I:=ORD(C);
110            WRITELN(I,' ',C)
120        END
130 END.
```

Tipul contorului poate fi oricare tip scalar (mai puțin tipul REAL). La ieșirea din corpul ciclului valoarea contorului este nedefinită.

Instrucțiunea REPEAT

Sintaxa instrucțiunii este prezentată în fig. 7. Corpul ciclului se execută pînă cînd condiția devine adevărată (cel puțin odată).

Exemplu:

```
10 PROGRAM FIBONACCI;
20 VAR F1,F2,F3,N:INTEGER;
30 BEGIN
40     N:=2; F1:=1; F2:=1;
50     REPEAT
60         F3:=F2+F1;
70         F2:=F1;F1:=F3;
80         N:=N+1;
90     WRITELN(F3)
100 UNTIL N>10
110 END.
```

Programul calculează primele 10 numere din șirul lui Fibonacci (care este definit prin recurența $F(n)=F(n-1)+F(n-2)$ cu $F(1)=1$ și $F(2)=1$). Se observă că înainte de cuvîntul cheie UNTIL nu este necesar separatorul; .Existența sa nu este totuși o greșeală (la fel

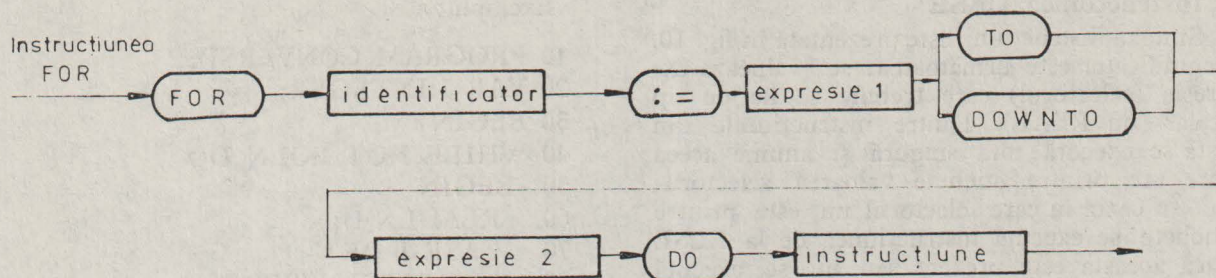


Fig. 6

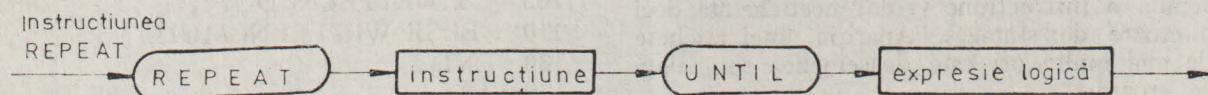


Fig. 7

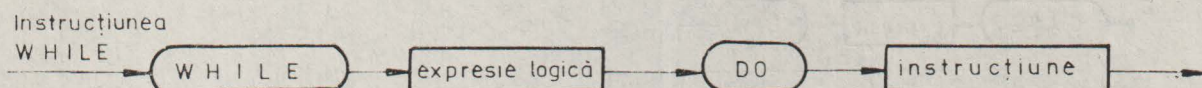


Fig. 8

ca în cazul lui END se consideră o instrucțiune vidă).

Instrucțiunea WHILE

Sintaxa instrucțiunii este prezentată în fig. 8. WHILE codifică ciclul cu test inițial; dacă condiția este falsă corpul ciclului nu se execută nici o dată.

Exemplu:

```
10 PROGRAM SUMA;
20 VAR I:INTEGER;
30 S:REAL;
40 BEGIN
50 I:=1;
60 S:=0;
70 WHILE I<1000 DO
```

```
80 BEGIN
100 S:=S+1.0/I;
110 I:=I+1
120 END;
130 WRITE(' S = ',S)
140 END.
```

Programul SUMA calculează suma $1/I$ până la 999 ($1000 < 1000$ este FALSE și ciclul nu se execută pentru $I:=1000$).

Instrucțiunea IF

Sintaxa sa este prezentată în fig. 9. (ATENȚIE! Înainte de ELSE nu se pune;). Ambele instrucțiuni pot fi instrucțiuni compuse sau instrucțiuni care conțin instrucțiuni compuse

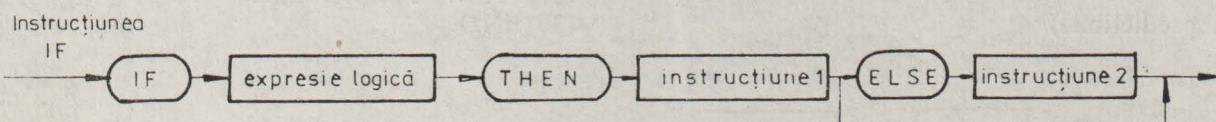


Fig. 9

Exemplu:

```
.....
FOR I:=1 TO N DO
  IF A[I, 1]=0 THEN
    FOR J:= TO N DO
      IF A[J]=0 THEN
        A[I, 1]:=A[I, 1]+1
      ELSE
        A[I, 1]:=A[I, 1]-1
    ELSE
      A[I, 1]:=0;
```

Corpul primei instrucțiuni FOR îl constituie instrucțiunea IF următoare. Aceasta are pe ramura TRUE ciclul FOR după J (care execută IF-ul următor până la al doilea ELSE)

și pe ramura FALSE instrucțiunea $A[I, 1]:=0$.

Regula generală este ca o ramură ELSE aparține întotdeauna celui mai apropiat IF neînchis încă mai puțin cazurile când se specifică altfel prin utilizarea cuvintelor cheie BEGIN și END.

Exemplu:

```
IF cond1 THEN
  BEGIN
    IF cond2 THEN
      instrucțiune1
    END
  ELSE
    instrucțiune 2
```


Instrucțiunea CASE

Sintaxa instrucțiunii este prezentată în fig. 10. Semnificația este următoarea: se evaluează expresia (selectorul) care trebuie să fie de tip scalar (nu REAL). Dintre instrucțiunile din listă se execută una singură și anume aceea care are printre etichete valoarea selectorului. În cazul în care selectorul nu este printre etichete se execută instrucțiunea de la ELSE dacă aceasta este prezent sau nu se execută nimic. **ATENȚIE!** NU se pune; înainte de END-ul sau ELSE-ul lui CASE (ar însemna o instrucțiune (vidă) neetichetată deci o eroare de sintaxă). Apariția unei etichete de mai multe ori este deasemenea semnalată ca eroare.

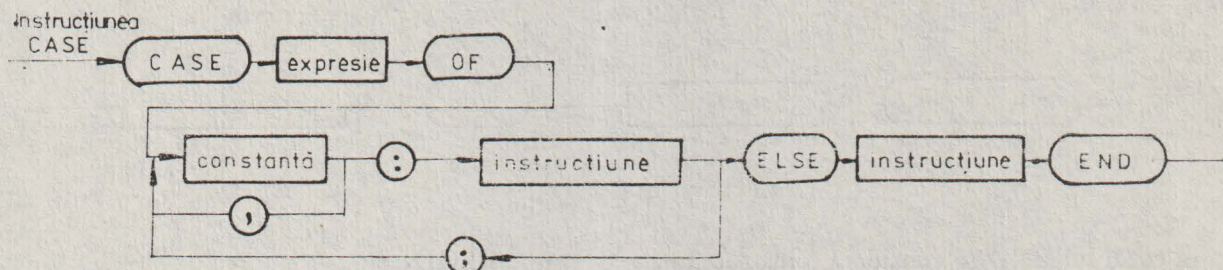
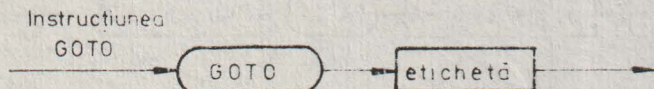


Fig. 10

Instrucțiunea GOTO

Sintaxa sa este prezentată în fig. 11. Etichetele trebuie declarate în blocul curent prin LABEL. Nu este posibil saltul într-un alt bloc sau pe un alt nivel de îmbricare al blocului curent (în cazul definirii de blocuri în blocuri). Folosirea lui GOTO nu este recomandată deoarece diminuează lizibilitatea programului și poate constitui o sursă de erori (de exemplu saltul în interiorul unei bucle FOR are efecte impredictibile).



Exemplu:

```
10 PROGRAM SUMA;
20 LABEL 1;
30 VAR S1,S2,S3:REAL;
40 BEGIN
50 S1:=0;S2:=1;S3:=0;
60 REPEAT
70 S1:=S2+S3;
80 S2:=S2+1/S1;
90 IF S3>5 THEN GOTO 1;
100 S3:=S3+1/S1/S1+1/S2/S2
110 UNTIL FALSE;
120 1: WRITE(S1,S2,S3)
130 END.
```

Fig. 11

Desigur folosirea lui GOTO pentru a afla S1,S2,S3 calculați pentru primul S3 > 5 se poate evita schimbând liniile 90,100,110 după cum urmează:

```
.....
90 IF S3<=5 THEN S3:=S3+1/S1/S1+1/
S2/S2
```

```
100 UNTIL S>35;
.....
```

Instrucțiunea WITH

După cum am văzut această instrucțiune se folosește în conjuncție cu tipul articol pentru a evita scrierea repetată a numelui variabilei. Sintaxa sa este descrisă în fig. 12.

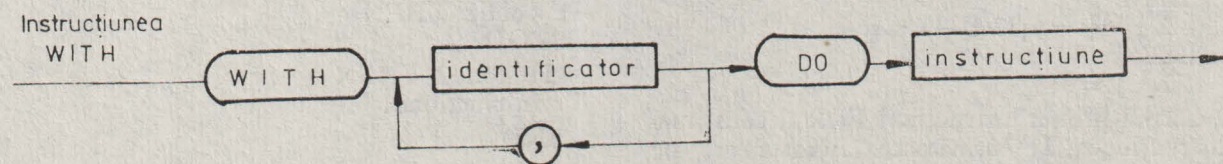


Fig. 12

4.1.3. Proceduri și funcții

Scrierea programelor se face mult mai ușor împărțind problema în mai multe subprobleme relativ independente pentru care se scriu subprograme mai simple. Acest mod de abordare oferă multiple avantaje în special în ceea ce privește claritatea și punerea la punct a programelor rezultate. Pascalul oferă o mare flexibilitate în această privință. Există două moduri în care poate fi organizată o rutină: ca procedură sau ca funcție. Deosebirea rezidă din modul în care se face transmiterea rezultatelor înapoi în programul apelant: prin lista de parametri sau ca valoare asociată numelui rutinei.

Declararea procedurilor și funcțiilor.

Vizibilitatea

În Pascal există o serie de proceduri și funcții standard, ce nu trebuie declarate, ele asigurând o serie de facilități necesare în orice program (proceduri de intrare — ieșire, funcții matematice, conversii între tipuri diferite, accesul direct la memoriei, alocarea variabilelor dinamice etc) și care sînt descrise în partea a doua referitoare la implementare.

Declararea subprogramelor proprii se face înainte de folosirea lor în partea de declarații a blocului. Pentru proceduri sintaxa este descrisă de fig. 13 iar pentru funcții de fig. 14.

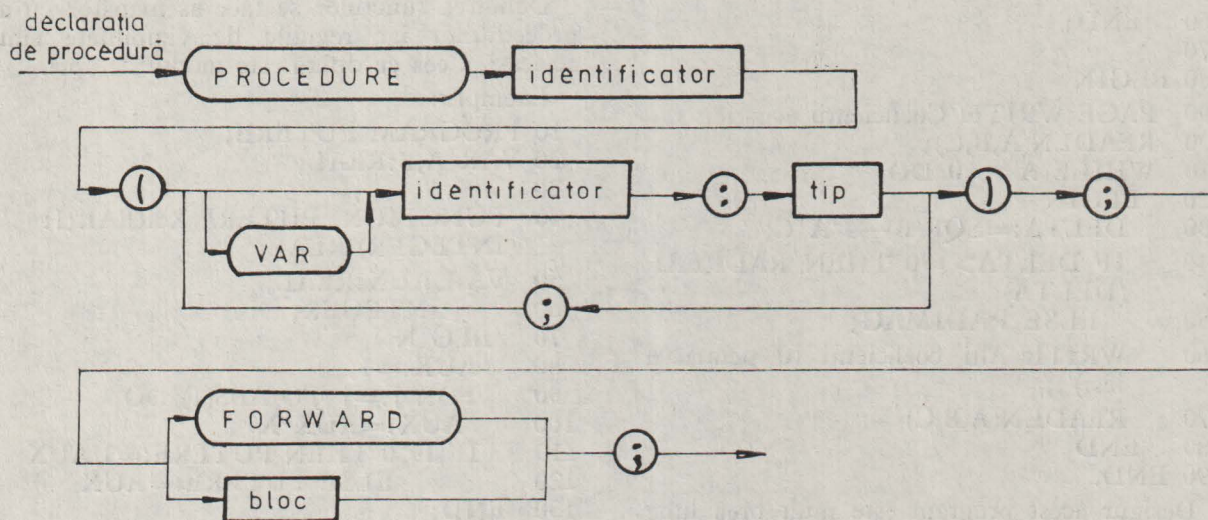


Fig. 13

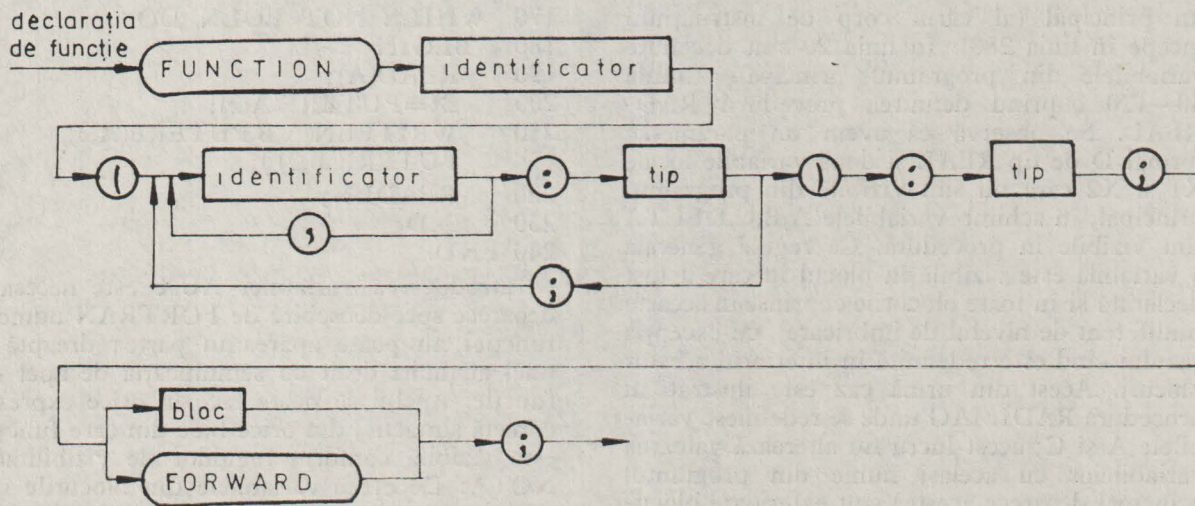


Fig. 14

Exemplu:

```
10 PROGRAM ECGRAD2;
20 VAR A,B,C,DELTA:REAL;
30
40 PROCEDURE RADREAL(D:REAL);
50 VAR X1,X2:REAL;
60 BEGIN
```

```
70 X1:=(B+SQRT(D))/2;
80 X2:=(-B-SQRT(D))/2;
90 WRITELN;
100 WRITELN('Radacinile ecuatiei sint');
110 WRITELN('X1= ',X1); WRITELN
    ('X2= ',X2)
120 END;
```



```

130
140 PROCEDURE RADIMAG;
150 TYPE COMPLEX=RECORD
160   RE,IM:REAL
170 END;
180 VAR A,C:COMPLEX;
190 BEGIN
200   A.RE:=-B/2;C.RE:=-B/2;
210   A.IM:=SQRT(-DELTA);
220   C.IM:=-A.IM;
230   WRITELN('Ecuatia are radacini
    complexe');
240   WHIT A DO WRITELN('X1= ',
    RE,'+', IM, 'i');
250   WHIT C DO WRITELN('X2= ',
    RE,'+',IM,'i')
260 END;
270
280 BEGIN
290 PAGE;WRITE('Coeficientii ecuatiei:');
300 READLN(A,B,C);
310 WHILE A<>0 DO
320   BEGIN
330     DELTA:=SQR(B)-4*A*C;
340     IF DELTA>=0 THEN RADREAL
        (DELTA)
350     ELSE RADIMAG;
360     WRITE('Alti coeficienti (0 pentru a
        iesi)');
370     READLN(A,B,C)
380   END
390 END.

```

Desigur acest program este mult prea lung pentru ceea ce face dar este un bun exemplu pentru a ilustra folosirea procedurilor. Liniile 10—270 constituie partea de declarații a blocului principal (al cărui corp de instrucțiuni începe în linia 280). În linia 20 sînt declarate variabilele din programul principal. Liniile 40—120 cuprind definirea procedurii RADREAL. Se observă că avem un parametru formal D de tip REAL și două variabile locale X1 și X2 care nu sînt vizibile din programul principal, în schimb variabilele A,B,C,DELTA sînt vizibile în procedură. Ca regulă generală o variabilă este vizibilă în blocul în care a fost declarată și în toate blocurile cuprinse în aceasta (indiferent de nivelul de imbricare), cu excepția cazului cînd este redefinită în interiorul acestor blocuri. Acest din urmă caz este ilustrat în procedura RADIMAG unde se redefinesc variabilele A și C; acest lucru nu alterează valoarea variabilelor cu același nume din programul principal deoarece acestea sînt exterioare blocului RADIMAG. Variabilele B și DELTA rămîn vizibile deoarece nu sînt redefinite.

Această regulă de vizibilitate se aplică de fapt nu numai pentru variabile ci pentru orice identificator. Putem astfel avea pe diferite nivele variabile, tipuri, proceduri sau funcții cu același nume.

După cum se vede din diagrama de sintaxă o rutină de tip funcție nu poate avea parametri

formali variabili. Tipul unei funcții trebuie să fie scalar (inclusiv REAL).

Tipul parametrilor de apel trebuie să fie același cu tipul parametrului formal corespunzător din declarația PROCEDURE sau FUNCTION. Aceasta implică faptul că nu se pot transfera variabile de tip anonim (la care definirea tipului s-a făcut în declarația VAR). De exemplu:

```

VAR a : ARRAY[1..10] OF REAL;

```

atribuie lui "a" un tip anonim (fără nume), ceea ce face ca "a" să nu poată fi transferat chiar dacă tipul parametrului formal corespunzător este echivalent (de ex. TYPE vec = ARRAY[1..10] OF REAL).

Definirea funcțiilor se face asemănător cu a procedurilor iar regulile de vizibilitate sînt aceleași. Ceea ce diferă este modul de apel.

Exemplu:

```

10 PROGRAM PUTERE;
20 VAR A,B:REAL;
30
40 FUNCTION PUTERE(X:REAL;I:
    INTEGER):REAL;
50 VAR AUX:REAL;
60   J:INTEGER;
70   BEGIN
80     AUX:=1;
90     FOR J:=1 TO ABS(I) DO
100      AUX:=AUX*X;
110     IF I<0 THEN PUTERE:=1/AUX
120     ELSE PUTERE:=AUX
130   END;
140
150 BEGIN
160   READLN;
170   WHILE NOT EOLN DO
180     BEGIN
190       READ(A);
200       B:=PUTERE(A,2);
210       WRITELN(A,B,PUTERE(A,3),
        PUTERE(B,2));
220     READLN;
230   END
240 END.

```

Introducerea variabilei AUX este necesară deoarece spre deosebire de FORTRAN numele funcției nu poate apărea în partea dreaptă a unei atribuirii decît cu semnificația de apel de funcție. Apelul se poate face în orice expresie corectă sintactic, din orice bloc din care funcția este vizibilă conform regulilor de vizibilitate. NOTA: Deoarece variabilele din blocurile înconjurătoare sînt vizibile în orice rutină internă lor apelul unei proceduri sau al unei funcții poate produce un efect lateral de schimbare a valorii acestor variabile dacă se fac atribuiri și identificatorii respectivi nu sînt redefiniți. Acest lucru se poate întîmpla voit sau ca urmare a unei greșeli de program (cel mai frecvent omiterea redefinirii variabilelor). În cazul în care programul se manifestă "ciudat" la rulare verificați în primul rînd această posibilitate, în

special variabilele folosite pe post de index (de genul I,J,K etc).

Recursivitate

Procedurile și funcțiile Pascal pot apela la rîndul lor alte proceduri și funcții inclusiv pe ele însele. În acest ultim caz avem de-a face cu un apel recursiv, ce poate fi direct sau indirect. Un exemplu de recursivitate directă îl reprezintă următorul mod de calcul al factorialului unui număr:

```
10 PROGRAM FACTORIAL;
20 VAR I:INTEGER;
30
40 FUNCTION FACT(N:INTEGER):
  INTEGER;
50 BEGIN
60   IF N=0 THEN FACT:=1
70   ELSE FACT:=N*FACT(N-1)
80 END;
90
100 BEGIN
110 READ(I);
120 WRITELN;
130 WRITE('Factorial de ',I,' este
    ',FACT(I));
140 END.
```

La apel lucrurile decurg în următorul mod: să zicem că apelăm cu FACT(5). N va fi diferit de 0 și se va executa ramura ELSE. Aceasta însă are nevoie pentru calcul de FACT(4) și așa mai departe. Se creează astfel un șir de așteptare pînă cînd se face apelul FACT(0). În acest moment se execută ramura THEN și FACT va fi 1 pentru ultimul apel. Acum se poate efectua înmulțirea și se rezolvă penultimul apel etc. Putem urmări modul în care decurge recursivitatea adăugînd două linii în program și anume:

```
55 WRITELN('INTRARE',N);
75 ;WRITELN('IESIRE',N);
NOTA: Linia 75 trebuie să înceapă cu ;
deoarece lipsește separatorul după IF în 70.
```

Rezultatul este:

```
INTRARE 5
INTRARE 4
.....
INTRARE 0
IESIRE 0
IESIRE 1
.....
IESIRE 5 120
```

Ultima linie cuprinde și rezultatul (120=5!). Deși această definiție pare echivalentă cu cea iterativă consumul de memorie și timp este mai mare în cazul abordării recursive. Din acest motiv este recomandat să nu se recurgă la o abordare recursivă decît în cazul în care nu există soluție iterativă.

Există situații în care nu este posibilă respectarea regulii de a defini procedurile înainte de a le folosi (de exemplu în cazul a două proceduri care se apelează una pe cealaltă). Soluția problemei este declararea uneia din cele două proceduri FORWARD. Aceasta înseamnă declararea completă a titlului, inclusiv lista de parametri formali și tipul (în cazul funcțiilor) urmată de cuvîntul cheie FORWARD, fără a specifica corpul de instrucțiuni. Procedura se consideră declarată în program în acest punct deși corpul ei de instrucțiuni va apărea la o a doua declarare undeva mai jos. Această a doua declarare se face precizînd numai numele procedurii (fără lista de argumente sau tip).

Exemplu:

```
.....
PROCEDURE UNU(A:REAL); FORWARD;
PROCEDURE DOI(I,J,K);
.....
BEGIN
.....
UNU(X);
.....
END;
PROCEDURE UNU;
BEGIN
.....
DOI(1,I,L);
.....
END;
```

4.1.4. Variabile dinamice

Variabilele declarate în interiorul unui bloc 'trăiesc' atît timp cît este activat blocul respectiv. Ele sînt create automat la intrarea în bloc și distruse la ieșire (Atenție! acest mod de alocare diferă de cel din FORTRAN unde variabilele locale se păstrează între două activări ale unei subrutine), acest mod de utilizare numindu-se alocare statică. Există posibilitatea de a controla crearea și distrugerea variabilelor în mod explicit prin program. Spre deosebire de variabilele alocate static cele alocate dinamic nu sînt accesibile prin numele lor ci prin intermediul unei alte variabile asociate — variabila pointer — ce conține adresa variabilei dinamice.

Tipul pointer

Sintaxa definiției unui tip pointer este prezentată în fig. 15. Deși o variabilă de tip pointer este întotdeauna o adresă, se observă că prin declarare, unui tip pointer i se asociază un tip (tip referit), care poate fi orice tip standard sau definit de programator (în particular poate

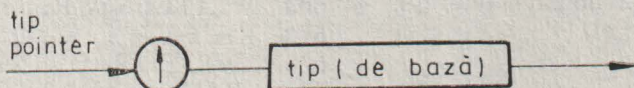


Fig. 15

fi chiar un tip pointer). Pentru variabile pointer de același tip singura operație permisă este asignarea ($=$); se pot aplica operatorii relaționali $=$ și $<$ pentru a verifica dacă două variabile pointer adresează aceeași variabilă dinamică. Există o constantă specială NIL care specifică ca o variabilă pointer nu adresează nici o variabilă dinamică (spre deosebire de orice alte variabile, cele de tip pointer se inițializează în mod automat pe NIL).

Pentru a crea o variabilă dinamică se utilizează procedura NEW iar pentru a elibera memoria se utilizează perechea de proceduri MARK și RELEASE. Descrierea acestor proceduri este dată în partea a doua.

Exemplu:

```
10 PROGRAM EX;
20 TYPE INDICATOR = ^INTEGER;
30   REF = LISTA;
40   LISTA = RECORD
50     INF:INTEGER;
60     LEG:REF;
70   END;
80 VAR I,J,K,L:INDICATOR;
90
100 BEGIN
110   NEW(I);
120   MARK(L);
130   I^:=10;
140   J:=I;
150   J^:=J^+1;
160   WRITELN(I^,J^);
170   NEW(K);
180   K^:=100;
190   I:=K;
200   WRITELN(I^,J^,K^);
210   RELEASE(L);
220 END.
```

În 110 se alocă o variabilă dinamică de tip întreg care se inițializează cu 10 ($I^:=10$). În urma asigurării $J:=I$, J^+1 va fi 11 și deci în 160 $J^$ va fi 11; totodată deoarece I pointează spre aceeași variabilă și $I^$ va fi tot 11. După execuția liniei 190 vom avea $I=K$, $J^=11$, $I^=K^=100$. Execuția procedurii RELEASE va distruge variabila referită de I și K însă nu va reinițializa $I=K=NIL$. Din această cauză folosirea în continuare a lui $I^$ este posibilă însă rezultatele sînt imprevizibile.

Declararea tipului LISTA, care nu a fost folosit, urmărește evidențierea unei excepții, și anume posibilitatea ca tipul referit de o variabilă pointer să nu fi fost declarat pînă în momentul declarării tipului pointer respectiv. Aceasta este singura excepție admisă în PASCAL de la regula definirii înainte de folosire a simbolurilor. Totodată tipul LISTA definește una din structurile tipice ce pot fi create prin alocare dinamică și anume lista simplu înlănțuită. Să urmărim un exemplu de creare și folosire a unei liste pentru ordonarea unui șir de numere.

```
10 PROGRAM ORDON;
20 TYPE REF = ^LISTA;
```

```
30   LISTA = RECORD
40     INF:INTEGER;
50     LEG:REF;
60   END;
70 VAR RAD:REF;
80   I,J:INTEGER;
90
100 PROCEDURE INSERT(A:INTEGER);
110 VAR AUX,ACT:REF;
120   INCA:BOOLEAN;
130 BEGIN
140   IF RAD = NIL THEN
150     BEGIN
160       NEW(RAD);
170       RAD^.INF := A;
180     END
190   ELSE
200     BEGIN
210       NEW(AUX);
220       AUX^.INF := A;
230       IF A < RAD^.INF THEN
240         BEGIN
250           AUX^.LEG := RAD;
260           RAD := AUX;
270         END
280       ELSE
290         BEGIN
300           ACT := RAD;
310           INCA := TRUE;
320           WHILE (ACT^.LEG > NIL)
330             AND INCA DO
340               IF ACT^.LEG^.INF < A THEN
350                 ACT := ACT^.LEG;
360               ELSE
370                 BEGIN
380                   AUX^.LEG := ACT^.LEG;
390                   ACT^.LEG := AUX;
400                   INCA := FALSE;
410                   END;
420                 IF INCA THEN ACT^.LEG :=
430                   AUX;
440                 END;
450 {PROGRAM PRINCIPAL}
460 BEGIN
470   PAGE;
480   WRITELN('Sirul initial');
490   READLN;
500   WHILE NOT EOLN DO
510     BEGIN
520       WHILE NOT EOLN DO
530         BEGIN
540           READ(I);
550           INSERT(I);
560         END;
570       READLN;
580     END;
590   WRITELN;WRITELN('sirul ordonat');
600   REPEAT
610     WRITE(RAD^.INF:4);
620     RAD := RAD^.LEG;
630
```


640 UNTIL RAD=NIL
650 END.

Lista este construită de procedura INSERT. Această procedură primește un număr pe care îl introduce în listă pe poziția necesară pentru ca atunci când lista este parcursă să avem o succesiune crescătoare de valori în câmpul INF. Dacă lista este vidă (RAD=NIL) atunci se introduce imediat primul element, altfel se caută poziția în care trebuie "lipit" noul număr: la stînga să avem numere mai mici iar la dreapta mai mari. Se introduce noul element în listă prin comutarea legăturilor (se "taie lanțul" legăturilor și se adaugă o nouă "za"). Lista astfel creată este apoi parcursă de la rădăcină spre capăt și informația afișată.

* * *

Aici se termină succinta prezentare a limbajului PASCAL. Între a cunoaște limbajul și a-l folosi eficient există însă o diferență pe care numai experiența o poate estompa. Însușirea unui stil de programare adecvat structurilor de control PASCAL (mult diferite de cele BASIC sau FORTRAN) poate părea dificilă la început. În acest sens parcurgerea unui manual de programare structurată este un lucru binevenit.

4.2. Implementarea

Implementarea PASCAL disponibilă pe calculatoarele SINCLAIR SPECTRUM și compatibile aparține firmei HISOFT. Există actualmente în circulație diferite versiuni ale acestei implementări care nu diferă între ele decât prin eliminarea la variantele mai noi a unor 'bug'-uri. Versiunile HP4T15M și HP4T16M sînt cele mai bune (HP4T15M are o eroare la comanda W din care cauză nu se poate compila de pe bandă). În continuare se prezintă versiunea HP4T16M.

Notatii:

CC — CAPS SHIFT and 1 (reîntoarcere în editor)
CH — CAPS SHIFT and 0 ('DELETE')
CI — CAPS SHIFT and 8 (salt la următorul TAB)
CP — CAPS SHIFT and 3 (listare la imprimantă)
CX — CAPS SHIFT and 5 (listează și la imprimantă)
Cs — CAPS SHIFT and SPACE ('BREAK' — abandonează linia în curs)

4.2.1. Încărcarea și lansarea în execuție

Compilerul se încarcă cu LOAD " " sau LOAD "HP4T16M" el intrînd automat în execuție. Înainte de a intra în editor trebuie stabiliți o serie de parametri, răspunzînd la următoarele trei întrebări:
TOP of RAM?

— cea mai mare adresă RAM disponibilă. Poate fi cel mult 65535 (se poate răspunde cu ENTER). Stiva compilerului se pune aici iar adresele superioare pot fi păstrate pentru cod.

TOP of RAM for 'T'?

— se folosește ca vîrf al stivei de Runtime în cazul în care compilarea se face cu translatarea codului. Dacă se răspunde ENTER se ia valoarea de la TOP of RAM. Adresele superioare pot fi păstrate pentru cod.

Table size?

— mărimea zonei de tabele a compilerului. Dacă se răspunde ENTER atunci se va alocă 1/16 din mărimea spațiului disponibil. Valoarea maximă ce se poate specifica este 2184.

După ce s-a răspuns la aceste întrebări controlul este trecut editorului încorporat, lucru indicat de prompterul '>' ce specifică că este așteptată o comandă editor.

Formatul general al unei comenzi editor este C N1,N2,S1,S2

C — comanda de executat

N1,N2 — numere

S1,S2 — șiruri de maximum 20 de caractere

Comenzile se pot introduce atît cu litere mari cît și cu litere mici. Dacă unii parametri nu se precizează, atunci ei se "moștenesc" de la comenzile anterioare. Inițial N1=N2=10 și S1=S2=" ".

Comenzile editor:

I — Insert

pentru a introduce o linie se specifică numărul liniei apoi textul (ca în BASIC) sau se dă:

In,m

cea ce are ca efect numerotarea automată a liniilor de text. Se iese cu CC

Cel mai mare număr de linie este 32767

L — List

listează textul sursă. Are forma

Ln,m

Numărul de linii listate pe un ecran se poate fixa cu 'K'. Listarea se poate abandona cu CC sau continua cu orice altceva. L fără parametri listează tot textul sursă.

K

are forma

Kn

fixează numărul de linii ce se afișează pe un ecran

Comenzi de editare:

Dn,m — Delete

se șterg liniile de la n la m. Ambii parametri sînt obligatorii altfel nu se întîmplă nimic
Mn,m — Move

copiază linia n sub numărul m. Dacă există deja o linie cu numărul m atunci aceasta se pierde

Nn,m — reNumber

renumerotează textul (tot textul!) începînd cu linia n din m în m. Ambii parametri trebuie să fie prezenți altfel nu se întîmplă nimic

F_{n,m,f,s} — Find

caută șirul f între liniile n și m. Dacă se găsește (linia respectivă este afișată și se trece în mod editare. Se poate înlocui f cu s și să se continue căutarea. Parametrii nu sînt obligatorii ci se pot "mosteni" de la un F anterior

V — listează parametrii lui F (ultimii dați)

En — Edit

editează linia n. Dacă n nu se specifică nu se întîmplă nimic, altfel este listată linia și se intră în mod editare. Editarea se face pe o copie a liniei ce se poate abandona.

Subcomenzi de editare (active în mod editare) tipărește caracterul următor

CH merge înapoi în text

CI tipărește pînă la următorul TAB

ENTER se iese din modul editare

Q quit — abandonează editarea

R reload — reîncarcă bufferul de editare ignorîndu-se modificările făcute

L list — listează restul liniei și reintră în editare

K kill — șterge caracterul curent

Z șterge restul caracterelor din linie

F find — caută următoarea apariție a șirului f din comanda 'F' (se iese din editarea liniei curente)

S substitute — înlocuiește șirul f cu s (din comanda 'F') apoi caută următoarea apariție a lui f

I insert — se inserează caractere în poziția curentă a cursorului pînă la primul ENTER cînd se reintră în editare

C corect — serie peste caracterele existente pînă la primul ENTER cînd se reintră în editare

X intră în inserare la sfîrșitul liniei

Comenzi de lucru cu caseta:

P_{n,m,s} — salvează textul de la linia n la m sub numele s

Salvarea începe imediat după ENTER deci casetofonul trebuie pornit înainte. Fișierul se salvează în format HP4T (ca blocuri de cod) G, s — încarcă fișierul cu numele s.

Dacă s nu se specifică se încarcă primul fișier HP4T întîlnit. Comanda se poate abandona cu CS urmat de CC

W_{n,m,s} — salvează textul de la linia n la m sub numele s.

Salvarea se face în blocuri de 128 de octeti pentru a putea fi încărcat textul la compilare (obișnuea F). Aceasta comandă nu poate fi folosită în versiunea HP4T15M.

Comenzi de compilare:

Cn — compilare. Compilează textul începînd de la linia n.

Generează un text de compilare ce se poate opri cu CS (se reia cu orice altceva în afară de CC).

Formatul liniei este

XXXX nnnn text
unde

XXXX — adresa unde începe codul generat de linie

nnnn — numărul liniei

text — textul sursa

Dacă apare o eroare se afișează *ERROR* un pointer după simbolul care a generat eroarea și listarea se oprește. Se poate edita linia în cauză imediat ('E') sau linia anterioară ('P') sau se continuă compilarea cu orice altceva.

Dacă compilarea s-a făcut fără erori apare mesajul RUN? la care se poate răspunde 'Y' pentru rularea imediată sau orice altceva pentru a reintra în editor.

Tn — compile&translate Compilează textul de la linia n și dacă compilarea s-a făcut fără erori emite mesajul 'O.K.?' . Dacă se răspunde 'Y' atunci codul produs de compilare este mutat la sfîrșitul Runtime (se distruge compilatorul și editorul). În același timp se salvează pe banda (decî Casetofonul trebuie pornit anterior!) codul în format HP4T (împreună cu biblioteca ce conține rutinele de I/O, de gestiune a memoriei și de calcul matematic). Numele sub care se salvează este cel din ultimul 'F'. Orice altceva în afară de 'Y' provoacă abandonarea translatareii și reintrarea în editor.

Alte comenzi:

B — ieșirea în BASIC

On,m — provoacă comprimarea textului (dacă acesta a fost elaborat cu alt editor ce nu îl comprimă). Cuvintele cheie se țin minte pe un singur octet.

S, d — schimbă delimitatorul din sintaxa comenzii (virgula) cu caracterul d. (d nu poate fi ' ')

X — afișează adresa de început a textului sursă

R — rulează programul compilat (nu este activă dacă de la ultima compilare s-au făcut modificări în text).

4.2.2. Proceduri și funcții standard

Proceduri de intrare/ieșire

WRITE(p1,p2,...pn)

WRITELN(p1,p2,...pn)

Pi — parametri. Au forma

e — expresie

sau e:m = întregă

sau e:m:n = reală

sau e:m:H = caracter/șir

= boolean

m — lungime totală

n — lungime exponent

H — tipărire în hexa

Caractere de control:

CHR(8) — echivalent cu 'DELETE'

CHR(12) — CLS (salt la pagina nouă)

CHR(13) — CR și LF

CHR(16) — schimbă tipărirea ecran/printer

PAGE

este echivalent cu WRITE(CHR(12)) — salt la pagină nouă

READ(v1,v2,...vn)

Vi — variabile de tip predefinit

are diferite efecte funcție de Vi. La sfârșitul bufferului EOLN este TRUE și dacă este necesar o nouă linie se citește automat de la tastatură.

READLN

citește o linie nouă de la tastatură. EOLN este FALSE imediat după un READLN. (Este TRUE doar dacă se citește o linie de lungime 0) EOLN

funcție de tip Boolean care întoarce TRUE dacă următorul caracter din bufferul de intrare este CHR(13)

INCH

este echivalent cu INKEY\$ în BASIC. Dacă nici o tastă nu este apasată atunci se întoarce CHR(0). Funcția întoarce un rezultat de tip CHAR.

Funcții de transfer:

TRUNC(x) — x întreg sau real. Întoarce un întreg

TRUNC(-1.5)=-1

TRUNC(1.9)=1

ROUND(x) — x întreg sau real. Întoarce un întreg

ROUND(-6.5)=-6 ROUND(11.7)=12

ROUND(-6.51)=-7 ROUND(23.5)=24

ENTIER(x) — x întreg sau real. Întoarce un întreg.

Este echivalent cu INT din BASIC

ENTIER(-6.5)=-7

ENTIER(11.7)=11

ORD(x) — x poate fi orice tip scalar (mai puțin real)

Întoarce o valoare întreagă ce dă ordinalul valorii x

ORD('a')=97

CHR(x) — x trebuie să fie întreg.

CHR întoarce o valoare caracter care corespunde lui ASC(x)

CHR(97)='a'

CHR(49)='1'

Funcții matematice:

(se pot aplica atât pe reali cât și pe întregi)

ABS(x) — modul de x. Are același tip cu x.

SQR(x) — pătratul lui x. Are același tip cu x.

SQRT(x) — radical din x. Întoarce un real.

Dacă $x < 0$ apare o eroare la rulare "Maths Call Error".

FRAC(x) — partea fracționară a unui real (x-ENTIER(x))

SIN(x) — x în radiani. Întoarce un real.

COS(x) — x în radiani. Întoarce un real.

TAN(x) — x în radiani. Întoarce un real

ARCTAN(x) — Întoarce un real.

EXP(x) — Întoarce un real.

LN(x) — dacă $x \leq 0$ apare o eroare "Maths Call Error"

4.2.3. Alte funcții și proceduri predefinite

NEW(p) — alocă spațiul pentru o variabilă dinamică. Variabila p este o variabilă pointer de același tip cu cea alocată. Accesul la variabilă se face cu p^ (p conține adresa variabilei)

MARK(v) — se salvează în v starea stivei de variabile dinamice ce se poate reface cu RELEASE. Tipul lui v nu contează însă el nu se va folosi decât în MARK și RELEASE (niciodată în NEW).

RELEASE(v) — restabilește stiva de la MARK-ul în care s-a alocat v. Se distrug toate variabilele dinamice create de atunci!

INLINE(c1,c2,...) — inserează în codul generat cod Z80 ce se dă în c1,...,cn (constante întregi). Se poate scrie în zecimal sau hexa.

USER(v) — v trebuie să fie întreg. Apelează o rutină în cod de la adresa v. Rutina trebuie să se termine cu RET și să nu altereze registrul IX. Pentru adrese mai mari de 32767 (#7FFF) adresa se specifică în complement față de 2 (de ex #C000=-16384).

HALT — produce oprirea programului cu mesajul

Halt at PC=xxxx

unde xxxx este adresa unde s-a oprit pro-

unde xxxx este adresa unde s-a oprit

programul.

Se folosește la punerea la punct a programului.

POKE(x,v) — pune în memorie expresia v la adresa x. Efectul depinde de tipul lui v, de exemplu:

POKE(#6000,'A') pune la #6000 #41

POKE(#6000,3.6E3) pune la #6000 #00
#0b #80 #70

TOUT(name, start, size) — Salvează variabile pe bandă. Primul parametru este de tip ARRAY[1..8] OF CHAR și reprezintă numele sub care se salvează.

start — adresa de început

size — lungimea în octeți

TIN(name, start) — se folosește pentru a încărca de pe bandă variabile, cod, etc. salvate cu TOUT. Se lucrează în același format cu editorul și deci textele editor pot fi încărcate în program pentru a fi prelucrate.

OUT(p,c) — scoate pe portul p al procesorului caracterul c OUT(254,'A')

RANDOM — întoarce un număr (întreg) aleator între 0 și 255

SUCC(x) — x este scalar (nu real). Întoarce succesorul lui

SUCC('a')='b'

SUCC(5)=6

PRED(x) — Întoarce predecesorul lui x

ODD(x) — x este întreg iar ODD este Boolean și este TRUE dacă x este par (FALSE dacă este impar)

ADDR(x) — x poate fi orice tip. Întoarce un întreg care este adresa de început a lui x

PEEK(x,t) — x este un întreg (adresa) iar t este un tip Execuția depinde de tipul t. De exemplu dacă la adresa #5000 avem #50 #61 #73 #63 #61 #6c atunci

PEEK(#5000,ARRAY[1..6] OF CHAR)=
'PASCAL')

PEEK(#5000,CHAR)='P'

PEEK(#5000,INTEGER)=24912

PEEK(#5000,REAL)=2.46227E29

SIZE(v) — întoarce un întreg care dă lungimea lui v. v poate avea orice tip

INP(p) — întoarce un caracter ce dă valoarea din portul p al procesorului

Opțiuni de compilare

Se pune \$ imediat după paranteza de comentariu și apoi obținți urmate de + (activare) sau — (dezactivare) despărțite prin virgulă.

L list

L+ generează listing la compilare

L— nu se listează decât liniile cu erori

Implicit este L+

O overflow

verificarea depășirilor în virgulă flotantă (pentru scădere și adunare — la * și / se fac întotdeauna)

O+ se fac verificări

O— nu se fac verificări

Implicit este O+

C

verifică tastatura la intrarea în fiecare bloc și dacă s-a apăsât CC execuția se oprește cu mesaj de Halt at PC=xxxxx

Implicit este C+

S stack

verifică dacă mai este loc în stivă la intrarea în fiecare bloc sau procedură. Dacă nu mai este loc și a fost S+ apare mesajul

OUT of RAM at PC=xxxxx

(S— poate duce la blocarea sistemului în cazul depășirii stivei)

Implicit este S+

A array

se verifică dacă indicii de tablouri sînt în valorile declarate. Dacă nu sînt apare un mesaj

Index too high

Implicit este A+

I integer

se verifică operațiile cu numere întregi

(se poate depăși prin calcul MAXINT)

Implicit este I+

P printer

nu este urmat de + sau — și schimbă destinația listingului (ecran <—> imprimanta)

Implicit ecran

F file

trebuie urmat de un spațiu și apoi de opt caractere ce constituie numele unui fișier Pascal pe bandă. Provoacă introducerea în fișierul compilat a unui fișier aflat pe bandă. Aceasta trebuie salvat cu comanda Wn și NU cu Pn. Se pot compila astfel programe foarte mari (doar 128 de octeti se folosesc pentru textul sursă).

Listarea trebuie anterior suspendată altfel compilatorul nu poate să lucreze suficient de repede.

4.2.4. Semnificația codurilor de eroare la compilare

1. număr prea mare

2. ; așteptat

3. identificator nedeclarat

4. identificator așteptat

5. în declararea de constante se folosește '=' nu ':='

6. '=' așteptat

7. cu acest identificator nu poate începe o instrucțiune

8. ':=' așteptat

9. ')' așteptat

10. tip greșit

11. '.' așteptat

12. factor așteptat

13. constanta așteptată

14. acest identificator nu este o constantă

15. 'THEN' așteptat

16. 'DO' așteptat

17. 'TO' sau 'DOWNT0' așteptat

18. '(' așteptat

19. nu poate fi scris acest tip de expresie

20. 'OF' așteptat

21. '..' așteptată

22. ':' așteptat

23. 'PROGRAM' așteptat

24. variabila așteptată deoarece parametrul a fost declarat variabil

25. 'BEGIN' așteptat

26. variabilă așteptată în lista de READ

27. nu se pot compara expresii de acest tip

28. trebuie să fie de tip REAL sau INTEGER

29. nu se pot citi variabile de acest tip

30. acest identificator nu este un identificator de tip

31. se așteaptă exponentul unui număr real

32. expresie scalară nenumerică așteptată

33. '' nu este permis (șirul nul se condică cu CHR(0))

34. '[' așteptat

35. ']' așteptat

36. tipul indicelui de tablou trebuie să fie scalar

37. '..' așteptat

38. ']' sau ',' așteptat în declarația de ARRAY

39. marginea inferioară mai mare decât marginea superioară

40. mulțime prea mare (mai mult de 256 de elemente)

41. funcția trebuie să aibe un tip

42. ')' sau ']' așteptat în SET

43. '..' sau ',' sau ']' așteptat în SET

44. tipul parametrului trebuie să fie tipul identificatorului

45. mulțimea vidă nu poate fi primul factor într-o instrucțiune de ne-asignare

46. scalar (inclusiv real) așteptat

47. scalar (exclusiv real) așteptat

48. mulțimi incompatibile

49. '<' și '>' nu pot fi folosite pentru a compara mulțimi

50. 'FORWARD', 'LABEL', 'CONST', 'VAR', 'TYPE', 'BEGIN' așteptat

54. 'END' sau ';' așteptat în definiția lui RECORD

55. identificator de cîmp așteptat

56. variabila așteptată după WITH

57. variabila din WITH trebuie să fie de tip RECORD
58. identificatorul de câmp nu este asociat cu RECORD
59. întreg fără semn așteptat după LABEL
60. întreg fără semn așteptat după GOTO
61. această etichetă este la un nivel greșit
62. eticheta nedeclarată
63. parametrul lui size trebuie să fie o variabilă
64. se pot folosi doar teste de egalitate pentru pointeri
67. singurul parametru pentru scrierea întregilor cu două ':' este e:m:H
68. șirurile nu pot conține EOL
69. parametrul lui NEW, MARK sau RELEASE trebuie să fie o variabilă pointer
70. parametrul lui ADDR trebuie să fie o variabilă

Mesaje de eroare la rulare

HALT

— oprirea programului ca urmare a întâlnirii unui apel la procedura HALT.

Overflow

— depășire în virgula mobilă (Atenție! Poate fi depășire inferioară — rezultatul unei operații este mai mic în modul decât cel mai mic număr reprezentabil diferit de zero).

OUT of RAM

— depășirea memoriei disponibile (Variabilele declarate nu pot fi create din lipsă de spațiu.) / by 0

— tentativă de împărțire la zero

Index too low

— depășire inferioară a limitei de variație a unui indice de tablou

Index too high

— depășire superioară a limitei de variație a unui indice de tablou

Maths Call Error

— apel ilegal de funcție (de exemplu tentativă de a extrage radical dintr-un număr negativ)

Number too big

— depășire întreagă (număr întreg mai mare decât MAXINT)

Number expected

— eroare de conversie în READ

Line too long

— linia citită în READ are mai mult de 128 de caractere

Exponent expected

— eroare de conversie în READ

4.2.5. Reprezentarea internă a datelor

Întregi

Pe doi octeți, în complement față de doi.

1 → #0001

256 → #0100

−256 → #FF00

Registrul standard pentru lucrul cu întregi este HL.

Caracter, Boolean, scalari

Pe un octet fiecare în forma binară fără semn.

Caracterele în cod ASCII

Boolean

TRUE → #01

FALSE → #00

Registrul standard folosit de compilator este A.

Realii

Pe patru octeți în forma (mantisa, exponent).

semn	mantisă	normalizată	exponent
<u>H</u>	<u>L</u>	<u>B</u>	<u>C</u>

Semnul mantisei

1 — număr negativ

0 — număr pozitiv

Mantisa normalizată are forma 1.xxxxxxx (exceptând reprezentarea lui 0 care este DE=HL=0)

Exponentul este în forma binară în complement față de 2.

Exemplu:

2 → LD HL, #4000

LD DE, #0100

1 → LD HL, #40000

LD DE, #00000

−12.5 → LD HL, #E400

LD DE, #0300

0.1 → LD HL, #6660

LD DE, #FC66

Realii se stochează în memorie în ordinea EDIH.

Întregistrări și tablouri

Întregistrările folosesc același spațiu ca toate componentele la un loc.

Tablourile folosesc n*s unde

n — nr. de componente

s — lungimea unei componente

Mulțimile

Sînt păstrate în memorie ca șiruri de biți se folosesc (n−1) DIV 8+1 unde n este numărul de componente

Pointeri

Ocupă doi octeți ce conțin adresa (în format Intel) a variabilei spre care pointează

Alocarea variabilelor

Variabile globale

Se alocă de la vârful stivei în jos. De exemplu dacă vârful stivei este la #B000 atunci

VAR I: INTEGER;

CH: CHAR;

va aloca

I la #AFFF, #AFFE

CH la #AFFD

Variabile locale

Nu pot fi accesate pe stivă ușor de aceea registrul IX conține începutul blocului curent deci IX-4 pointează spre începutul blocului de variabile.

Exemplu:

PROCEDURE TEST;

VAR I,J:INTEGER;

va aloca

I la IX−4−2, IX−4−1

J la IX−4−4, IX−4−3

Parametrii și valorile de return

Sînt la fel ca variabilele locale numai că adresa crește în sus adică cresc de la IX+2 în sus. Parametrii variabili sînt tratați ca și parametrii constanți dar se alocă doar 2 octeți ce conțin adresa variabilei. Valoarea de return a funcției este prima dintre valorile parametrilor (la IX+2).

Bibliografie

- N. WIRTH, K. Jensen Pascal, user manual and report in Computer Science, 18, Springer Verlag, 1974
*** HISOFT PASCAL user manual, Prentice Hall Inc., 1983
H. Ciocirle, P. Eles, I. Ralla Limbajele de programare Pascal și Pascal Concurrent, Ed Facla, 1985
L. Livovschi, H. Georgescu Bazele informaticii. Algoritmi. Elaborarea și complexitate, curs, București, 1985

CAPITOLUL 5

INTRODUCERE ÎN SISTEMUL dBASE

În cadrul taberei s-a realizat o aplicație pentru ținerea evidenței informațiilor legate de—tabără, sub forma unui fișier — baza de date cuprinzând lista participanților și a rezultatelor obținute.

Acest fișier a fost construit pe un calculator JUNIOR cu ajutorul interpretorului dBASE II pentru limbajul dBASE. Acesta este un limbaj special creat pentru gestionarea bazelor de date, cu scopul de a servi în informatica "de birou", așa numita "birotică". Sistemul dBASE este foarte răspândit în momentul de față (peste un milion de utilizatori, prevăzându-se pentru viitor o utilizare și mai largă și că va înlocui COBOL-ul) mai ales că există de pe acum versiuni mai puternice, dBASE III, de exemplu, pentru calculatoare pe 16 biți (μ P 8086) FELIX PC.

În cele ce urmează se vor introduce în mod progresiv, urmărind aplicația amintită, o serie de elemente ale acestui limbaj pentru ilustrarea stilului de lucru cu un astfel de sistem.

Sistemul dBASE se lansează din sistemul de operare CP/M cu comanda "dBase". Acesta, după ce întreabă data sesiunii de lucru, afișează un punct ("prompter"-ul său), ceea ce înseamnă că așteaptă o comandă.

Prima comandă este crearea unui fișier nou (deoarece încă nu posedăm niciunul) astfel:

- CREATE (CR)

și sistemul cere numele fișierului:

ENTER FILE NAME:

Să presupunem că se răspunde cu *elevi* (CR). Mai departe, se cere structura fișierului:

ENTER RECORD STRUCTURE AS
FOLLOWS:

FIELD, NAME, TYPE, WIDTH, DECIMAL PLACES.

Putem răspunde astfel:

001	nume,	c,	20	(CR)
002	pren,	c,	20	(CR)
003	lic,	c,	10	(CR)
004	jud,	c,	10	(CR)
005	cl,	n,	2	(CR)
006				(CR)

Aceasta înseamnă că fiecare înregistrare a fișierului va conține cinci câmpuri:

- nume și pren. — de tip caracter alfa-numeric și lungime 20

- lic. și jud. — tot de tip caracter, dar lungime 10

- cl. — de tip numeric și lungime 2 (2 cifre).

Tipul numeric este caracterizat în cazul general de doi indicatori: lungime totală și număr de zecimale. Astfel, am putea avea un câmp de forma: înălțime, *n*, 3, 2 cu valori de

tipul înălțime = 1.72 (atenție: punct zecimal, nu virgulă). Mai sus nu s-a dat numărul de zecimale, deoarece cl (clasa) este un câmp de tip întreg (zero zecimale).

Este de menționat că dBASE acceptă și tipul L = Logic.

După indicarea structurii fișierului urmează introducerea datelor. Se poate însă amâna această operație, lăsând astfel o bază de date vidă. Pentru aceasta sistemul întreabă astfel:

INPUT DATA NOW?

Dacă se răspunde cu "y" se trece la adăugarea datelor în modul care se va vedea la comanda APPEND. Dacă se răspunde cu "n" sistemul revine la punctul care indică așteptarea unei noi comenzi.

Disponem în acest moment de un fișier — bază de date — pe care să presupunem că l-am lăsat vid. Pentru a lucra însă cu el acesta trebuie "pus în uz". Aceasta se face cu comanda "USE nume". De exemplu: • USE ELEVI (CR).

Se poate lucra și cu două baze de date simultan, folosind "comutatorul" de fișier numit SELECT. Iată cum se folosește acest comutator:

USE ELEVI (CR)

- se introduce în uz fișierul ELEVI.DBF
- SELECT SECONDARY (CR)
- USE PROFESOR (CR)
- (dacă există fișierul PROFESOR.DBF)
- SELECT PRIMARY (CR)

Astfel fișierul ELEVI devine fișierul "primar" și PROFESOR fișierul "secundar". Asteriscul (sau cuvântul NOTE) indică o linie de comentariu (ca și REM în BASIC).

Comanda APPEND

Să presupunem că avem în lucru fișierul ELEVI și vrem să-i adăugăm înregistrări. Procedăm astfel:

- APPEND (CR)

dBase răspunde cu:

RECORD 00001 (numărul înregistrării)

NUME: :

PREN: :

LIC: :

JUD: :

CL: :

Perechile de puncte delimitează lungimea fiecărui câmp. Ele pot fi eliminate/activate (înainte de a se da comanda APPEND) cu comanda SET COLON OFF/ON.

Mai departe se introduc datele terminând fiecare câmp cu (CR) acolo unde lungimea sa e mai mică decât spațiul rezervat. Trecerea de la o înregistrare la alta se face automat.

Dacă la începutul unei înregistrări se introduce (CR) se iese din APPEND. Iată un exemplu:

- SET COLON OFF
- APPEND

```
RECORD      00001
NUME        POPESCU           (CR)
PREN        TIBERIU          (CR)
LIC         CARAGIALE        (CR)
JUD         PRAHOVA          (CR)
CL          9                 (CR)
RECORD      00002
NUME        STANCA           (CR)
PREN        MARIAN           (CR)
LIC         MAT.-FIZ.        (CR)
JUD         IALOMITA         (CR)
CL          11 (OBS: 2 cifre — nu se mai
                        dă (CR))
```

(ș.a.m.d.)

```
RECORD      0009
NUME        (CR)
```

S-au introdus 8 înregistrări în bază.

În modul APPEND mai sînt disponibile comenzile:

- ctrl/R înscrie pe dischetă înregistrarea curentă și trece la următoarea
- ctrl/cstl/W înscrie înregistrarea curentă și revine la "punct" (iese din APPEND)
- ctrl/Q ignoră înregistrarea curentă și revine la "punct".

În cadrul comenzii APPEND se mai dispune și de unele comenzi de deplasare în fișier (vezi comanda BROWSE).

Cu comanda LIST se poate lista baza de date astfel:

- LIST (CR)

Se poate face și listare selectivă, comandînd de exemplu

- LIST FOR CL=9 (CR)

listează toți elevii din clasa a IX-a.

Pentru corectarea (sau actualizarea) datelor introduce în fișier se folosește comanda

- BROWSE (CR)

La această comandă dBase afișează o "fereastră" prin care se poate privi în fișier, oferind o serie de comenzi pentru modificarea datelor ce se găsesc în poziția cursorului, precum și pentru mișcarea acestuia în fișier.

Aceste comenzi sînt:

```
ctrl/X      sau ctrl/F — mută cursorul pe cîmpul următor
ctrl/D      — mută cursorul un caracter la dreapta
ctrl/S      — mută cursorul un caracter la stînga
ctrl/G      — șterge caracterul indicat de cursor
Del         — șterge caracterul precedent
ctrl/Y      — șterge cîmpul curent
ctrl/V      — comută între modurile inserare și suprascriere
ctrl/B      — mută fereastra cu un cîmp spre dreapta (dacă lungimea înregistrării este mai mare decît ecranul)
ctrl/Z      — idem — spre stînga
ctrl/Q      — iese din BROWSE ignorînd modificările
ctrl/W      — Scrie modificările pe disc și iese din BROWSE
```

Dacă s-a greșit o singură întregire (de ex. 5) se poate proceda astfel:

- GO 5 (sau GO TO 5).
- EDIT.

În modul EDIT se dispune de aceleași comenzi plus ctrl/C și ctrl/R pentru scrierea pe disc a înregistrării curente și trecere la modificarea înregistrării următoare respectiv celei precedente.

După revenirea la punct se poate da comanda DISPLAY pentru afișarea înregistrării curente. Comanda DISPLAY mai are și alte forme între care:

- DISPLAY STRUCTURE
afișează structura fișierului în uz.
- DISPLAY FILES (ON disc — specif.)
afișează bazele de date de pe discul curent (sau cel specificat)
- DISPLAY FILES (ON disc-spezif.) LIKE nume-ambiguu
afișează fișierele de pe discul curent (sau cel specificat) care se potrivesc cu nume-ambiguu (în sens CP/M). Dacă nume-ambiguu = " *.* " se afișează toate fișierele de pe disc.

În aceste din urmă forme DISPLAY este echivalent cu LIST.

Mai departe vom prezenta una din cele mai importante facilități ale unui sistem pentru baze de date: sortarea. În dBASE aceasta se realizează cu comanda SORT. Iată un exemplu:

- SORT ON NUME TO CATALOG

Această comandă creează fișierul CATALOG.DBF care conține înregistrările fișierului în uz (ELEV1.DBF) ordonate după cîmpul NUME. Ordinea este alfabetică (sau cum se mai spune — lexicografică); atenție însă la blancurile inițiale; ordonarea se face după codul ASCII în care blankul este primul caracter (codul 32). Așa că de exemplu valoarea

"ZZZZ" precede valoarea "A".

Dacă în continuare se dă:

- USE CATALOG
- SORT ON CL TO CLASE

se obține fișierul CLASE.DBF în care elevii sînt aranjați în ordinea claselor, iar în cadrul fiecărei clase după nume.

Acum să creăm o altă bază de date cu elevi. O variantă ar fi să o copiem pe cea care o avem deja. Aceasta se face astfel:

- USE ELEV1
- COPY TO ELEV12

Totuși s-ar putea să avem nevoie de o bază de date diferită, dar cu aceeași structură a unei înregistrări (nume-pren-lic-jud-cl).

Pentru a crea o astfel de bază de date dăm comanda:

- COPY STRUCTURE TO ELEV12.

Acum avem fișierul ELEV12.DBF care reprezintă o bază de date vidă. Pentru a umple baza putem folosi comanda APPEND în mai multe moduri:

- 1) Adăugăm noi înregistrări, una cîte una, cum am văzut mai sus.

- 2) Adăugăm selectiv înregistrări din fișierul ELEV1.DBF.

Aceasta se face astfel:

- USE ELEV1
- APPEND FROM ELEV1 FOR CL=9

și se copiază toți elevii din clasa a IX-a.

Să presupunem acum că avem două baze de date cu înregistrări de același tip și vrem să le unim într-o bază centralizatoare care să cuprindă tot ceea ce cuprind ele. Aceasta este, după sortare, o altă facilitate deosebit de importantă a unui sistem pentru baze de date. Comanda care realizează acest lucru este JOIN.

Să unim ELEV1 cu ELEV12 în BAZA.DBF:

- SELECT PRIMARY
- USE ELEV1
- SELECT SECONDARY
- USE ELEV12
- SELECT PRIMARY
- JOIN TO BAZA.

Putem face și o unire selectivă înlocuind ultima comandă de exemplu cu:

• JOIN TO BAZA FOR JUD="PRAHOVA" pentru a obține o bază de date (BAZA.DBF) care conține toate înregistrările din fișierele "PRIMARY" și "SECONDARY" (adică ELEV1 și ELEV12) ce au JUD="PRAHOVA".

Să mai demonstrăm încă două facilități importante ale sistemului:

- 1) Găsirea înregistrărilor care satisfac anumite condiții.
- 2) Tipărirea de informații din baza de date în formă de raport.

Primul lucru se realizează cu comanda LIST sau DISPLAY. Forma generală a acesteia este:

• LIST [<scop>] [<listă de câmpuri>] [FOR <expresie>] [OFF] (parantezele pătrate înseamnă: opțional) unde:

— scop indică unde să se efectueze căutarea și are una din valorile:

- ALL (valoare implicită): în tot fișierul,
- NEXT n dintre următoarele n,
- RECORD n la înregistrarea n,
- lista de câmpuri este ceea ce se dorește

a se afișa din înregistrările găsite; de exemplu: NUME, CL,

— expresie este condiția logică pentru afișare. Ea se compune din expresii relaționale ca: JUD=PRAHOVA sau CL 11, formate cu operatorii relaționale: "<", ">", "<=", ">=", "=" și "<>" (ca în Basic) și legate cu așa zișii conectori logici:

- AND. = și
- OR. = ori (sau)
- NOT. = nu

De exemplu:

(JUD="PRAHOVA" .OR. JUD="CLUJ") .AND. CL=11

— OFF suprimă afișarea numerelor de ordine ale înregistrărilor.

Pentru obținerea de rapoarte tipărite din baza de date se folosește comanda REPORT. Forma generală a acestei comenzi este:

• REPORT [FORM <tip>] [<scop>] [TO PRINT] [FOR expresie] unde:

— scop și expresie au aceiași semnificație ca mai sus;

— TO PRINT cere trimiterea raportului la imprimantă.

— tip este un fișier în care se specifică forma în care se va face raportul — extensia sa implicită este .FRM. Dacă nu avem un astfel de fișier "sărim" peste opțiunea FORM și sistemul va înțelege că trebuie să creeze unul pe loc. De aceea va cere numele fișierului precum și titlul de pe fiecare pagină, capul de tabel, informațiile ce trebuie scrise în el precum și modul de transmitere la imprimantă. Dialogul are loc astfel:

ENTER REPORT FORM NAME: Q (se cere numele fișierului —Q.FRM)

ENTER OPTIONS, M=LEFT MARGIN, L=LINES/PAGE, W=PAGE WIDTH (se cer: lățimea marginii din stânga, numărul de rînduri/pagină și lățimea paginii)

M=15 L=50 W=50

PAGE HEADINGS? (Titlurile de pagini)

DOUBLE SPACE REPORT? (Y/N) (scriere la 2 rînduri?)

ARE TOTALS REQUIRED? (Y/N) (e nevoie de totaluri?)

SUBTOTALS IN REPORT? (Y/N) (total pe fiecare pagină?)

COL WIDTH CONTENTS (coloana, lățimea și conținutul: Cîmpul.

001 25, NUME (Nume pe 25 de poziții)

002

Pentru a încheia, să vedem cum se face un program.

În general un program este un fișier cu extensia CMD care conține mai multe comenzi dBASE. Acest fișier poate fi făcut cu editorul propriu al dBASE-ului prin comanda MODIFY COMAND (în care sînt valabile comenzile de deplasare a cursorului prezentate mai sus) sau orice alt editor de texte: Wordstar, Wordmaster, ED etc.

În cadrul programelor comenzile au unele particularități. Iată cîteva dintre acestea împreună cu un set de instrucțiuni indispensabile programării:

1) Nu are loc dialogul sistemului cu utilizatorul. Pentru a-l elimina se folosește comanda SET TALK OFF (pentru a-l reactiva se dă comanda SET TALK ON). Comanda ERASE șterge ecranul.

2) Este nevoie de variabile care să memoreze anumite valori. Memorarea se face cu comanda STORE. De exemplu:

• STORE 5 TO I

sau

• STORE I TO J

3) Comanda are rolul de a tipări un mesaj la anumite coordonate pe ecran și eventual a

citi într-o variabilă ceea ce se introduce prin comanda READ. Iată câteva exemple:

- 3,20 SAY "Buna ziua"

sau

- 4,5 SAY "Cit e ceasul" GET C

READ

Citirea lui C se face doar la comanda READ; primul număr semnifică rîndul al doilea coloana (coordonatele) unde începe scrierea șirului care urmează după SAY (= "spune,zi").

Citirea se poate face și prin comanda WAIT (= "așteaptă"). În acest caz se așteaptă pînă la introducerea unei valori.

4) Comanda SKIP face trecerea la înregistrarea următoare, comanda APPEND BLANK adaugă o înregistrare inițializată cu spații goale și zerouri, iar REPLACE înlocuiește valorile cîmpurilor cu valori noi (indicate prin "WITH" (adică "cu").

Iată un exemplu de programare care crează un fișier prin selecția anumitor elevi din fișierul ELEVI.DBF:

```
SET TALK OFF
SELECT PRIMARY
USE ELEVI
COPY STRUCTURE TO LISTA
SELECT SECONDARY
USE LISTA
STORE " " TO C
ERASE
DO WHILE .NOT. EOF
@ 9,17 SAY NUME+PREN+ " ? (D/N)"
WAIT TO C
IF C="D" .OR. C="d"
    STORE NUME N
    STORE PREN TO P
    STORE JUD TO J
    STORE LIC TO L
    STORE CL TO CLS
    SKIP
    SELECT SECONDARY
    APPEND BLANK
    REPLACE NUME WITH N, JUD
    WITH J, LIC WITH L, CL WITH
    CLS
    SELECT PRIMARY
    ELSE SKIP
ENDIF
END DO
RETURN
```

Ar mai fi de făcut următoarele observații:
a) bucla cu text inițial, prezentă în cele mai multe limbaje de programare, are forma:

```
DO WHILE condiție
    comenzi
```

```
ENDDO
```

b) EOF = ("END OF FILE") înseamnă sfîrșit de fișier

c) Execuția condiționată are forma:

```
IF condiție
    comenzi
ELSE comenzi
ENDIF
```

în care ELSE — comenzi poate lipsi.

d) Pentru concatenarea șirurilor se folosește operatorul +

e) RETURN nu este neapărat necesar în programul principal, dar dacă se dorește folosirea ca subprogram este obligatoriu. Dacă fișierul de mai sus are numele PROGR.CMD, apelarea sa ca subprogram se face cu comanda:

DO PROG.

În rezumat iată o listă a comenzilor prezentate:

: afișează în coordonate-ecran un mesaj și eventual pregătește o citire:

@ y,x SAY mesaj GET variabilă

APPEND: adaugă înregistrări în fișier

BROWSE: afișare și editare a fișierului prin "fereastră" de mărimea ecranului

CLEAR: închide toate fișierele în uz și șterge toate variabilele din memorie (create în STORE)

COPY: copiază un fișier

COUNT: numără înregistrările care satisfac anumite condiții

CREATE: crează o nouă bază de date

DISPLAY: afișează diferite informații

DO: execută subprograme sau bucle

EDIT: începe editarea de înregistrări din baza de date

ELSE: altfel (vezi IF)

END DO: sfîrșitul unei bucle DO WHILE

END IF: sfîrșitul unei comenzi IF

ERASE: șterge ecranul

GO sau

GO TO: se poziționează înregistrarea specificată

GO 5: înregistrarea 5

GO TOP: început de fișier

GO BOTTOM: sfîrșit de fișier

HELP: oferă ajutor (numai dacă există fișierul dBASE MSG.TXT)

IF: (dacă) permite execuția condiționată a unor comenzi

JOIN: reunește datele din două baze ca și DISPLAY

LIST: din interiorul unui ciclu DO

LOOP: WHILE sare la începutul lui (și verifică îndeplinirea condițiilor)

MODIFY: creează și/sau editează un fișier de comenzi (program) sau modifică structura unei baze de date

NOTE: sau *: comentariu

PACK: șterge înregistrările marcate

RECALL: șterge mărcile de ștergere ale înregistrărilor din fișierul în uz

RELEASE: "uită" variabilele din memorie (eliberează memoria)

RENAME: redenumeste (redefinește un fișier: RENAME (fișier) TO (nume nou))

REPLACE: înlocuiește informații în fișier

REPORT: construiește un raport format din informațiile existente în baza de date

RESET: reinițializează sistemul de operare după schimbarea unei diskete

RETURN: sfârșește un subprogram
SELECT: comută între fișierele în uz
"PRIMARY" și "SECONDARY"
SET: schimbă parametrii de control ai
sistemului
SORT: sortează o bază de date
STORE: memorează date
USE: introduce un fișier în uz. Dacă
nu are parametri îl închide pe
cel existent
WAIT: oprește execuția programului până

ce se introduc date
Toate comenzile se pot prescurta
la patru litere!
QUIT: părăsește sistemul dBASE

Pentru aprofundarea sistemului dBASE se
poate consulta
— revista "Studii și cercetări de calcul econo-
mic și cibernetică economică", numerele
2, 3/1987 și 2, 3/1988 la rubrica "În aju-
torul utilizatorilor de mini- și microcalculatoare".
— revista "Știință și tehnică", numerele 4,5,
6,7,8,9,10/1988, rubrica "Să învățăm dBASE".

CAPITOLUL 6

PROGRAME APLICATIVE

6.1. Grafica

Reprezentările grafice constituie una dintre cele mai spectaculoase aplicații ale calculatoarelor. Acesta a și fost motivul pentru care o parte importantă a programelor realizate au avut această temă. Programele realizate au urmărit:

- a. exemplificarea principalilor algoritmi folosiți în grafica bidimensională:
 - încadrarea reprezentării grafice într-o porțiune de ecran (clipping) — programul 1
 - translații și omotetii — programul 2
 - transformări geometrice plane de rotație și de scalare — programele 3, 4, 5 și 6
- b. exemplificarea principalilor algoritmi folosiți în grafica tridimensională:
 - proiecția — programul 7
 - vizibilitatea — programul 8
- c. utilizarea procedurilor recursive în grafică: realizarea unor modele grafice pornind de la

o figură geometrică simplă — programul 9 și programul 10

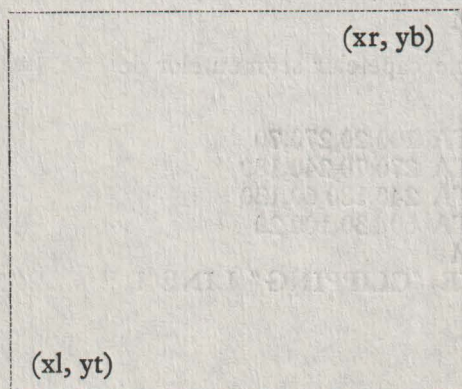
- d. aplicații de grafică tridimensională:
 - reprezentarea funcțiilor de 2 variabile în spațiu — programul 11
 - rotirea corpurilor în spațiu — programul 12.

Programele sînt realizate în limbaj BASIC. Pentru lizibilitate unele dintre ele sînt listate din BETA BASIC putînd fi rulate însă și în BASIC. Programele avînd un scop didactic, sînt bogat comentate, putînd fi astfel înțelese cu ușurință.

Programele au fost realizate de elevii: Toma Gruia, Manolescu Dragoș, Stan Mugurel, Aprodu Marian, Secărea Cristian, Văgii Mihai, Mahler Andreea, Filez Zoltan, Bordy Levente, Cotoi Iulian, sub îndrumarea prof. Huțanu Liliana, prof. Geartu Cristina și st. Stănescu Nicolae.

PROGRAMUL 1

```
1 BORDER 1
  PAPER 1
  INK 7
  CLS
  POKE 23609,30
100 REM
program pentru trasarea unor seg-
mente de dreapta cu decuparea lor
intr-un dreptunghi de afisare
110 REM
parametrii rutinei de decupare sint
```



```
120 REM
x=xl  lim.verticala dreapta
x=xr  lim.verticala stinga
```

```
y=yt  lim.oriz. inferioara
y=yb  lim.oriz. superioara
130 LET xl=75
    LET xr=250
    LET yt=20
    LET yb=170
140 REM
trasarea dreptunghiului de incadrare
150 PLOT xl,yt
    DRAW xr—xl,0
    DRAW 0,yb—yt
    DRAW xl—xr,0
    DRAW 0,yt—yb
160 REM
x(2),y(2)=coord.capetelor segmentu-
lui de decupat
l(),r(),t(),b()=indicatori booleeni(au
valori 0 sau 1)ai depasirii limi-
telor de incadrare
170 DIM x(2)
    DIM y(2)
    DIM l(2)
    DIM r(2)
    DIM t(2)
    DIM b(2)
180 REM
citeste capetele punctelor dreptun-
ghiului
190 READ nrp
200 FOR j=1 TO nrp
```



```

210 FOR i=1 TO 2
220 READ x(i), y(i)
230 NEXT i
240 GO SUB 320
REM

```

rutina de decupare si desenare

```

250 NEXT j
260 STOP
300 REM

```

rutina de decupare si desenare

```

310 REM
1.clasifica capete segment
320 FOR i=1 TO 2
330 GO SUB 610
REM

```

rutina de clasificare a punctelor

```

340 NEXT i
350 REM

```

2. determina daca sementul intersec
tează dreptunghiul de incadrare

```

360 FOR i=1 TO 2
370 LET elim=l(1)*l(2)+r(1)*r
(2)+t(1)*t(2)+b(1)*b(2)
380 IF elim THEN LET i=2
GO TO 410
390 LET dcp=l(i)+r(i)+t(i)+b(i)
400 IF dcp THEN
GO SUB 510
GO TO 370
REM

```

realizează decuparea segmentului pina
cind el dispare sau nu mai avem ce
decupa

```

410 NEXT i
420 IF elim THEN
GO TO 450
430 REM

```

traseaza partea vizibila a segmentului

```

440 PLOT x(1),y(1)
DRAW x(2)-x(1),y(2)-y(1)
450 RETURN
500 REM

```

rutina de decupare a segmentelor de
dreapta

```

510 IF l(i)=1 THEN
LET y(i)=y(1)+(y(2)-y(1))
*(x1-x(1))/(x(2)-x(1))
LET x(i)=x1
GO TO 610
520 IF r(i)=1 THEN
LET y(i)=y(1)+(y(2)-y(1))

```

```

*(xr-x(1))/(x(2)-x(1))
LET x(i)=xr
GO TO 610
530 IF t(i)=1 THEN
LET x(i)=x(1)+(x(2)-x(1))
*(yt-y(1))/(y(2)-y(1))
LET y(i)=yt
GO TO 610
540 IF b(i)=1 THEN
LET x(i)=x(1)+(x(2)-x(1))
*(yb-y(1))/(y(2)-y(1))
LET y(i)=yb
GO TO 610
600 REM

```

rutina de clasificare a punctelor fata
de limitele dreptunghiului de inca
drare

```

610 LET l(i)=0
LET r(i)=0
LET t(i)=0
LET b(i)=0
620 REM

```

codificarea se face astfel

lrtb 1001	lrtb 0001	lrtb 0101
(xr,yb)		
lrtb 1000	lrtb 0000	lrtb 0100
(xl,yt)		
lrtb 1010	lrtb 0010	lrtb 0110

```

630 IF x(i)<xl THEN LET l(i)=1
640 IF x(i)>xr THEN LET r(i)=1
650 IF y(i)<yt THEN LET t(i)=1
660 IF y(i)>yb THEN LET b(i)=1
670 RETURN
700 DATA 4
REM

```

numărul de segmente de desenat
710 REM

coordonatele capetelor segmentelor de
desenat

```

720 DATA 100,20,270,70
730 DATA 270,70,240,180
740 DATA 240,180,60,130
750 DATA 60,130,100,20
760 REM
9999 SAVE "CLIPPING" LINE 1

```

PROGRAMUL 2

```

1>BORDER 1
PAPER 1
INK 7

```

```

POKE 23609,30
100 REM
programul realizeaza trasarea

```


parametrizata a unei figuri pentru
a exemplifica modul de realizare
a translatiilor si omotetiilor

110 REM

a,c = factorii de scalare definesc
omotetia

b,d — termenii de deplasare definesc
translatia

120 LET a=10

LET b=10

LET c=5

LET d=10

130 FOR j=1 TO 8

140 READ x,y

GO SUB 500

150 REM

aplica transformarea geometrica
punctului curent

160 GO SUB 500

170 REM

deseneaza punctul initial

180 PLOT sx,sy

190 REM

deseneaza conturul figurii

200 FOR i=1 TO 4

210 REM

citeste coord.punct de pe contur

220 READ x,y

230 REM

aplica transformarea geometrica
punctului de pe contur

240 GO SUB 500

250 REM

traseaza segment de contur

260 DRAW sx-PEEK 23677,sy-

PEEK 23678*

270 NEXT i

280 REM

modifica parametrii transformarii geometrice

290 LET a=a+12

LET b=b+13

300 LET c=c+9

LET d=d+10

310 RESTORE

320 NEXT j

330 STOP

500 REM

aplica transformarea geometrica punctului
curent

510 LET sx=a*x+b

LET sy=c*y+d

520 RETURN

700 REM

figura de desenat este un patrat cu latura
unitara, insa prin omotetia aplicata
pe ecran obtinem dreptunghiuri

710 DATA 0,0,1,0,1,1,0,1,0,0

730 REM

9999 SAVE "DREPTUNGHI" LINE 1

* realizează un DRAW absolut

PROGRAMUL 3

1> POKE 23609,30

BORDER 1

PAPER 1

INK 7

CLS

10 REM

programul arata efectul transforma-
rilor geometrice plane de rotatie
si de scalare, descrise cu ajutorul
matricilor 2×2

11 REM

pentru a incadra corect figurile la
centrul ecranului vom folosi o tran-
slatie cu parametrii xoff și yoff si
o omotetie cu parametrii sx si sy

100 LET xoff=120

LET yoff=80

LET sx=40

LET xy=40

105 REM

deseneaza cadrul si axele de coordonate

110 PLOT 0,0

DRAW 255,0

DRAW 0,175

DRAW -255,0

DRAW 0, -175

PLOT xoff,0

DRAW 0,175

PLOT 0,yoff

DRAW 255,0

140 REM

introducerea elementelor matricii de trans-
formare

T = $\begin{vmatrix} A & B \\ C & D \end{vmatrix}$

unde $0 \leq A, B, C, D \leq 1$

150 INPUT "A(-1 <= A <= 1) = "; A

153 IF NOT (-1 <= A AND A <= 1)

THEN GO TO 150

155 PRINT AT 1,1; "A = "; A

160 INPUT "B(-1 <= B <= 1) = "; B

163 IF NOT (-1 <= B AND B <= 1)

THEN GO TO 160


```

165 PRINT AT 2,1;"B=";B
170 INPUT "C(-1<=C<=1)=";C
173 IF NOT (-1<=C AND C<=1)
THEN GO TO 170
175 PRINT AT 3,1;"C=";C
180 INPUT "D(-1<=D<=1)=";D
183 IF NOT (-1<=D AND D<=1)
THEN GO TO 180
185 PRINT AT 4,1;"D=";D
190 REM
deseneaza figura supusa transformarii geome-
trice
250 FOR I=1 TO 4
260 READ x,y
265 REM
aplica transformarea T introdusa
270 GO SUB 500
273 REM
aplica transformarea de vizualizare pe ecran
275 GO SUB 600
277 REM
traseaza capat segment
280 PLOT Xd,Yd
290 READ x,y
300 GO SUB 500
305 GO SUB 600
310 DRAW Xd - PEEK 23677, Yd -
PEEK 23678
320 NEXT I
330 REM
340 PRINT #1;AT 1,0;" APASATI 0
TASTA PT. CONTINUARE"
PAUSE 0
RESTORE
345 REM

```

```

afiseaza menu pt. continuare
350 PRINT #1;AT 0,0; "APASA 'N'
PT. O NOUA MATRICE SAU 'S'
PT. SFIRSIT "
LET S$=INKEY$
360 IF S$=" " THEN GO TO 350
365 IF S$(1)="N" OR S$(1)="n" T
HEN CLS
GO TO 100
370 IF S$(1)="S" OR S$(1)="s" T
HEN
CLS
STOP
380 GO TO 350
500 REM
transformarea geometrica introdusa
510 LET tx=A*x+C*y
LET y=B*x+D*y
LET x=tx
520 RETURN
600 REM
tranformarea de vizualizare
610 LET Xd=sx*x+xoff
LET Yd=sy*y+yoff
620 RETURN
700 REM
coordonatele colturilor figurii transformate si
desenate
710 DATA 0,0,1,0
720 DATA 1,0,1,1
730 DATA 1,1,0,1
740 DATA 0,1,0,0
750 REM
9999 SAVE "MATRIX 2x2" LINE 1

```

PROGRAMUL 4

```

1>POKE 23609,30
BORDER 1
PAPER 1
INK 7
CLS
10 REM
programul arata efectul transformarilor geo-
metrice plane de rotatie, de scalare si trans-
latie descrise cu ajutorul matricelor 3x3 si al
coordonatelor omogene
11 REM
pentru a incadra corect figurile la centrul
ecranului vom folosi o translatie cu parametrii
xoff si yoff si o omotetie cu parametrii sx
si sy
100 LET xoff=120
LET yoff=80
LET sx=20

```

```

LET sy=20
105 REM
deseneaza cadrul si axele de coordonate
110 PLOT 0,0
DRAW 255,0
DRAW 0,175
DRAW -255,0
DRAW 0,-175
PLOT xoff,0
DRAW 0,175
PLOT 0,yoff
DRAW 255,0
140 REM
introducerea elementelor matricii de transfor-
mare

$$T = \begin{vmatrix} A & B & 0 \\ C & D & 0 \\ H & K & 1 \end{vmatrix}$$

unde  $0 \leq A,B,C,D \leq 1$ 

```



```

      H,K reali
150 INPUT "A(-1 <= A <= 1) = ";A
153 IF NOT (-1 <= A AND A <= 1)
THEN GO TO 150
155 PRINT AT 1,1; "A = ";A
160 INPUT "B(-1 <= B <= 1) = ";B
163 IF NOT (-1 <= B AND B <= 1)
THEN GO TO 160
165 PRINT AT 2,1; "B = ";B
170 INPUT "C(-1 <= C <= 1) = ";C
173 IF NOT (-1 <= C AND C <= 1)
THEN GO TO 170
175 PRINT AT 3,1; "C = ";C
180 INPUT "D(-1 <= D <= 1) = ";D
183 IF NOT (-1 <= D AND D <= 1)
THEN GO TO 180
185 PRINT AT 4,1; "D = ";D
190 INPUT "H(-2 <= H <= 2) = ";H
193 IF NOT (-2 <= H AND H <= 2)
THEN GO TO 190
195 PRINT AT 5,1; "H = ";H
200 INPUT "K(-2 <= K <= 2) = ";K
203 IF NOT (-2 <= K AND K <= 2)
THEN GO TO 200
205 PRINT AT 6,1; "K = ";K
240 REM

```

deseneaza figura supusa transformarii
geometrice

```

250 FOR I=1 TO 4
260   READ x,y
265   REM

```

aplica transformarea T introdusa

```

270   GO SUB 500
273   REM

```

aplica transformarea de vizualizare pe ecran

```

275   GO SUB 500
277   REM

```

traseaza capat segment

```

280   PLOT Xd,Yd
290   READ x,y
300   GO SUB 500
3 5   GO SUB 600

```

```

310   DRAW Xd-PEEK 23677,Yd-
PEEK 23678
320 NEXT I
330 REM
340 PRINT #1;AT 1,0;" APASATI 0
TASTA PT. CONTINUARE"
      PAUSE 0
      RESTORE
345 REM

```

afiseaza menu pt. continuare

```

350 PRINT #1;AT 0,0;"APASA 'N'
PT. 0 NOUA MATRICE      SAU 'S'
PT.SFIRSIT              "
      LET $$=INKEY$
360 IF $$= " " THEN GO TO 350
365 IF $$(1)="N" OR $$(1)="n" THEN
      CLS
      GO TO 100
370 IF $$(1)="S" OR $$(1)="s" THEN
      CLS
      STOP
380 GO TO 350
500 REM

```

transformarea geometrica introdusa

```

510 LET tx=A*x+C*y+H
      LET y=B*x+D*y+K
      LET x=tx
520 RETURN
600 REM

```

transformarea de vizualizare

```

610 LET Xd=sx*x+xoff
      LET Yd=sy*y+yoff
620 RETURN
700 REM

```

coordonatele colturilor figurii transformate si
desenate

```

710 DATA 0,0,1,0
720 DATA 1,0,1,1
730 DATA 1,1,0,1
740 DATA 0,1,0,0
750 REM
9999 SAVE "MATRIX 3x3" LINE 1

```

PROGRAMUL 5

```

1 > BORDER 1
      PAPER 1
      INK 7
      CLS
      POKE 23609,30
10   REM

```

programul deseneaza o figura prin compunerea
mai multor transformari geometrice succesive,
asupra unui punct initial

```

110   REM

```

pentru a compensa faptul ca ecranul calcula-
torului are punctele mai apropiate pe verticala
decit pe orizontala vom folosi un factor de for-
ma SC, egal cu 1 pentru HC 85 si cu alte valori
pentru alte calculatoare

```

120 LET SC=1
125 REM

```

X,Y sint coordonatele punctului curent, la
inceput X=x0=70 si Y=y0=0

CX,CY sînt componentele unei translatii aplicate pentru a situa originea sistemului de coordonate "in" care se deseneaza la centrul ecranului, deoarece transformările geometrice au drept centru de rotatie originea sistemului de axe

```
130 LET X=70
    LET Y=0
    LET CX=120
    LET CY=80
```

```
135 REM
```

unghiul de rotatie

```
140 LET C=COS (PI/3)
    LET S=SIN (PI/3)
```

```
150 REM
```

deseneaza un hexagon

```
200 FOR I=0 TO 6
```

```
210 LET SX=X+CX
    LET SY=CY-Y*SC
```

```
220 IF I=0 THEN PLOT SX,SY
```

```
230 DRAW SX-PEEK 23677,SY-
PEEK 23678
```

```
240 LET XN=X*C-Y*S
```

```
    LET Y=X*S+Y*C
```

```
    LET X=XN
```

```
250 NEXT I
```

```
260 STOP
```

```
9999 SAVE "HEXAGON" LINE 1
```

PROGRAMUL 6

```
1 > BORDER 1
```

```
    PAPER 1
```

```
    INK 7
```

```
    CLS
```

```
    POKE 23609,30
```

```
100 REM
```

programul deseneaza mai multe poligoane asemenea rotite intre ele, tot mai mici, realizate prin aplicarea unei rotatii si a unei omotetii

```
110 CLS
```

```
120 INPUT AT 0,0;"CITE LATURI S
A AIBA POLIGONUL ?" L(3 <= L
```

```
<=100)="";L
130 IF NOT (3 <= L AND L <=100)
```

```
THEN GO TO 120
```

```
140 REM
```

deseneaza dreptunghiul de incadrare

```
150 PLOT 0,0
```

```
    DRAW 255,0
```

```
    DRAW 0,175
```

```
    DRAW -255,0
```

```
    DRAW 0,-175
```

```
160 REM
```

alegem un factor de forma egal cu 1 pentru HC85

```
170 LET SC=1
```

```
180 REM
```

punctul initial de unde incepe desenarea poligonului

```
190 LET X=80
```

```
    LET Y=0
```

```
200 REM
```

parametrii translatiei ce muta originea sistemului de axe la centrul ecranului

```
210 LET CX=128
```

```
    LET CY=88
```

```
220 REM
```

parametrii rotatiei aplicate pentru determinarea colturilor

```
230 LET C=COS (2*PI/L)
```

```
    LET S=SIN (2*PI/L)
```

```
240 REM
```

parametrii rotatiei aplicate fiecarui nou poligon desenat

```
250 LET C1=COS (PI/6/L)
```

```
    LET S1=SIN (PI/6/L)
```

```
260 REM
```

parametrul omotetiei aplicate pentru micșorarea poligoanelor

```
270 LET SF=.95
```

```
280 REM
```

"deseneaza poligoanele in numar de 40"

```
290 FOR J=1 TO 40
```

```
300 REM
```

deseneaza un poligon

```
310 FOR I=0 TO L
```

```
320 REM
```

translatia aplicata colturilor pentru a face incadrarea la centrul ecranului

```
330 LET SX=X+CX
```

```
    LET SY=CY-Y*SC
```

```
340 REM
```

deseneaza laturile poligonului

```
350 IF I=0 THEN PLOT SX,SY
```

```
360 DRAW SX-PEEK 23677,SY-
```

```
PEEK 23678
```

```
370 REM
```

transformarea de rotatie aplicata pentru determinarea colturilor

```
380 LET NX=X*C-Y*S
```

```
    LET Y=X*S+Y*C
```

```
    LET X=XN
```

```
390 NEXT I
```

```
400 REM
```

transformarea de rotatie si omotetie aplicata poligonului urmator

```
410 LET XN=SF*(X*C1-Y*S1)
```

```
    LET Y=SF*(X*S1+Y*C1)
```



```

LET X=XN
420 NEXT J
430 REM
scrie menu-ul
440 PRINT #1;AT 1,0;" APASATI 0 TA-
STA PT. CONTINUARE"
PAUSE 0
450 PRINT #1;AT 0,0;"APASATI 'N' PT.
NOI POLIGOANE ""

```

```

'S' PT. SFIRSIT "
460 LET A$=INKEY$
IF A$=" " THEN GO TO 460
470 IF A$="N" OR A$="n" THEN
GO TO 100
480 IF A$="S" OR A$="s" THEN
STOP
490 GO TO 450
9999 SAVE "POLIGOANE" LINE 1

```

PROGRAMUL 7

```

1 >POKE 23609,30
BORDER 1
PAPER 1
INK 7
CLS
100 REM "
programul ilustreaza efectul schimbarii punc-
tului de observatie la reprezentarea in per-
spectiva prin proiectie paralela a unei figuri;
aceasta este un cub avind una dintre fete
marcata cu o diagonala pentru a identifica
mai usor modificarea petrecuta"
110 REM
parametrii translatiei ce muta o rignea siste-
mului de axe al ecranului la centrul sau
120 LET XOFF=128
LET YOFF=86
130 REM "
parametrii punctului de observare dat in coor-
donate sferice pentru a simplifica formulele
RHO distanta intre originea sistemului de axe
si punctul de observare
THETA, PHI unghiurile facute de directia de
observare cu axele D distanta intre punctul
de observare si planul ecranului"
140 LET RHO=10
LET D=200
LET THETA=.7
LET PHI=1.3
150 REM
deseneaza cubul vazut din punctul de obser-
vare curent
160 GO SUB 510
170 REM "
menu-ul pentru modificarea parametrilor de
afisare a figurii reprezentate in perspectiva"
180 PRINT AT 1,1;"RHO=";RHO
190 PRINT AT 2,1;"D=";D
200 PRINT AT 3,1;"THETA=";THETA
210 PRINT AT 4,1;"PHI=";PHI
220 PRINT #1;AT 0,0;"PT. SCHIMBA-
REA PARAMETRILOR"; INVERSE 1;"C
";INVERSE 0;"PT. INCHEIERE
"; INVERSE 1;"S"; INVERSE 0
230 LET A$=INKEY$
IF A$=" " THEN GO TO 230

```

```

240 IF A$="S" OR A$="s" THEN CLS
STOP
250 IF A$="C" OR A$="c" THEN GO
TO 280
260 GO TO 230
270 REM
citeste noii parametrii de vizualizare
280 LET CRT=1
290 PRINT AT 1,1;"RHO=";RHO;"
300 PRINT AT 2,1;"D=";D;"
310 PRINT AT 3,1;"THETA=";THETA
;"
320 PRINT AT 4,1;"PHI=";PHI;"
330 PRINT AT CRT,0;">"
340 PRINT #1;AT 0,0; INVERSE 1;
"1-SUS Q-JOS C-ALEGE E-AFISA-
RE"; INVERSE 0;"
350 LET A$=INKEY$
IF A$=" " THEN GO TO 350
360 IF A$="1" THEN IF CRT < > 1
THEN PRINT AT CRT,0;" "
LET CRT=CRT-1
PRINT AT CRT,0;">"
370 IF A$="Q" OR A$="q" THEN IF
CRT < > 4 THEN PRINT AT CRT,0;" ";
LET CRT=CRT+1
PRINT AT CRT,0;">";
380 IF A$="C" OR A$="c" THEN GO
TO 450
390 IF A$="E" OR A$="e" THEN GO
TO 420
400 GO TO 350
410 REM
deseneaza cubul vazut din punctul curent de
observare
420 GO SUB 510
430 REM
afiseaza parametrii de vizualizare si reia ciclul
440 GO TO 180

```



```

450 IF CRT=1 THEN
    INPUT "RHO=";RHO
    GO TO 290
460 IF CRT=2 THEN
    INPUT "D=";D
    GO TO 290
470 IF CRT=3 THEN
    INPUT "THETA=";THETA
    GO TO 290
480 IF CRT=4 THEN
    INPUT "PHI=";PHI
    GO TO 290
490 GO TO 290
500 REM "

```

reprezinta in perspectiva prin
proiectie paralela cubul "

```

510 CLS
    PLOT 1,1
    DRAW 253,0
    DRAW 0,174
    DRAW -253,0
    DRAW 0,-174
    RESTORE
520 LET S1=SIN THETA
    LET C1=COS THETA
530 LET S2=SIN PHI
    LET C2=COS PHI

```

540 REM
X,Y,Z coordonatele carteziene
ale unui colt

```

550 READ X,Y,Z
560 REM "

```

rutina de proiectie in perspectiva "

```

570 GO SUB 820
580 REM

```

deseneaza primul colt

```

590 PLOT SX,SY
600 REM

```

deseneaza cubul

```

610 FOR I=1 TO 11
620   READ X,Y,Z
630   REM "

```

rutina de proiectie in perspectiva "

```

640   GO SUB 820
650   REM

```

deseneaza o latura

```

660   DRAW SX-PEEK 23677,SY-
    PEEK 23678
670 NEXT I

```

680 REM deseneaza alte laturi

```

690 FOR I=1 TO 2

```

```

700   READ X,Y,Z

```

```

    GO SUB 820

```

```

    PLOT SX,SY

```

```

710   READ X,Y,Z

```

```

    GO SUB 820

```

```

    DRAW SX-PEEK 23677,SY-

```

```

    PEEK 23678

```

```

720 NEXT I

```

```

730 RETURN

```

```

800 REM "

```

rutina de reprezentare in
perspectiva prin proiectie
paralela "

```

810 REM

```

transformarea punctuala
de perspectiva

```

820 LET XE=-X*S1+Y*C1

```

```

830 LET YE=-X*C1*C2-Y*S1*C2+Z*S2

```

```

840 LET ZE=-X*S2*C1-Y*S1*S2-Z*C

```

```

2+RHO

```

```

850 REM

```

proiectia pe planul ecranului
si translatia la centrul sau

```

860 LET SX=D*XE/ZE+XOFF

```

```

870 LET SY=D*YE/ZE+YOFF

```

```

880 RETURN

```

```

900 REM

```

coordonatele carteziene
ale colturilor cubului

```

910 DATA 1,1,1,1,-1,1,-1,-1,1,1,1

```

```

920 DATA 1,1,1,1,-1,1,-1,-1

```

```

930 DATA -1,-1,-1,-1,1,-1,1,1,-1

```

```

940 DATA 1,-1,1,1,-1,-1,-1,1

```

```

950 DATA -1,-1,-1,-1,1,1,-1,1,-1

```

```

960 REM

```

```

9999 SAVE "MCUB" LINE 1

```

PROGRAMUL 8

```

1 > BORDER 1

```

```

    PAPER 1

```

```

    INK 7

```

```

    CLS

```

```

    POKE 23609,30

```

```

100 REM

```

programul deseneaza partile vizi-
bile ale unui cub, facind o re-
prezentare in perspectiva prin
proiectie paralela

```

110 REM

```

parametrii translatiei ce muta

originea sistemului de axe la
centrul ecranului

```

120 LET XOFF=128

```

```

    LET YOFF=86

```

```

130 REM

```

parametrii omotetiei aplicate
figurii pentru a umple cit mai
bine suprafata de afisare (cu-
bul desenat are latura unita-
ra)

```

140 LET XSCAL=40

```

```

    LET YSCAL=40

```



```

150 REM "
parametrii punctului de observare sînt dati in
sistemul de coordonate sferice pentru a simpli-
fica formulele.
RHO distanta intre originea sistemului de axe
si punctul de observare.
THETA, PHI unghiurile facute de directia de
observare cu axele D distanta dintre planul
ecranului si punctul de observare"
160 LET RHO=10
LET D=15
170 LET THETA=.7
LET S1=SIN THETA
LET C1=COS THETA
180 LET PHI=1.3
LET S2=SIN PHI
LET C2=COS PHI
190 REM
deseneaza dreptunghiul de incadrare
200 PLOT 0,0
DRAW 254,0
DRAW 0,175
DRAW -254,0
DRAW 0,-175
210 REM
citeste datele si deseneaza cubul
220 READ X,Y,Z
REM
X,Y,Z coordonatele colturilor cubului
230 GO SUB 380
REM
rutina ce realizeaza transformarea de perspectiva
240 PLOT SX,SY
REM deseneaza primul colt
250 REM
deseneaza laturile vizibile ale cubului
260 FOR I=1 TO 8
270 READ X,Y,Z
280 GO SUB 380
REM
rutina care realizeaza transformarea de pers-
pectiva
290 DRAW SX-PEEK 23677,SY-PEEK
23678

```

```

REM
deseneaza latura cubului
300 NEXT I
310 REM
deseneaza ultima latura vizibila a cubului
320 READ X,Y,Z
GO SUB 380
PLOT SX,SY
330 READ X,Y,Z
GO SUB 380
DRAW SX-PEEK 23677,SY-PEEK
23678
340 PRINT #1;AT 1,0;" APASATI 0
TASTA PT. CONTINUARE"
PAUSE 0
350 STOP
360 REM "
rutina ce realizeaza proiectia in perspectiva a
unei figuri tridimensionale"
370 REM
transformarea punctuala tridimensionala
380 LET XE=-X*S1+Y*C1
390 LET YE=-X*C1*C2-Y*S1*C2+Z*S2
400 LET ZE=-X*S2*C1-Y*S1*S2-Z*C
2+RHO
410 REM
proiectia punctului pe planul ecranului
420 LET SX=D*XE/ZE
430 LET SY=D*YE/ZE
440 REM
translatia ce pune figura desenata la centrul
ecranului
450 LET SX=XOFF+XSCAL*SX
460 LET SY=YOFF+YSCAL*SY
470 RETURN
480 REM
laturile vizibile ale cubului date prin coor-
donatele colturilor lor
490 DATA 1,1,0, 1,1,1, 0,1,1, 0,
1,0
500 DATA 1,1,0, 1,0,0, 1,0,1, 0,
0,1
510 DATA 0,1,1, 1,0,1, 1,1,1,
520 REM
9999 SAVE "CUB" LINE 1

```

PROGRAMUL 9

```

1 GO TO 500
5 DRAW x-PEEK 23677, y-PEEK 23
678: RETURN
10 LET vp=vp-1
20 IF vp=0 THEN LET vp=1:
RETURN
30 GO SUB 10: LET x=x+h: LET y
=y-h: GO SUB 5
40 GO SUB 100: LET x=x+2*h: GO
SUB 5
50 GO SUB 300: LET x=x+h: LET
y=y+h: GO SUB 5

```

```

60 GO SUB 10
70 LET vp=vp+1: RETURN
100 LET vp=vp-1
110 IF vp=0 THEN LET vp=1:
RETURN
120 GO SUB 100: LET x=x-h: LET
y=y-h: GO SUB 5
130 GO SUB 200: LET y=y-2*h: GO
SUB 5
140 GO SUB 10: LET x=x+h: LET y
=y-h: GO SUB 5
150 GO SUB 100

```



```

160 LET vp=vp+1: RETURN
200 LET vp=vp-1
210 IF vp=0 THEN LET vp=1:
RETURN
220 GO SUB 200: LET x=x-h: LET
y=y+h: GO SUB 5
230 GO SUB 300: LET x=x-2*h: GO
SUB 5
240 GO SUB 100: LET x=x-h: LET
y=y-h: GO SUB 5
250 GO SUB 200
260 LET vp=vp+1: RETURN
300 LET vp=vp-1
310 IF vp=0 THEN LET vp=1:
RETURN
320 GO SUB 300: LET x=x+h: LET
y=y+h: GO SUB 5
330 GO SUB 10: LET y=y+2*h: GO
SUB 5
340 GO SUB 200: LET x=x-h: LET
y=y+h: GO SUB 5

```

```

350 GO SUB 300
360 LET vp=vp+1: RETURN
500 INPUT n
600 LET i=0: LET h=32: LET x0=3
.2*h: LET y0=3*h
610 LET i=i+1: LET x0=x0-h
620 LET h=h/2: LET y0=y0+h
630 LET x=x0: LET y=y0: PLOT x, y
640 LET vp=i+1
660 GO SUB 10: LET x=x+h: LET y
=y-h: GO SUB 5
670 GO SUB 100: LET x=x-h: LET
y=y-h: GO SUB 5
680 GO SUB 200: LET x=x-h: LET
y=y+h: GO SUB 5
690 GO SUB 300: LET x=x+h: LET
=y+h: GO SUB 5
y695 PAUSE 0: CLS
700 IF i<n THEN GO TO 610
710 PAUSE 0
720 STOP

```

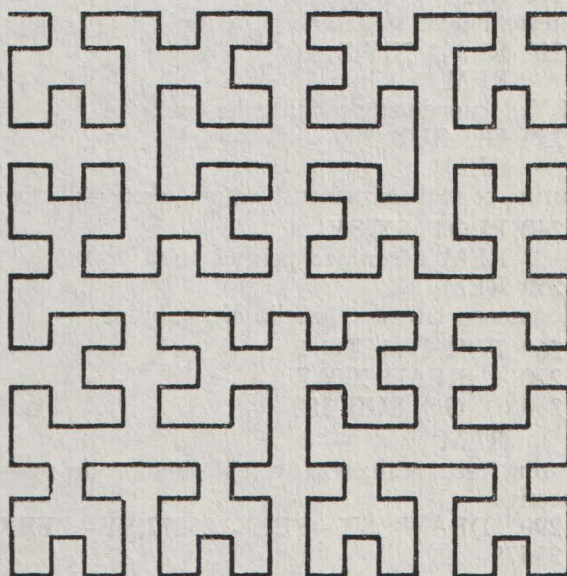
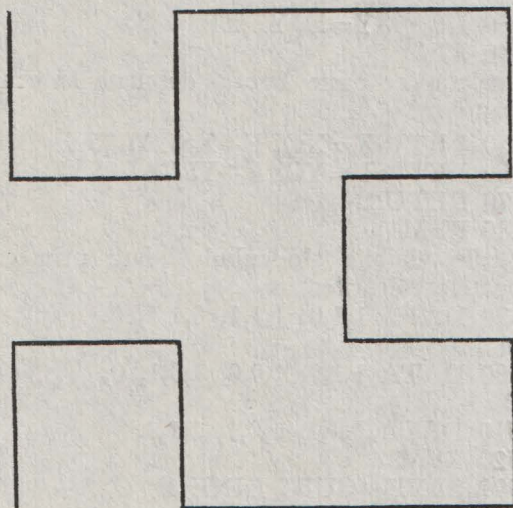


Fig. 16 Imagini oferite de programul 9

PROGRAMUL 10

```

1 GO TO 500
5 DRAW x-PEEK 23677,y-PEEK 23
678: RETURN
10 LET vp=vp-1
20 IF vp=0 THEN LET vp=1:
RETURN
30 GO SUB 300: LET x=x-h:
GO SUB 5
40 GO SUB 10: LET y=y-h:
GO SUB 5

```

```

50 GO SUB 10: LET x=x+h: GO SUB 5
60 GO SUB 100
70 LET vp=vp+1: RETURN
100 LET vp=vp-1
110 IF vp=0 THEN LET vp=1:
RETURN
120 GO SUB 200: LET y=y+h:
GO SUB 5
130 GO SUB 100: LET x=x+h:
GO SUB 5

```



```

140 GO SUB 100: LET y=y-h: GO SUB 5
150 GO SUB 10
160 LET vp=vp+1: RETURN
200 LET vp=vp-1
210 IF vp=0 THEN LET vp=1: RETURN
220 GO SUB 100: LET x=x+h: GO SUB 5
230 GOSUB 200: LET y=y+h: GO SUB 5
240 GO SUB 200: LET x=x-h: GO SUB 5
250 GO SUB 300
260 LET vp=vp+1: RETURN
300 LET vp=vp-1
310 IF vp=0 THEN LET vp=1: RETURN
320 GO SUB 10: LET y=y-h: GO SUB 5
330 GO SUB 300: LET x=x-h: GO SUB 5

```

```

340 GO SUB 300: LET y=y+h: GO SUB 5
350 GO SUB 200
360 LET vp=vp+1: RETURN
500 INPUT n
600 LET i=0: LET h=124: LET x0=
.8*h+90: LET y0=h/2+90
610 LET i=i+1: LET h=h/2
620 LET x0=x0+h/2: LET x0=y0+h/2
630 LET x=x0: LET y=y0: PLOT x, y
640 LET vp=i+1
660 GO SUB 10
665 PAUSE 0: CLS
670 IF i<n THEN GO TO 610
680 PAUSE 0
690 STOP

```

PROGRAMUL 11

1 REM PROGRAM REALIZAT DE STAN MUGUREL; liceul "RADU GRECEANU" ;SLATINA; SACAREA CRISTIAN; liceu 1 ind. nr. 1, REGHIN; APRODU MARIAN; liceul de chimie, CORABIA; 1987

3 BORDER 0: INK 7: PAPER 0: CLS

5 POKE 23609,10

10 LET t\$="PROGRAM DE REPREZENTARE SPATIALA "+CHR\$ 13+" A FUNCTIONILOR DE DOUA VARIABLE"

30 FOR i=1 TO LEN t\$: PRINT INVERSE 1;t\$(i); INVERSE 0;: BEEP .03,INT (32*RND): PRINT CHR\$ 8;: PRINT t\$(i);: NEXT i: INPUT ;:

PRINT #1; INK 5; "Apasati o tasta ptr. continuare."

40 LET j=1+INT (6*RND): BEEP .01,4*j: FOR i=22528 TO 22528+7: POKE i,8+j: IF INKEY\$ < ">" THEN CLS: GO TO 60

50 NEXT i: GO TO 40

60>CLS: PRINT: PRINT: PRINT "Programul functioneaza pe baza definirii unei functii: " ' ' ' f(x,y): (a,b) X (c,d) > R ' ' ' " Functia si valorile capetelor intervalelor se introduc de la tastatura " ' ' ' " ATENTIUNE a<b,c<d!!"

65 PRINT: PRINT "Directia de privire este definita prin doua unghiuri:"

66 PRINT: PRINT " Rotatia in planul [XoY]: PRINT: PRINT " Elevatia (unghiului fata de planul [XoY])"

67 PRINT: PRINT "Ambele unghiuri sint in grade.": PRINT #1;"Apasati o tasta ptr. continuare."

: BEEP 0.03,10: BEEP 0.01,5

70 PAUSE 0

80 INPUT "f(x,y)= "; LINE a\$

90>INPUT "a=" ;a;" b=" ;b:

INPUT "c=" ;c;" d=" ;d

100 INPUT "Rotatia :";alfa;" Elevatia :";beta: LET alfa=alfa/180*PI: LET beta=beta/180*PI

105 INPUT "Caroiaj rectangulare (r) Caroiaj polare (p) "; LINE d\$

106 IF d\$="p" THEN INPUT "Pozitia polului x=" ;polx; (CHR\$ 13); " y=" ;poly

110 IF a>=b OR c>=d THEN CLS: PRINT AT 10,2; "EROARE!!! Vezi instructiunile!": BEEP 0.05,20: PAUSE 100: CLS: GO TO 60

120 IF (ABS alfa>2*PI) OR (ABS (beta)>PI/2) THEN CLS: PRINT AT

10,2; "EROARE!!! Vezi instructiunile!": BEEP 0.05,20: PAUSE 100:

CLS: GO TO 60

125>LET uy=COS alfa: LET ux=COS (alfa+PI-beta): LET vx=SIN (alfa+PI-beta): LET vy=-SIN alfa: LET vz=COS beta

130 DIM u(31,31): DIM v(31,31): LET pasx=(b-a)/30: LET pasy=(d-c)/30

135 PRINT #1;"Asteptati :";: IF d\$="p" OR d\$="p" THEN GO TO 500

140 LET x=a: LET y=c: LET z=VAL a\$: LET umin=ux*x+uy*y: LET umax=umin: LET vmin=vz*z+vx*x+vy*y: LET vmax=vmin

160 FOR i=0 TO 30

170 FOR j=0 TO 30


```

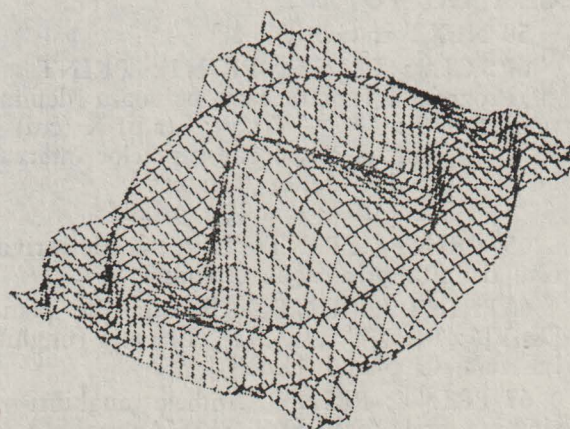
175 PRINT #1; AT 1,15; 372-INT ((i*30
+j)/2.5); " s ";
180 LET x=a+pasx*i: LET y=c+pasy*j:
LET z=VAL a$
190 LET uc=ux*x+uy*y
200 LET vc=vx*x+vy*y+vz*z
210 > IF vc <= vmin THEN LET vmin=vc
220 IF vc >= vmax THEN LET vmax=vc
230 IF uc <= umin THEN LET umin=uc
240 IF uc >= umax THEN LET umax=uc
250 LET u(i+1, j+1)=uc: LET v(i+
1, j+1)=vc: NEXT j: NEXT i: CLS
260 LET scu=230/(umax-umin): LET
scv=140/(vmax-vmin)
270 PLOT 0,0: DRAW 255,0: DRAW
0,175: DRAW -255,0: DRAW 0,-175
280 FOR i=1 TO 31 STEP 3: PLOT
(u(i, 1)-umin)*scu+14, (v(i,1)-vmin)*scv
+16: FOR j=2 TO 21
290 LET uc=u(i,j)-u(i,j-1): LET
vc=v(i,j)-v(i,j-1)
300 DRAW uc*scu, vc*scv: PLOT (u
(i,j)-umin)*scu+13, (v(i,j)-vmin)*scv+16
310 > NEXT j: NEXT i
320 FOR j=1 TO 31 STEP 3: PLOT (u(i,j)
-umin)*scu+13, (v(i,j)-vmin)*scv+16:
FOR i=2 TO 31
330 LET uc=u(i,j)-u(i-1,j): LET vc
=v(i,j)-v(i-1,j)
340 DRAW uc*scu, vc*scv: PLOT (u
(i,j)-umin)*scu+13, (v(i,j)-vmin)*scv+16
350 NEXT i: NEXT j
360 IF PEEK 23609 < > 10 THEN PRINT
#1; AT 1,0 "Apasati o tasta ptr.continua";
PAUSE 0: RUN
400 INPUT "Alta reprezentare?(d/n)";
LINE d$
405 IF PEEK 23609 < > 10 THEN
410 IF d$="n" OR d$="N" THEN STOP
420 INPUT "Aceasi functie?(d/n)"; LINE d$
430 IF d$="n" OR d$="N" THEN GO
TO 60
440 INPUT "Acelasi interval?(d/n)"; LINE d$
450 IF d$="n" OR d$="N" THEN GO
TO 90
460 GO TO 100
499 STOP

```

```

500 DIM r(4): LET r(1)=ABS (b-polx)
510 LET r(2)=ABS (d-poly)
520 LET r(3)=ABS (a-polx)
530 LET r(4)=ABS (c-poly)
540 LET rmax=r(1): LET rmin=r(1)
550 FOR i=2 TO 4
560 IF rmax < r(i) THEN LET rmax=r(i)
570 IF rmin > r(i) THEN LET rmin=r(i)
580 NEXT i
590 IF (polx > a) AND (polx < b) AND
(poly > c) AND (poly < d) THEN LET
rmin=SQR (pasx*pasx+pasy*pasy)/3
595 LET x=a: LET y=c: LET z=VAL a$:
LET umin=ux*x+uy*y: LET uma x
=umin: LET vmin=vz*z+vx*x+vy*y:
LET vmax=vmin
600 LET pasr=(rmax-rmin)/30
610 FOR i=0 TO 30
620 FOR j=0 TO 30
625 PRINT #1; AT 1,15; 372-INT ((i*30+j)
/2.5); "s ";
630 LET r=rmin+pasr*i
640 LET th=2*PI/30*j
650 LET x=polx+r*SIN th
660 LET y=poly+r*COS th
670 LET z=VAL a$
680 GO TO 190
900 BORDER 0: PAPER 0: OVER 0:
INVERSE 0: INK 7: BRIGHT 1: CLS
910 PRINT #1; " 3 D Graf - Demo ";
920 GO TO 260
1000 SAVE "3DGraf" LINE 900

```



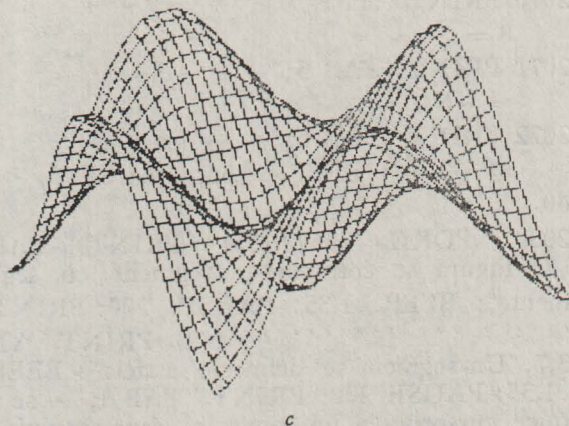
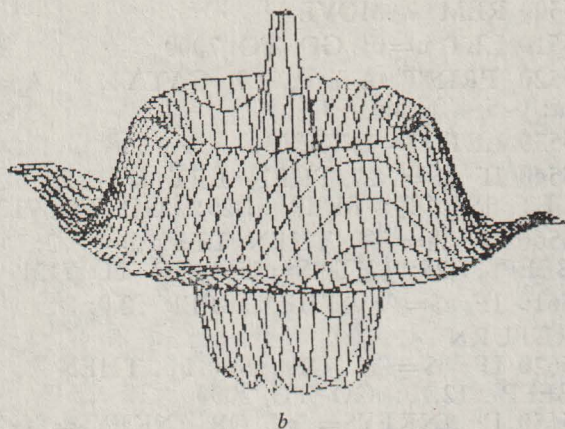


Fig. 17 a, b, c.
Imaginile aferente programului 11

PROGRAMUL 12

```

2 LOAD "DRAW" CODE 50000: LOAD
"DATA2"CODE: POKE 23562,1: POK E
23609,30
3 GO TO 1000
4
6 LET ex=0: INK 0
10 LET Xb=127: LET Yb=170
12 PLOT 2,2: DRAW 0,173: PLOT 0,166:
DRAW 2,7: DRAW 2,-7
13 PLOT 2,2: DRAW 198,0: PLOT 191,4:
DRAW 7,-2: DRAW -7,-2
14 PLOT 2,2: DRAW 121,121: PLOT
118,115: DRAW 4,7: DRAW -7,-4:
PRINT BRIGHT 1;AT 0,1; "?z?=0";A* T
21,26; "?x?=60";AT 6,10; "=60"
15 POKE 23606,1: PRINT AT 6,9; "?y?":
POKE 23606,0
17 DIM X(75): DIM Y(75): DIM Z (75):
GO SUB 9000
20 GO SUB 9500
80
90
1000 REM ETICHETA
1001
1002 PAPER 7: CLS: INK 0: BORDER 7
1020 FOR q=0 TO 9.5 STEP .8: FOR
x=0 TO 7: PLOT q*10,2.5*x*q: DRAW
0,8: BEEP .008,x*q: NEXT x: NEXT q
1025 FOR q=9.5 TO 0 STEP -.8: FOR
x=7 TO 0 STEP -1: PLOT 195-q*10,
x*q: DRAW -5,0: BEEP .008,x*q: NEXTx:
NEXT q
1026 FOR d=0 TO 7: FOR f=0 TO 7:
BORDER 7-f: BEEP .05,10-f*d: NEXT
f: NEXT d
1027 FOR a=19 TO 1 STEP -1: BEEP
.01,50-a: PRINT AT a,20-a; "SPACE":
NEXT a

```

```

1028 BORDER 2:
1029 FOR q=1 TO 17: PRINT AT g,24;
"0 ": BEEP .01,50-g*3: NEXT g
1030 PRINT AT 18,24;"K ";AT 19,24;"?!?"
1032 BORDER 7: BEEP .7,0: BEEP .7,4:
BEEP .7,7: BEEP 1,12: PAUSE 10:BEEP
.1,12: BEEP .1,7: BEEP .1,4: BEEP
.3,0: PAPER 7: PAUSE 35:INK 0:
BORDER 7
1199
1200 REM desen
1202 LET u=+.3: LET l$="8"
1204 LET ex=1
1210 FOR i=0 TO 8: GO SUB 7121:
BEEP .009,15: NEXT i: LET ex=0:
PAUSE 25
2000 RESTORE 2005: LET ad=USR "a":
FOR i=0 TO 47: READ a: POKE a
d+i,a: NEXT i
2005 DATA 8,28,62,127,28,28,28,28
2006 DATA 0,63,31,31,63,123,241,96
2007 DATA 2,7,143,222,252,248,24 8,252
2008 DATA 56,56,56,56,254,124,56,16
2009 DATA 0,8,12,254,255,254,12,8
2010 DATA 0,16,48,127,255,127,48,16
2020 CLS: PAPER 7: CLS: INK 0:
BORDER 7
2050 PRINT AT 0,10; FLASH 1;" IN
STRUCTIUNI "
2052 BEEP 2.5,25: PRINT AT 0,10;"?
INSTRUCTIUNI?"
2055 PAUSE 65: PRINT AT 3,7; "Dese
narea figurii se face cu ajutorul ?cursorului,
de coordonate initiale: 60,60,0.?": BEEP
.1,35:
PAUSE 300
2060 PRINT AT 7,7; "Pentru? deplasarea
cursorului? in cele 6 sensuri se utilizează
urmatoarele taste:?" : BEEP .1,35: PAUSE
300

```

* semnele de întrebare dintr-o linie cu PRINT reprezintă atribute care apar în acest fel la listare


```

2070 PRINT "TAB 5;" — ~0~
      8 = ~0~"
2071 PRINT "TAB 5;" — ~5~
      = ~8~"
2072 PRINT "TAB 5;" — ~6~
      = ~7~": BEEP .1,35: PAUSE 5
60
2080 POKE 23692,255: PRINT "TAB
7;"Figura se construiește segment cu seg-
ment.": BEEP .1,35: PAUSE 200: PRINT
".....": PRINT AT
3,7;"Un segment se definește astfel.": BEEP
.1,35: PAUSE 200: PRINT "TAB 8;" — se a
duce cursorul la un capăt al segmentului;"
2082 BEEP .07,40: PAUSE 15: BEEP
.12,10: PAUSE 200: PRINT "TAB 8;" —
se apasă ?tasta ~M~, pentru a fixa punctul
în memoria computerului — ?ATENȚIE! ?
dacă la apăsarea tastei nu s-a emis nici un
semnal sonor, se mai apasă pînă se aude
semnalul (care marchează memorarea pun-
ctului);"
2085 BEEP .06,40: PAUSE 15: BEEP .1,10:
PAUSE 570: BEEP .1,40: PAUSE 15: BEEP
.1,40: PAUSE 15: BEEP .1,40: PAUSE
18: PRINT "TAB 8;" ?NOTA: ? Dacă nu
se dorește unirea a două puncte consecutive,
înainte de tasta ~M~ se apăsata sta ~A~,
verificînd emiterea semnalului sonor pentru
fiecare tasta?."
2115 PAUSE 490: BEEP .07,40: PAUSE
15: BEEP .1,10: PRINT "TAB 8;" — pentru
a marca ?terminarea de senului? se apasă pe
?tasta ~R~ (verificînd emiterea semnalului
/sonor);"
2117 PAUSE 460: BEEP .07,40: PAUSE
15: BEEP .1,10: PAUSE 18: PRINT "TAB
8;" — pentru a roti figura (numai în jurul
axei verticale, ce trece prin centrul sau de
greutate) se așteaptă afisarea masesajului?
GATA! ? și apoi se apasă pe taste astfel:"
2118 PAUSE 440: PRINT "TAB 2;" ?* ?
?tasta ~5~()?...rotire în sens"; TAB 18;"
trigonometric": BEEP .1,40: PAUSE 120:
PRINT "TAB 2;" ?* ? ?tasta ~8~()?...
rotire în sens"; TAB 18;"invers"
2120 BEEP .1,40: PAUSE 100: BEEP
.1,10: PRINT "TAB 16;"*"; "TAB 13
;"*
2121 PAUSE 100: PRINT "TAB 7;"PENTRU
a ?genera un nou desen? se apasă ?tasta
~G~?": BEEP .1,40
2125 PAUSE 250: BEEP .1,25: PRIN
T "TAB 5;" ?p? PENTRU ?INSTRUCTI
UNI? SE APĂSA PE ?TASTA ~I~.?"
2130 PAUSE 250: PRINT #0; "TAB 10
;"
      SUCCES? ?! ?
      ": FOR i=1 TO 20: BEEP .09,i*2: NEXT i
2140 PAUSE 270
2500 CLS: GO TO 5

```

```

5500 REM ? MOVE ?
5510 LET u=0: GO TO 7000
5520 PRINT AT 0,0;" ? GATA! ? Apa
sa: ~5~() sau ~8~()"
5530 LET a$=INKEY$
5540 IF a$="8" THEN LET u=+.2:
LET l$="8": BEEP .1,25: GO TO 712
5560 IF a$="5" THEN LET u=-.2:
BEEP .1,25: LET l$="5": GO TO 7121
5610 IF a$="r" THEN BEEP .2,0:
RETURN
5620 IF a$="i" OR a$="I" THEN
BEEP .12,10: GO TO 2000
5650 IF INKEY$="g" OR INKEY$="G"
THEN BEEP .1,35: GO TO 2500
5700 GO TO 5520
5710
7000 REM ? ROTIRE PE ORIZONTALA
?
7003 LET ss=2: LET semn=1
7005 DIM i(nn,2): DIM o(nn,2)
7006 DIM d(nn)
7020 LET a=45
7025 LET a=PI/2-a/180*PI
7027 LET y1=0: LET x1=0: FOR f=1
TO f-1: LET y1=y1+y(f): LET x1=
x1+x(f): NEXT f: LET X1=x1/(f-1)
: LET y1=y1/(f-1)
7030
7032 FOR k=1 TO nn
7042 LET d(k)=60+z(k)
7045 LET x=x(k): LET y=y(k)
7050 LET m=x-x1: LET n=y-d(k)
7060 IF n=0 THEN LET n=1
7065
7070 LET cos a=COS a
7075 LET cos t=ABS (m*cos a*SR (
1/n*n+m*m*cos a*cos a) )
7080 LET R(k)=1/cos a*SQR (n*n+m*
m*cos a*cos a)
7090 LET sint=1-cos t*cos t
7100 LET S(k)=ASN (n/(R(k)*cos a))
7110 LET arc 0: GO SUB 7250
7115 LET S(k)=ASN (n/(R(k)*cos a))
+arc
7117 NEXT k: BEEP .02,0: BEEP .0
2,-2
7120 GO TO 5515
7121
7123 LET t=0
7125 LET add=59999: LET jp=60000
+(PEEK add+1)*2+1
7126 FOR k=1 TO nn
7127
7128 LET add=add+1
7129 IF add=jp THEN LET jp=add+(

```



```

PEEK jp+1)*2+1: LET add=add+1:
POKE add,gg: LET add=add+1
7130 LET S(k)=S(k)+u
7140 LET t=S(k)
7150 LET gg=x1+R(k)*COS t: LET h
h=d(k)+R(k)*cosa*SIN t
7165
7166 POKE add,gg: LET add=add+1
7167 IF add=jp THEN LET jp=add+(
PEEK jp+1)*2+1: LET add=add+1:
POKE add,gg: LET add=add+1
7168 POKE add,hh
7170 NEXT k
7185 LET v=ss
7215 CLS: RANDOMIZE USR 50000
7217 IF ex=1 THEN RETURN
7220 IF INKEY$ < > " " THEN GO TO 55
30
7225 STOP
7240 GO TO 7125
7250 IF m<0 AND n>0 THEN LET arc
=PI-2*S(k)
7260 IF m<0 AND n<0 THEN LET arc
=PI-2*S(k)
7270
7300 RETURN
9000
9010 REM ? INKEY$ ?
9015 LET ff=1: LET ctl=1: LET HL
=60000
9020 DIM a(75,2): LET ct=1: LET
aa=1: LET a(1,1)=2
9030 LET f=1: LET x=60: LET y=60
: LET z=0: PLOT x+y,y+z
9050
9070 PRINT AT 21,28; BRIGHT i;x;
BRIGHT 0; " "
9072 IF INKEY$="8" THEN LET x=x+1
9075 IF INKEY$ < > "8" THEN GO TO 9
090
9077 IF x+y>253 THEN GO TO 9075
9080 PLOT x+y,y+z: PLOT INVERSE
1;x+y-1,y+z: GO TO 9070
9090 PRINT AT 21,28; BRIGHT 1;x;
BRIGHT 0; " "
9092 IF INKEY$="5" THEN LET x=x-1
9095 IF INKEY$ < > "5" THEN GO TO 9
110
9097 IF x+y<3 THEN GO TO 9095
9100 PLOT x+y,y+z: PLOT INVERSE
1;x+y+1,y+z: GO TO 9090
9110 PRINT AT 6,11; BRIGHT 1;y;
BRIGHT 0; " "
9112 IF INKEY$="7" THEN LET y=y+1
9115 IF INKEY$ < > "7" THEN GO TO 9
130
9117 IF x+y>253 OR y+z>173 THEN
GO TO 9115
9120 PLOT x+y,y+z: PLOT INVERSE
1;x+y-1,y+z-1: GO TO 9110
9130 PRINT AT 6,11; BRIGHT 1;y;

```

```

BRIGHT 0; " "
9132 IF INKEY$="6" THEN LET y=y-1
9135 IF INKEY$ < > "6" THEN GO TO 9
150
9137 IF x+y<3 OR y+z<3 THEN GO TO
9135
9140 PLOT x+y,y+z: PLOT INVERSE
1;x+y+1,y+z+1: GO TO 9130
9150 PRINT AT 0,3; BRIGHT 1;z;
BRIGHT 0; " "
9152 IF INKEY$="0" THEN LET z=z+1
9155 IF INKEY$ < > "0" THEN GO TO 9
170
9157 IF y+z>171 THEN GO TO 9155
9159 PRINT AT 0,3; BRIGHT 1;z;
BRIGHT 0; " "
9160 PLOT x+y,y+z: PLOT INVERSE
1;x+y,y+z-1: GO TO 9150
9170 PRINT AT 0,3; BRIGHT 1;z;
BRIGHT 0; " "
9175 IF INKEY$="9" THEN LET z=z-1
9180 IF INKEY$ < > "9" THEN GO TO 9
200
9185 IF z<1 THEN GO TO 9180
9190 PLOT x+y,y+z: PLOT INVERSE
1;x+y,y+z+1: GO TO 9170
9200
9210 IF (INKEY$="m" OR INKEY$ "M
") AND ctl>0 THEN LET ctl=ctl-1
9220 IF INKEY$="m" OR INKEY$="M"
THEN LET x(f)=x+y: BEEP .2,30:
LET y(f)=y+z: LET z(f)=z: POKE H
L,X(F): POKE HL+1,Y(F): LET L=H
L+2: POKE (hl-2*FF-1),FF-1: LET
f=f+1: LET ff=ff+1: FOR t=1 TO 5
0: NEXT t
9260 IF INKEY$="r" OR INKEY$="R"
THEN LET nn=f-1: DIM R(nn): DIM
S(nn): INK 0: BEEP .1,40: LET a
(aa,2)=f: POKE 50007,aa: LET a(a
a+1,1)=f+1: LET n=a(aa,2): GO TO
5500
9270 IF INKEY$="a" OR INKEY$="A"
THEN LET hl=hl+2: LET ff=1: LET
ctl=2: BEEP .1,10: BEEP .1,2: LET
ct=1: LET a(aa,2)=f: LET a(aa+1,1)=f
+1: LET aa=aa+1: FOR t=1 TO 40:
NEXT t
9275
9276 IF INKEY$="i" OR INKEY$="I"
THEN GO TO 2000
9280 IF ctl<0 THEN LET ctl=2
9295>IF ctl=1 OR f<3 THEN PLOT x
(f),y(f): GO TO 9070
9297 IF ctl=0 THEN GO SUB 9320
9298 IF INKEY$="g" OR INKEY$="G"
THEN BEEP .1,35: GO TO 2500
9300 GO TO 9070
9301
9310 REM ? DRAW ?
9320 PLOT x(f-1),y(f-1)

```



```
9330 DRAW -x(f-1)+x(f-2), -y(f-1)
+y(f-2)
9350 RETURN
```

	USR 50000 *
50000	217 exx
50001	229 push hl
50002	217 exx
50003	33,95,234 ld hl,59999
50006	6,1 ld b,1
50008	197 push bc
50009	70 ld b,(hl)
50010	35 inc hl
50011	197 push bc
50012	78 ld c,(hl)
50013	35 inc hl
50014	70 ld b,(hl)
50015	35 inc hl
50016	94 ld e,(hl)
50017	35 inc hl
50018	86 ld d,(hl)
50019	35 inc hl
50020	237,67,125,92 ld (23677),bc
50024	229 push hl
50025	33,233,253 ld hl,65001
50028	205,146,195 call 50066
50031	43 dec hl
50032	43 dec hl
50033	43 dec hl
50034	65 ld b,c
50035	83 ld d,e
50036	205,146,195 call 50066
50039	237,75,232,253 ld bc,(65000)
50043	237,91,234,253 ld de,(65002)
50047	205,186,36 call 9402
50050	225 pop hl
50051	43 dec hl
50052	43 dec hl
50053	193 pop bc
50054	16,211 djnz,211
50056	35 inc hl
50057	35 inc hl
50058	35 inc hl
50059	193 pop bc
50060	16,202 djnz,202
50062	217 exx
50063	225 pop hl
50064	217 exx
50065	201 ret
50066	122 ld a,d

* rutina în cod mașină

50067	184	cp b
50068	56,11	jr c,11
50070	184	cp b
50071	40,14	jr z,14
50073	144	sub b
50074	22,1	ld d,1
50076	119	ld (hl),a
50077	35	inc hl
50078	35	inc hl
50079	114	ld (hl),d
50080	201	ret
50081	120	ld a,b
50082	146	sub d
50083	22,255	ld d,255
50085	24,245	jr 245
50087	62,0	ld a,o
50089	22,0	ld d,o
50091	24,239	jr 239
50093	22,0	ld d,o
50095	24,239	jr 239
50097	0	nop

6.2. Instruire asistată de calculator

Se dau spre exemplificare două programe realizate cu BETA BASIC. Extensia oferă numeroase facilități, dînd astfel posibilitatea concentrării efortului mai mult spre redarea cunoștințelor decît spre "artificii" de programare necesitate de programe adecvate domeniului instruirii asistate de calculator.

Tematica a fost aleasă din domeniul fizicii în care se pot simula cu ajutorul calculatorului experiențe și fenomene.

Programul 13, RLC, vine în ajutorul profesorilor de liceu, în predarea fizicii, capitolul "Legile curentului alternativ" (clasa a X-a). După prezentarea noțiunilor teoretice se propune rezolvarea cîtorva probleme.

Programul 14, "OPTIC", permite exemplificarea unor experiențe de optică prin care se evidențiază modul de formare a imaginilor în oglinzi concave și lentile biconvexe în condiții diverse (distanța focală, distanța lentilă obiect etc.).

Programele au fost realizate de elevii: Ganea Cătălin, Toca Dorel, Kiss Ferenc, Mureșan Horațiu, Doneanu Rodica, Floarea Radu, Iakob Tibor, Ritter Csaba sub îndrumarea cerc. șt. Diamandi Ion și a prof. Boeriu Romulus.

PROGRAMUL 13

```
10 PLOT 5 ,171; "CALCULUL RLC"
20 LET dat=180
30 DEF PROC menu pr, REF optiuni
40   LOCAL cim,g$
50   LET cim= DPEEK(DPEEK(2361
3)+2) +7429
60   RESTORE dat
70   LET ii=163
80   DO
```

```
LET ii=ii-12
READ LINE a$
PLOT 20,ii;a$(1 TO 2)
PLOT INVERSE 1; PAPER 0
; INK 7;32,ii;a$(3 TO )
LOOP UNTIL ITEM( )=0
90   DPOKE 23639,cim-1
100  GET g$
LET g$=SHIFT$(1,g$)
LET optiuni=0
```



```

110 RESTORE dat
120 READ LINE a$
    LET optiuni=optiuni+1
    IF a$(1)=g$ THEN GO TO 140
130 IF ITEM( )=0 THEN GO TO 90
    ELSE GO TO 120
140 IF INKEY$ < > " " THEN GO TO
140
150 CLS
160 END PROC
170 menu 0,optiuni
180 DATA 1-TEORIE,2-RLC- PARALE
L,3-PROBLEMA 1,4-PROBLEMA 2
190 GO TO ON optiuni;330,340,355,360
330 PROC TEORIE
    GO TO 1
340 PROC PARALEL
    GO TO 1
355 PROC PROBLEME
    GO TO 1
360 prob2
    GO TO 1
500 DEF PROC TEORIE
510 PRINT CSIZE 8,16;"*TEORIE
    - CURENTUL ALTERNATIV*"
520 CSIZE 8 PRINT AT 4,0;" PE BAZA
MANUALULUI CLASEI a X-a LEGILE
CURENTULUI ALTERNATIV SINT:"
530 KEYWORDS 0
    PLOT 30,110;"I =I DPOKE 2 "
540 PLOT OVER 1;70,110;" EXIT IF "
    KEYWORDS 1
550 PLOT 38,108;"m"
560 KEYWORDS 0
    PLOT 30,100;"U =U DPOKE 2 "
570 PLOT OVER 1;70,100;" EXIT IF "
    KEYWORDS 1
580 PLOT 38,98;"m"
600 PLOT 56,54;"m"
610 PLOT 56,22;"m"
620 PRINT AT 21,27; INK 2; ">"
; INK 6; ">"; INK 4; ">"; INK 1; ">"
630 GET K$
640 PRINT AT 21,27;" "
    LET K=1
    DO
        SCROLL 12,4;0,160;32,160
        BEEP K/1000,K+5
        EXIT IF K=64
        LET K=K+1
    LOOP
650 PRINT AT 0,0;"C - ESTE CA
PACITATEA CONDENSATORULUI (i
nROLL F Z - REA

```

```

CTANTA INDUCTIVA(in USING)
L
- INDUCTANTA BOBINEI(in mH)
R
- REZISTENTAREZISTORULUI(USING)
660 PRINT AT 21,27; INK 2;">"
; INK 6;">"; INK 4;">"; INK 1;">"
670 GET K$
680 PRINT AT 21,27;" "
    LET K=1
    DO
        SCROLL 12,4;0,160;32,160
        BEEP K/1000,K+5
        EXIT IF K=64
        LET K=K+1
    LOOP
685 CLS
690 END PROC
700 DEF PROC paralel
705 PRINT AT 0,5; CSIZE 8,16;
"*CIRCUIT - PARALEL*"
710 PRINT AT 3,0; "CIRCUITUL
CONTINE: - REZISTENTA R
    - BOBINA CU
    INDUCTANTA L
    - CONDENSATOR
    CU CAPACITATEA C"
720 KEYWORDS 0
    PRINT AT 8,0; "LA O TENSIU
NE ALTERNATIVA u=DPOKE
2U sin ON ERROR t"
730 PRINT OVER 1;AT 9,6;" EXIT IF "
    PROC RAJZ
    GO TO 805
734 DEF PROC RAJZ
735 KEYWORDS 0
    PLOT 120,80
    A
740 DEF PROC A
    DRAW 0,8
    DRAW 32,0
    DRAW 0,-8
    DRAW -32,0
    END PROC
750 PLOT 96,84
    DRAW 24,0
    PLOT 152,84
    DRAW 24,0
    DRAW 0,-32
    DRAW -24,0
    PLOT 96,84
    DRAW 0,-32

```



```

DRAW 24,0
PLOT 120,48
PROC A
PLOT 96,52
DRAW 0,-32
DRAW 32,0
PLOT 176,52
DRAW 0,-32
DRAW -32,0
760 PLOT 96,20
DRAW 0,-16
DRAW 30,0
PLOT 176,20
DRAW 0,-16
DRAW -30,0
770 PLOT 144,14
DRAW 0,12
PLOT 143,14
DRAW 0,12
PLOT 130,14
DRAW 0,12
PLOT 129,14
DRAW 0,12
PLOT 110,86;" UNTIL "
PLOT 110,54;" UNTIL "
PLOT 110,22;" UNTIL "
FILL 122,50
780 PLOT 132,96;"R"
PLOT 132,74;"L"
PLOT 133,35;"C"
CIRCLE 128,4,2
CIRCLE 144,4,2
790 PLOT 99,83;"i"
PLOT CSIZE 5;105,80;"R"
PLOT 99,51;"i"
PLOT CSIZE 5;105,48;"L"
PLOT 99,19;"i"
PLOT CSIZE 5;105,16;"C"
PLOT 94,14;"POP"
PLOT 86,14;"i"
800 KEYWORDS 1
803 END PROC
805 PRINT AT 21,27; INK 2;">"
; INK 6;">"; INK 4;">"; INK 1;">"
810 GET K$
815 KEYWORDS 0
820 PRINT AT 21,27;" "
LET K=1
DO
    SCROLL 12,4;0,160;32,160

    BEEP K/1000,K+5
    EXIT IF K=64
    LET K=K+Y1
    LOOP
830 PLOT 10,145;"i= sin ON
ERROR t"
PLOT CSIZE 5;18,140;"R"
PLOT 34,150;"DPOKE 2U"
PLOT OVER 1;42,15;"EXIT"
IF "
PLOT 42,138;"R"

```

```

PLOT 35,141
DRAW 24,0
PLOT 10,115;"i= sin(ON
ERROR t-PI/2)"
PLOT CSIZE 5;18,110;"L"
PLOT 34,120;" DPOKE 2U"
PLOT OVER 1;42,120;"EXIT"
IF "
PLOT 42,108;"X"
PLOT CSIZE 5;49,105;"L"
PLOT 35,111
DRAW 24,0
840 PLOT 10,85;"i= sin( ON
ERROR t+PI/2)"
PLOT CSIZE 5;18,80;"C"
PLOT 34,90;" DPOKE 2U"
PLOT OVER 1;42,90;" EXIT"
IF "
PLOT 42,78;"X"
PLOT CSIZE 5;49,75;"C"
PLOT 35,81
DRAW 24,0
850 PLOT 10,65;"tg ON =R( ON
ERROR C-1/L ON ERROR)"
860 PLOT 10,45;"I=U DPOKE (1/
R)^2+(1/Xc-1/X1)^2"
870 PLOT 42,45
DRAW 170,0
880 PLOT 10,20;"Z=1/(DPOKE (
1/R)^2+(1/Xc-1/X1)^2)"
PLOT 58,20
DRAW 168,0
900 PRINT #1;"APASATI O TASTA
PT. CONTINUARE"
PAUSE 0
CLS
910 PLOT CSIZE 8,16;10,170;"IN
CAZUL REZONANTIEI : "
920 PLOT 10,140;"Irez=0"
PLOT 10,120;"Zrez WHILE
SCROLL DO"
PLOT 10,100;"X1=Xc"
930 PRINT #1;"APASATI O TASTA
PT. CONTINUARE"
PAUSE 0
CLS
940 KEYWORDS 1
998 END PROC
999 STOP
1000 DEF PROC PROBLEME
1001 CLS
1005 RAJZ
1010 FOR z=1 TO 65
    SCROLL 11
    NEXT z
    FOR n=1 TO 70
        SCROLL 12
        NEXT n
1015 KEYWORDS 0
1016 PRINT CSIZE 8,16;AT 0,3;"
*PROBLEME -1-*"
1020 INPUT "SCRIETI: U=";U;" R
=";R;" L=";L;" C=";C;" ELSE =" ;v
1030 PLOT 30,140;"U=" +STR$ U+"V"

```



```

1040 PLOT 30,130; "R=" +STR$ R +
      USING "
1050 PLOT 30,120; "L=" +STR$ L +
      mH"
1060 PLOT 30,110; "C=" +STR$ C +
      ROLL F"
1070 PLOT 30,100; "ELSE=" +STR$
      v + "Hz"
1080 PLOT 0,90; " - - - - -
      - - - - -"
1085 LET c=c/1000000
      LET l=1/1000
1090 LET w=2*PI*v
      LET x1=w*1
      LET xc=1/(w*c)
      LET i=u*SQR (1/(r*r)+(1/ xc-1/
x1)*(1/xc-1/x1)))
1100 LET z=u/r
      LET f1=ATN (r*(xc-1/x1))
      LET f=f1*180/PI
      LET p=INT ((f-INT f)*60))
1120 PRINT #1;"APASATI 0 TASTA
      PT. REZULTAT"
1125 LET F1=INT (F*100)
      IF F1=0 THEN PROC REZ
          GO TO 1200
1130 PAUSE 0
1140 PRINT AT 12,4;"I="; USING
      "#####.##";I;"A"
1150 PRINT AT 14,4;"Z="; USING
      "#####.##";Z;"USING"
1160 PRINT AT 16,4;"ON="; US
      ING "###";f;"BLANK";p;"' "
1165 KEYWORDS 1
1200 INPUT "MAI VRETI PROBLEME
      (d/n)";a$
1210 IF a$="d" OR a$="D" THEN
      CLS
          GO TO 355
1230 IF a$="n" OR a$="N" THEN
      CLS
          END PROC
1240 GO TO 1210
2000 DEF PROC REZ
2005 >KEYWORDS 0
2010 PRINT AT 12,4;"I=";0
2020 PRINT AT 14,4;"Z= SCROLL DO"
2030 PRINT AT 16,4;" ON =0"
2040 PRINT CSIZE 8,16;AT 9,0;"**
      -CIRCUITUL ESTE IN REZONANTA**"
2045 KEYWORDS 1
2050 END PROC
3000 >DEF PROC prob2
3001 CLS
3005 rajz
3010 FOR z=1 TO 65
          SCROLL 11
          NEXT z
          FOR n=1 TO 70
              SCROLL 12
              NEXT n
3015 KEYWORDS 0

```

```

3020 PRINT CSIZE 8,16;AT 0,3;"
      *PROBLEME -2-"
3030 INPUT "u=SQR 2*UUU*SIN ON
      ERROR ON ERROR ON ERROR *ttt;u=
      ";u$
3040 LET u=VAL u$(4 TO 6)
3050 LET w=VAL u$(9 TO 11)
3060 LET t=VAL u$(13 TO 15)
3070 INPUT "R=";R;" L=";L;" C=
      ";C
3075 PLOT 0,151;"u= DPOKE 2*" +
      STR$ u + "SIN" + " " + "*" +STR$ t
          PLOT OVER 1;24,152;" EXIT
      IF "
3077 PRINT AT 3,11; USING "###"
      ";w
3080 PLOT 30,140;"U=" +STR$ U +
      V"
3090 PLOT 30, 130;"R=" +STR$ R +
      USING "
3100 PLOT 30,120;"L=" +STR$ L +
      mH"
3105 PLOT 30,110;"C=" +STR$ C +
      ROLL F"
3110 LET v=w/6.28
      IF v < > INT v THEN LET v=((
      INT (v*100) ) / 100)+0.005
3120 PLOT 30,100;"ELSE=" +STR$ v
      + "Hz"
3130 PLOT 0,90; " - - - - -
      - - - - -"
3140 LET w=2*PI*v
      LET x1=w*1
      LET xc=1/(w*c)
      LET i=u*SQR (1/(r*r)+(1/
xc-1/x1)*(1/xc-1/x1)))
3150 LET z=u/r
      LET f=ATN (r*(xc-1/x1))
      LET p=INT ((f-INT f)*60))
3160 PRINT #1;"APASATI 0 TASTA
      PT. REZULTAT"
3170 LET F1=INT (F*100)
      IN F1=0 THEN PROC REZ
          GO TO 1200
3180 PAUSE 0
3190 PRINT AT 12,4;"I="; USING
      "#####.##";"A"
3200 PRINT AT 14,4;"Z="; USING
      "#####.##";Z;"USING"
3210 PRINT AT 16,4;" ON ="; US
      ING "###";f;"BLANK";p;"' "
3220 KEYWORDS 1
3230 INPUT "MAI VRETI PROBLEME
      (d/n)";a$
3240 IF a$="d" OR a$="D" THEN
      CLS
          GO TO 360
3250 IF a$="n" OR a$="N" THEN
      CLS
          END PROC
3999 STOP
4000 FOR n=0 TO 7

```



```

      INPUT a
      POKE USR "d" + n, a
NEXT n
8999 STOP
9000 LET rt = PEEK 23730 + 256 * PEEK
23731
      SAVE "PINK Beta" LINE 9010
      POKE PEEK 23631 + 256 * PEEK 23
632 + 2, 181
      SAVE "PINKSTAR CODE" "CODE r
t + 1, 65536 - rt
      STOP
9010 POKE PEEK 23613 + 256 * PEEK 23
614, 0
      BORDER 0
      PAPER 0
      INK 7
      CLEAR rt
      LOAD "PINKSTAR CODE" "CODE
CLS
9020 POKE 23606, 112
      POKE 23607, 179
      RANDOMIZE USR 46962
      RANDOMIZE USR 58419
      PRINT CSIZE 16; AT 0, 0; "Press
any key !"
9030 PAUSE 0
9040 PRINT CSIZE 16; AT 0, 0; " cha
racter sets "; AT 2, 4; "(1-5) ?"
9050 GET a
9060 IF a < 1 OR a > 5 THEN GO TO 92
50
9070 IF a = 1 THEN DPOKE 23606, 153
60
      CLEAR 46959
      GO TO 9370
9080 CLS
      PRINT "Select:"
9090 LET s$ = " "
      FOR n = 32 TO 127
          LET s$ = s$ + CHR$ n
      NEXT n
9100 DIM A$(4, 768)
9110 FOR n = 1 TO 4
9120     LET m = 46960 - n * 768
9130     LET A$(n) = MEMORY$(m TO
m + 767)
9140     DPOKE 23606, 46704 - 768 * n
9150     PRINT TAB 15; CHR$ (64 + n)

9160     PRINT s$
9170 NEXT n
9180 FOR n = 1 TO a - 1
9190     PRINT OVER 1; AT 0, 0; "      "
9200     GET b
9210     IF b < 10 OR b > 13 THEN GO TO
9190
9220     PRINT OVER 1; AT 0, 0; "      "
9230     POKE 46960 - 768 * n, a$(b - 9)
9240 NEXT n
9250 CLEAR 47727 - a * 768
9260 DEF KEY "1"

```

```

      DPOKE 23606, 15360
9270 DEF KEY "2"
      DPOKE 23606, 45936
9280 IF DPEEK(23730) < 46000 THEN
DEF KEY "3"
      DPOKE 23606, 45168
9290 IF DPEEK(23730) < 45300 THEN
DEF KEY "4"
      DPOKE 23606, 44400
9300 IF DPEEK(23730) < 44500 THEN
DEF KEY "5"
      DPOKE 23606, 43632
9310 DEF KEY "1"
      CLS
      PRINT AT 10, 10; CSIZE 16; "
LOAD "
      LOAD " "
9320 DEF KEY "n"
      CLS
      PRINT AT 10, 9; CSIZE 16; " M
ERGE "
      MERGE " "
9330 DEF KEY "i"
      CLS
      PRINT AT 0, 10; "INFORMATII"
; AT 2, 5; "DISPLAY FILE: 16384"; AT
3, 5; "ATTRIBUTES: 22528"; AT 4,
5; "PRINTER BUTT: 23296"; AT 5, 5; "
SYSTEM VAR.: 23552"; AT 6, 5; "MIC
RODRIVE: 23734"; AT 7, 5; "BASIC
PRG.: 23635"; AT 8, 5; "VARS S.VA
LT.: 23627"; AT 9, 5; "E-LINE.S.V.:
23641"; AT 10, 5; "RAM TOP:
23730"; AT 11, 5; "UDG GRAFIX.: 23
675"; AT 12, 5; "P-RAMT: 23732"
9340 DEF KEY "m"
      CLS
9350 DEF KEY "s"
      INPUT "SAVE LINE: "; sa; " TO
LINE: "; sb
      LET s$ = "LOAD " + STR$ sa + " TO
" + STR$ sb
      CLS
      PRINT AT 11, 1; "SAVE LINE: "; sa; "
TO LINE: "; sb
      SAVE sa TO sb; s$
      CLS
      PRINT AT 11, 1; "VERIFY LINE:
"; sa; " TO LINE: "; sb
      VERIFY sa TO sb; s$
9360 DEF KEY "r"
      INPUT "RENUM LINE: "; la; " TO
LINE: "; lb
      INPUT "NEW LINE: "; lin; " STEP
: "; ste
      RENUM (la TO lb) LINE lin STEP
ste
9370 DPOKE 23606, 15360
      DPOKE DPEEK (23613), 64259

```


PROGRAMUL 14

```

1>CLS
  INK 0
  PAPER 0
  CLS
  LOAD " "SCREEN$
  INK 7
5  PAUSE 0
6  CLS
10 PRINT AT 0,0; "*****
*****
***** "
11 PRINT AT 21,0; "*****
*****
***** "
13 PRINT AT 4,0; " "
14 PRINT AT 6,0; " "
15 PRINT AT 7,0; " "
16 PRINT AT 8,0; " "
25 PRINT AT 12,20; "FORMAREA IM
AGINILOR IN:"
30 PRINT AT 14,20; "a) [0]glinzi concave "
35 PRINT AT 16,20; "b) [L]entile bicon-
vexe "
100 GO TO 5000
1000 REM OGLINDA
1010 CLS
      PLOT 0,90
      DRAW 255,0
      FOR a=0 TO 255 STEP 5
        PLOT a,90
        DRAW 0,-1
      NEXT a
      PLOT 85,91
      PLOT 165,91
1020 PLOT 104,20
      DRAW 0,140,PI/3
      DRAW 4,0
      DRAW 0,-140,-PI/3
      DRAW -4,0
      FILL 1 06,22
      FILL 106,158
1030 PLOT 0,10
      DRAW 255,0
      DRAW 0,165
      DRAW -255,0
      DRAW 0,-165
1040 PRINT AT 0,0; "*****
FORMAREA IMAGINILOR IN OGLINZI
CONCAVE ***** "
1399 RETURN
1500 REM LENTILA
1510 CLS
      PLOT 0,90
      DRAW 255,0
      FOR a=0 TO 255 STEP 5
        PLOT a,90
        DRAW 0,-1
      NEXT a
      PLOT 85,91
      PLOT 165,91
1520 PLOT 125,20
      DRAW 0,140,PI/14
      DRAW 0,-140,PI/14
1530 PLOT 0,10
      DRAW 255,0
      DRAW 0,165
      DRAW -255,0
      DRAW 0,-165
1540 PRINT AT 0,0; "*****
FORMAREA IMAGINILOR IN LENTILE
BICONVEXE ***** "
1899 RETURN
5000 PAPER 0
      INK 7
5100 LET A$=INKEY$
5110 IF A$="0" THEN CLS
      GO TO 6000
5120 IF A$="1" THEN CLS
      GO TO 7000
5199 GO TO 5100
6000 REM SOFT 0
6010 GO SUB 1000
6020 PRINT #0; "Dist. obiect—ogli
nda [0 <x1 <120]"
      PAUSE 0
      INPUT #0;x1
      IF x1 < 0.1 OR x1 > 120 THEN GO
TO 6020
6023 IF x1=40 THEN PRINT AT 21,3
5; " "; FLASH 1
;AT 21,35; "Imagine la infinit "
6025 PLOT 125-x1,90
      DRAW 0,15
      DRAW -2,-3
      DRAW 4,0
      DRAW -2,3
6030 PRINT AT 21,0; " '=40 ";AT 2
1,10; " ";AT 21,7; "x1=";x1
6040 LET f=40
      ON ERROR 6100
      LET x2=(x1*f)/(x1-f)
      IF x2 < 0 THEN PRINT AT 21,
48; "Imagine virtuala "
      GO TO 6060
6050 PRINT AT 21,48; "Imagine reala "
6060 PRINT AT 21,18; " "
;AT 21,15; "x2=";ABS x2
6070 LET y1=15
      PRINT AT 21,28; "y1=";y1
      LET y2=-y1*x2/x1
      PRINT AT 21,36; "y2=";ABS y2
      PLOT 125-x2,90
      IF x2 > 125 THEN GO TO 6999
6080 PLOT 125-x2,90
      DRAW 0,y2
      DRAW -2, (SGN x2)*3
      DRAW 4,0

```



```

DRAW -2, (SGN x2)*(-3)
6200 IF x1 > 49 THEN GO SUB 8000
6998 PAUSE 0
6999 GO TO 5100
7000 REM SOFT L
7010 GO SUB 1500
7020 PRINT #0; "Dist. obiect-lentila [0 <
x1 < 120]"
PAUSE 0
INPUT #0;x1
IF x1 < 0.1 OR x1 > 120 THEN GO
TO 7020
7025 PLOT 125-x1,90
DRAW 0,15
DRAW -2,-3
DRAW 4,0
DRAW -2,3
7030 PRINT AT 21,0; " '=40 ";AT 2
1,10; " ";AT 21,7; "x1=";x1
7035 IF x1=40 THEN PRINT FLASH 1
;AT 21,35; "Imagine la infinit"
GO TO 5100
7040 LET f=40
ON ERROR 6100
LET x2=(x1*f)/(-x1+f)
IF x2 < 0 THEN PRINT AT 21,
48; "Imagine virtuala"
GO TO 7060
7050 PRINT AT 21,50; "Imagine reala"
7060 PRINT AT 21,15; "x2=";ABS x2
7070 LET y1=15
PRINT AT 21,26; "y1=";y1
LET y2=y1*x2/x1
PRINT AT 21,34; "y2=";ABS y2
PLOT 125-x2,90
IF x2 > 125 THEN GO TO 6999
7080 PLOT 125-x2,90
DRAW 0,y2
DRAW -2, (SGN x2)*(-3)
DRAW 4,0
DRAW -2, (SGN x2)*3
7100 IF x1 > 40 THEN GO SUB 8200
7998 PAUSE 0
7999 GO TO 5100
8000 REM RAZE DE LUMINA pt OGLIN
DA
8001 PRINT AT 0,0; "**** Daca doriti
reprezentarea razelor de lumina tastati R ****"
8002 LET A$=INKEY$
8003 IF A$="r" THEN GO TO 8009
8004 IF A$="o" THEN CLS
GO SUB 6000
8005 IF A$="l" THEN CLS
GO SUB 7000
8006 GO TO 8002
8009 PRINT AT 0,0; "*****"
FORMAREA IMAGINILOR IN OGLINZI C
ONCAVE *****"
8010 LET a=125-x1
LET b=90+y1
LET c=125-x2
LET d=90+y2

```

```

8020 LET a=a+1
8030 IF POINT (a,b)=1 THEN GO TO
8050
8040 GO TO 8020
8050 LET c=c+1
8060 IF POINT (c,d)=1 THEN GO TO
8080
8070 GO TO 8050
8100 PLOT 125-x1,90+y1
DRAW a-125+x1,b-90-y1
PAUSE 25
DRAW 85-a,90-b
DRAW 40-x2,y2
PAUSE 25
8110 PLOT 125-x1,90+y1
DRAW -40+x1,-y1
DRAW c-85,d-90
PAUSE 25
DRAW 125-x2-c,90+y2-d
8199 RETURN
8200 REM RAZE DE LUMINA pt LENTILA
8201 PRINT AT 0,0; "**** Daca doriti
reprezentarea razelor de lumina tastati R ****"
8202 LET A$=INKEY$
8203 IF A$="r" THEN GO TO 8209
8204 IF A$="o" THEN CLS
GO SUB 6000
8205 IF A$="l" THEN CLS
GO SUB 7000
8206 GO TO 8202
8209 PRINT AT 0,0; "*****"
FORMAREA IMAGINILOR IN LENTILE
BICONVEXE *****"
8210 LET a=125-x1
LET b=90+y1
LET c=125-x2
LET d=90+y2
8220 LET a=a+1
8230 IF POINT (a,b)=1 THEN GO TO
8250
8240 GO TO 8220
8250 LET c=c-1
8260 IF POINT (c,d)=1 THEN GO TO
8280
8270 GO TO 8250
8300 PLOT 125-x1,90+y1
DRAW a-125+x1,b-90-y1
PAUSE 25
DRAW 165-a,90-b
DRAW -40-x2,y2
PAUSE 25
8310 PLOT 125-x1,90+y1
DRAW -40+x1,-y1
DRAW c-85,d-90
PAUSE 25
DRAW 125-x2-c,90+y2-d
8399 RETURN

```

6.3. Metode numerice de calcul

Se dau spre exemplificare două programe de utilizare a calculatorului pentru calcule științifice. Programul 15, MATINV, rezolvă o pro-

blemă clasică a algebrei lineare: inversarea matricilor. Se folosește algoritmul de eliminare cu pivot al lui Gauss. Programul este scris în limbaj PASCAL deoarece inversarea matricilor necesită un volum mare de calcule. S-a urmărit, în același timp, exemplificarea tehnicilor de programare specifice limbajului.

Programul 16, PD (pentru prelucrarea datelor experimentale), realizează prelucrarea datelor experimentale în fizică, chimie etc. Se poate afișa graficul datelor introduse, graficul funcției $y=f(x)$, se poate calcula abaterea medie pătratică a funcției față de datele reale și se pot face interpolări după polinoamele lui Lagrange.

Programul are un caracter aplicativ avînd facilități de stocare a datelor măsurate sau prelucrate pe caseta magnetică, de extragere a rezultatelor (inclusiv grafice) la imprimantă, Este ușor de utilizat în acest scop fiind realizat pe baza tehnicii listelor de opțiuni.

Programele au fost realizate de elevii: Domokos Horia, Ianculescu Mihai, Babotă Radu, Brînzănescu Oana, Beju Cătălin, Bodean Andrei, Bogdan Cristian, Pană Diana, Popa Dan, Crișan Ovidiu, Miu Bogdan, sub îndrumarea ing. Stoica Andrian și a st. Constantinescu Cristian.

PROGRAMUL 15

```

10{$L- }
20 PROGRAM TEST;
30 TYPE MAT=ARRAY[1..10,1..10
   ]OF REAL;
40 VAR A,C:MAT;
50     N,I,J,K:INTEGER;
60     DET,ERR:REAL;
70
80
90
100 PROCEDURE MATINV(VAR A:MAT
   ;VAR DET:REAL;N: INTEGER);
110 VAR B:MAT;
120     I,J,K,L:INTEGER;
130     AUX:REAL;
140 BEGIN
150   FOR I:=1 TO N DO
160     FOR J:=1 TO N DO
170       B[I,J]:=0;
180   FOR I:=1 TO N DO
190     B[I,I]:=1;
200   DET:=1;
210   FOR L:=1 TO N DO
220     BEGIN
230       AUX:=A[L,L];
240       K:=L;
250       FOR I:=L TO N DO
260         IF ABS(AUX) < ABS(A[I,L])
           THEN
270           BEGIN
280             AUX:=A[I,L];
290             K:=I;
300           END;
310       IF AUX=0 THEN
320         BEGIN
330           WRITELN;
340           WRITELN('Matrice singulara');
350           HALT
360         END;
370       DET:=DET*AUX;
380       IF K < > L THEN
390         FOR J:=1 TO N DO
400           BEGIN
410             AUX:=A[K,J];
420             A[K,J]:=A[L,J];

```

```

430             A[L,J]:=AUX;
440             AUX:=B[K,J];
450             B[K,J]:=B[L,J];
460             B[L,J]:=AUX
470           END;
480       FOR I:=1 TO N DO
490         IF I < > L THEN
500           BEGIN
510             AUX:=A[I,L]/A[L,L];
520             FOR J:=1 TO N DO
530               BEGIN
540                 A[I,J]:=A[I,J]-AUX
550                 *A[L,J];
560                 B[I,J]:=B[I,J]-AUX
570                 *B[L,J];
580               END
590             END;
600           ELSE
610             BEGIN
620               AUX:=A[I,I];
630               FOR J:=1 TO N DO
640                 BEGIN
650                   A[I,J]:=A[I,J]/AUX;
660                   B[I,J]:=B[I,J]/AUX
670                 END
680               END;
690             FOR I:=1 TO N DO
700               FOR J:=1 TO N DO
710                 A[I,J]:=B[I,J];
720               END;
730             END;
740           END;
750 {     MAIN     }
760
770 BEGIN
780 PAGE;
790 WRITELN;
800 WRITE(' Dimensiunea matricii ');
810 READLN;READ(N);
820 WRITELN;
830 WRITELN;
840 WRITELN;

```



```

850 FOR I:=1 TO N DO
860 BEGIN
870 WRITELN;
880 WRITELN(' Linia ',I:2);
890 WRITELN;
900 READLN;
910 FOR J:=1 TO N DO
920 READ(A[I,J]);
930 END;
940 PAGE;
950 C:=A;
960 MATINV(A,DET,N);
970 WRITELN('Determinantul este
',DET:10);

```

```

980 WRITELN;
990 FOR I:=1 TO N DO
1000 BEGIN
1010 WRITELN;
1020 FOR J:=1 TO N DO
1030 WRITE(A[I,J]:8:4);
1040 END;
1044 MATINV(A,DET,N);
1045 ERR:=0;
1050 FOR I:=1 TO N DO
1060 FOR J:=1 TO N DO
1070 ERR:=SQR(A[I,J]-C[I,J]);
1080 WRITELN;
1090 WRITELN('EROAREA ',ERR);
1100 END.

```

PROGRAMUL 16

```

1 DEF FN x(x)=98+(x-xmin)/(xmax
-xmin)*156
2 DEF FN y(x)=10+(x-ymin)/(ymax
-ymin)*164
3 BORDER 0: PAPER 0: INK 6:
CLS
5 LET flagf=0: DIM V(233,2):
LET ni=0: LET ix=0: LET flagi=0
10 GO SUB 9000
15 IF c$(7)="1" AND c$(TO 6)=
"000000" THEN GO SUB 7000: GO TO
10
20 IF c$(1)="0" AND ni=0 AND c
$(6)="0" THEN INPUT: PRINT #1;
AT 1,3; INK 5; "NU AU FOST INTROD
USE DATE.": PAUSE 0: GO TO 10
25 IF c$(1)="1" THEN GO SUB 1000
30 IF c$(6)="0" THEN GO TO 35
31 IF c$(7)="0" THEN CLS: GO
SUB 6100: GO TO 35
32 GO SUB 6000
35 IF c$(2)="1" OR c$(3)="1" THEN
GO SUB 2000
40 IF c$(4)="1" THEN GO SUB 4000
: LET xf=LEN (STR$ abmp): PRINT
#1;AT 0,1; "ABATEREA MEDIE PATR
ATICA ESTE: " TAB 16-xf/2; abmp: P
AUSE 0
45 IF c$(5)="1" THEN GO SUB 5000
50 GO TO 10
1000 IF c$(7)="1" THEN GO SUB 1600:
GO TO 1007
1005 LET pz=1: GO SUB 9200
1007 IF ni=0 THEN GO TO 1025
1010 LET t$="Un nou set de date?(d/n).":
GO SUB 8400
1015 IF INKEY$="n" THEN GO SUB
1900: GO TO 1035
1020 IF INKEY$ < > "d" THEN GO TO
1015
1025 LET flagf=0: DIM V(233,2):
LET ni=0: LET ix=0: LET flagi=0
1030 LET flagi=1: INPUT AT 0,0;"

```

```

Dati funcția f(x): " LINE f$: IF f$
=" " OR LEN f$ > 32 THEN GO TO
1030
1035 INPUT AT 0,0; "Dati nr. de date de
introduș: " nit: IF nit=0
THEN RETURN
1037 IF nit < 1 OR INT nit < > nit
THEN GO TO 1035
1040 LET nit=ni+nit: IF nit > 127
THEN INPUT: PRINT #1;AT 0,9;
INK 2;; "PREA MULTE.": PAUSE 0: GO
TO 1035
1045 BORDER 0: PAPER 0: INK 6: CLS
1050 FOR i=ni+1 TO nit
1055 PRINT ("0" AND i < 100)+("0"
AND i < 10)×STR$ i+";
1060 INPUT "x: ";x: GO SUB 8500:
PRINT x$," ";
1065 INPUT "y: ";y: LET x1=x: LET
x=y: GO SUB 8500: LET x=x1:
PRINT x$
1070 IF i=1 THEN LET ix=1: LET V
(1,1)=x: LET V(1,2)=y: GO TO 1115
1075 FOR u=1 TO i-1
1080 IF x > V(z,1) THEN NEXT z: LET
V(i,1)=x: LET V(i,2)=y: LET ix=i:
GO TO 1115
1085 IF V(z,1)=x THEN LET V(z,2)
=y: LET i=i-1: GO SUB 1500: LET
i=i+1: GO TO 1055
1090 LET ix=z
1095 IF ix=i THEN LET V(i,1)=x:
LET V(i,2)=y: GO TO 1115
1100 FOR z=i-1 TO ix STEP -1
1105 LET V(z+1,1)=V(z,1): LET V(z+1,2)
=V(z,2)
1110 NEXT z: LET V(ix,1)=x: LET
V(ix,2)=y
1115 GO SUB 1500: NEXT i: LET ni
=nit
1120 GO SUB 8200: RETURN
1500 CLS: LET st=(1 AND i < 21)+(i
-20 AND i > 20): LET fn=(i AND i <

```



```

21)+(20 AND i>20)
1505 FOR q=st TO fn
1510 PRINT INK 6+(ix=q); ("0" AND
q<100)+( "0" AND q<10)+STR$ q+"
";V(q,1), " ";V(q,2)
1520 NEXT q: RETURN
1600 BORDER 0: PAPER 0: INK 6:
CLS: LET xaf=0: LET yaf=0
1605 LET t$=" PD INTRODUCE
RE DATE
1610 INK 7: GO SUB 8100: INK 6
1615 LET xaf=1: LET yaf=4
1620 PAUSE 10: LET t$="Introducerea date-
lor se poate "+CHR$ 13+ "face in orice ordine ".
1625 GO SUB 8100
1630 LET xaf=1: LET yaf=9
1635 LET t$="Puteti completa setul de date
"+CHR$ 13+"introdus anterior sau introduce
"+CHR$ 13+"unul nou."
1640 PAUSE 10: GO SUB 8100
1645 PAUSE 10: LET xaf=1: LET yaf=16
1650 LET t$="Ptr. introducerea datelor de
la "+CHR$ 13+"casetofon utilizati optiunea "
+CHR$ 13 + " OPERATII I/O. "
1655 GO SUB 8100: RETURN
1900 LET t$="O noua functie ?(d/ n).
": GO SUB 8400
1905 IF INKEY$="n" THEN RETURN
1910 IF INKEY$ < > "d" THEN GO TO 1
905
1915 INPUT AT 0,0; "Dati functia
f(x): " 'LINE f$: IF f$=" " OR LE
N f$>32 THEN GO TO 1915
1920 RETURN
1999 STOP
2000 IF C$(7)="1" THEN GO SUB 26
00
2002 IF c$(3)="1" THEN LET t$="
ARE LOC PRELUCRAREA DATELOR. "
: GO SUB 8400
2003 IF c$(2)="1" AND c$(7)="0"
THEN LET pz=2: GO SUB 9200
2004 IF c$(3)="1" AND c$(7)="0"
THEN LET pz=3: GO SUB 9200
2005 LET xmin=V(1,1): LET xmax=V(ni,1)
2010 LET ymin=V(1,2): LET ymax=V(ni,2)
2020 GO SUB 2500: BEEP .2,7: BORDER
0: PAPER 0: INK 6: CLS
2025 FOR i=0 TO 20: PRINT AT i,12;
INK 5;"
NEXT i
2030 OVER 1: IF xmin*xmax>0 THEN
GO TO 2040
2035 PLOT INK 5;FN x(0),8: DRAW
INK 5;0,167: DRAW INK 5;-2,-2: D
RAW INK 5;4,0: DRAW INK 5;-2,2
2040 IF ymin*ymax>0 THEN GO TO
2047
2045 PLOT INK 5;96,FN y(0): DRAW
INK 5;159,0: DRAW INK 5;-2,-2:
DRAW INK 5;0,4: DRAW INK 5;2,-2
2047 GO SUB 2075
2050 IF c$(2)="0" THEN INVERSE 0
: GO TO 2300
2055 PLOT INK 5;FN x(V(2,1)),FN
y(V(1,2))
2060 FOR i=1 TO ni
2065 DRAW INK 5;FN x(V(i,1))-
PEEK 23677,FN y(V(i,2))-PEEK 23678
2070 NEXT i: OVER 0
2072 IF c$(3)="1" THEN INVERSE 1
: GO TO 2300
2073 GO SUB 8200: RETURN
2075 FOR i=0 TO 10
2080 LET y=ymin+(ymax-ymin)/10*i
2085 LET yf=LEN (STR$ y): PRINT
AT 20-2*i,0;TAB (12-yf);y;
2090 NEXT i
2095 LET x=xmin: GO SUB 8500:
PRINT AT 21,12;x$;: LET x=xmax: GO
SUB 8500: PRINT AT 21,32-LEN x$; x$
2100 RETURN
2299 INVERSE 0
2300 PLOT INK 5;FN x(xmin),FN y(Y(1)):
FOR i=1 TO 160
2305 LET x=xmin+(xmax-xmin)/160* i
2310 DRAW INK 5;FN x(x)-PEEK 236
77,FN y(Y(i))-PEEK 23678
2315 NEXT i: INVERSE 0
2320 GO SUB 8200: RETURN
2500 IF c$(3)="0" THEN GO TO 253 0
2505 LET flagf=1: DIM y(160): FOR
i=1 TO 160
2510 LET x=xmin+(xmax-xmin)/160* i:
LET y(i)=VAL f$
2515 IF y(i)<ymin THEN LET ymin=y(i)
2520 IF y(i)>ymax THEN LET ymax=y(i)
2525 NEXT i
2530 IF c$(2)="0" THEN RETURN
2535 FOR i=1 TO ni
2540 IF V(i,2)<ymin THEN LET ymin
=V(i,2)
2545 IF V(i,2)>ymax THEN LET ymax
=V(i,2)
2550 NEXT i: RETURN
2600 BORDER 0: PAPER 0: INK 6: C LS
2605 LET xaf=0: LET yaf=0
2610 LET t$=" PD GRAFIC DATE
/FUNCTIE
2615 INK 7: GO SUB 8100: INK 6
2620 PAUSE 20: LET xaf=1: LET ya f=3
2625 LET t$="Graficele se fac pe un spatiu
"+CHR$ 13+"de 160/160 pixeli. "

```



```

2630 GO SUB 8100
2635 PAUSE 10: LET xaf=1: LET yaf=8
2640 LET t$="Daca este necesar se traseaza
"+CHR$ 13+"si axele de coordonate."
2645 GO SUB 8100
2650 PAUSE 10: LET xaf=1: LET yaf=13
2655 LET t$="Pe ordonata se afiseaza valori
-"+CHR$ 13+"le extreme ale x-ului, iar
absci-"+CHR$ 13+"sa se gradeaza, afis-
indu-se zece"+CHR$ 13+"valori ale y-ului
intre maxime."
2660 GO SUB 8100
2665 GO SUB 8200: RETURN
2999 STOP
4000 LET abmp=0
4005 FOR i=1 TO ni
4010 LET x=V(i,1): LET y=VAL f$
4015 LET abmp=abmp+(V(i,v)-y)*(V
(i,2)-y)
4020 NEXT i
4025 LET abmp=abmp/(ni+1)
4030 RETURN
5000 IF ni<2 THEN INPUT ;: PRINT
#1;AT 1,3; INK 5; "NU AU FOST IN
TRODUSE DATE.": PAUSE 0: RETURN
5005 LET pz=5: GO SUB 9200
5015 LET xf=LEN (STR$ V(1,1))+LE
N (STR$ V(ni,1): PRINT #1;AT 0,3;
"X-ul pt. interpolare intre:" TAB (32-xf)/
2-4;V(1,1);" si ";V(ni,1)
5020 PAUSE 0
5025 INPUT AT 0,3; ("Dati x-ul: " " ");vex
5030 IF NOT (vex>V(1,1) AND vex<V(ni,1))
THEN GO TO 5025
5035 LET rez=0
5040 FOR i=1 TO ni: LET fra=1
5045 FOR j=1 TO ni
5050 IF j=i THEN GO TO 5060
5055 LET fra=fra*(vex-V(j,1))/(V(i,1)-
V(j,1))
5060 NEXT j
5065 LET rez=rez+V(i,2)*fra
5070 NEXT i
5075 LET x=vex: GO SUB 8500: PRINT
#1;AR 0,3;" ";x$
5080 LET x=rez: GO SUB 8500: PRINT
#1;AT 0,12;" ";x$,
5085 GO SUB 8200
5090 RETURN
6000 BORDER 0: PAPER 0: INK 6: CLS
6003 IF c$(7)="0" THEN GO TO 6100
6005 LET xaf=0: LET yaf=0
6010 LET t$=" PD OPERATII
I/O.
6015 INK 7: GO SUB 8100: INK 6
6020 PAUSE 20: LET xaf=1: LET yaf=3

```

```

6025 LET t$="Datele se pot salva, incarca,
ve-"+CHR$ 13+"rifica sau lista."
6030 GO SUB 8100
6035 PAUSE 10: LET xaf=1: LET ya f=8
6040 LET t$="Listarea se poate face atit la
"+CHR$ 13+"imprimanta cit si pe ecran."
6045 GO SUB 8100
6050 PAUSE 10: LET xaf=1: LET yaf=13
6055 LET t$="Datele se vor salv a conden-
sat"+CHR$ 13+"sub un nume dat de
utilizator."
6060 GO SUB 8100
6065 PAUSE 10: LET xaf=1: LET ya f=18
6070 LET t$="Se pot incarca date sub un
anu-"+CHR$ 13+"mit nume sau primele
gasite."
6075 GO SUB 8100: GO SUB 8200
6090 GO SUB 8200: PAPER 0: INK 6
: CLS
6100 PRINT "TAB 9;"PD OPERATII I/O."
6105 PRINT "TAB 14;"MENU"
6110 PLOT 25,131: DRAW 86,0: PLOT
143,131: DRAW 86,0: DRAW 0,-12 2:
DRAW -204,0: DRAW 0,122
6115 PRINT AT 8,7;"SALVARE D
ATE.":AT 10,7;"INCARCARE DATE.
":AT 12,7;"VERIFICARE DATE.":AT
14,7;"LISTARE DATE/IMP.":AT 16,
7;"LISTARE DATE/ECR."
6120 FOR z=113 TO 38 STEP -16
6125 PLOT 53,z: DRAW 147,0: DRAW
0,-10: DRAW -148,0: DRAW 0,10
6130 NEXT z
6135 LET pz=1: LET dpz=0: LET d$
="00000"
6140 PRINT AT 6+2*pz,5; INK 2;"> "
6145 IF INKEY$="a" AND pz<5 THEN
LET dpz=1
6150 IF INKEY$="q" AND pz>1 THEN
LET dpz=-1
6155 IF INKEY$=" " AND d$<>"0000
0" THEN GO TO 6200
6160 IF INKEY$=CHR$ 13 THEN GO
SUB 6700
6165 IF dpz=0 THEN GO TO 6140
6170 PRINT AT 6+2*pz,5;" ": LET
pz=pz+dpz: BEEP .02,0
6175 LET dpz=0: GO TO 6140
6200 IF d$(4)="1" AND d$(5)="1"
THEN PRINT #1;AT 0,13;"EROARE":
PAUSE 0: INPUT ;: GO TO 6140
6205 IF (d$(4)="1" OR d$(5)="1")
AND (d$(1)="1" OR d$(2)="1" OR
d$(3)="1") THEN PRINT #1;AT 0,13
;"EROARE": PAUSE 0: INPUT ;: GO
TO 6140

```



```

6210 IF d$(1)="1" AND d$(2)="1"
THEN PRINT #1;AT 0,13;"EROARE":
PAUSE 0: INPUT ;; GO TO 6140
6215 IF d$(3)="1" AND d$(v)="1"
THEN PRINT #1;AT 0,13;"EROARE"
PAUSE 0: INPUT ;; GO TO 6140
6220 IF d$(1)="1" AND ni=0 THEN
INPUT s: PRINT #1;AT 1,3; INK 5;
"NU AU FOST INTRODUSE DATE.": PA
USE 0: INPUT ;; GO TO 6140
6225 IF d$(1)="1" OR d$(2)="1" O
R d$(3)="1" THEN INPUT AT 0,0;"N
umele fisierului: " LINE n$: GO
TO 6230
6220 GO TO 6395
6230 IF LEN n$ > 5 THEN GO TO 6220
6235 IF d$(1)="0" THEN GO TO 626
6280 IF d$(2)="0" THEN GO TO 631 0
6240 GO SUB 6900
6245 IF n$=" " THEN PRINT #1;AT 0
,10;"NUME INVALID": PAUSE 0: INP
UT ;; GO TO 6140
6250 LET pz=1: GO SUB 9200: BEEP
.1,7: SAVE "PDATA"+n$ DATA V():
GO TO 6260
6260 IF d$(3)="0" THEN GO TO 628 0
6265 LET pz=3: GO SUB 9200: INK
0: PRINT AT 20,0;; BEEP .1,7
6270 IF n$=" " THEN VERIFY " " DAT
A V(): INK 6: GO TO 6280
6275 VERIFY "PDATA"+n$ DATA V():
INK 6
6280 IF d$(2)="0" THEN 60 TO 6310
2685 LET pz=2: GO SUB 9200: INK
0: PRINT AT 20,0: BEEP .1,7
6290 IF n$=" " THEN LOAD " " DATA
V(): INK 6: GO TO 6300
6295 LOAD "PDATA"+n$ DATA V(): I
NK 6
6300 GO SUB 6800: REM -decomp.-
6310 RETURN
6397 IF ni=0 THEN PRINT #1;AT 0,
3;"NU AU FOST INTRODUSE DATE.":
PAUSE 0: INPUT ;; RETURN
6417 IF flagf=0 THEN GO TO 6440
6420 INPUT ;; LET t$="Listarea val.
functiei ?(d/n). ": GO SUB 8400
6430 IF INKEY$="d" THEN GO TO
6500
6435 IF INKEY$="n" THEN GO TO
6430
6440 BORDER 0: PAPER 0: INK 6: CLS
6445 FOR q=1 TO ni
6450 IF d$(5)="1" THEN PRINT INK
6+(ix=q);("0" AND q<100)+("0" AND
q<10)+STR$ q+": ";V(q,1)": "
;V(q,2)
6452 IF d$(4)="1" THEN LPRINT INK
6+(ix=q);("0" AND q<100)+("0"

```

```

AND q<10)+STR$ q+": ";V(q,1),": ";V2,2)
6455 IF INT i/20=i/20 THEN GO
SUB 8300: PAUSE 0
6460 NEXT q: GO SUB 8200: RETURN
6700 FOR z=104-2*(pz-1)*8 TO 112
-2*(pz-1)*8
6705 PLOT 53,z: DRAW OVER 1;146, 0
6710 NEXT z
6715 LET d$(pz)=("0" AND d$(pz)= "1")
+("1" AND d$(pz)="0")
6720 BEEP .1,7: RETURN
6800 LET ni=V(201,1): LET flagi=
V(201,2): LET f$=" "
6805 FOR i=1 TO 32
6810 IF V(i+200,1) < > 0 THEN LET f
$=+$+CHR$ V(i+200,1): NEXT i
6815 RETURN
6900 LET V(201,1)=ni: LET V(201, 2)
=+flagi
6905 FOR i=1 TO LEN f$
6910 LET V(201+i,1)=CODE f$(i)
6915 NEXT i: RETURN
6999 RETURN
7000 BORDER 0: PAPER 0: INK 6: CLS
7005 LET xaf=0: LET yaf=0
7010 LET t$="PD V 0.1 ? 1988 IAN
CULESCU MIHAI"
7020 INK 7: GO SUB 8100: INK 6
7025 PAUSE 20: LET xaf=1: LET yaf
=3
7030 LET t$="Programul realizeaz a prelu-
cra-" +CHR$ 13+"rea datelor experimen-
tale in" +CHR$ 13+"fizica, chimie s.a."
7033 GO SUB 8100
7035 LET xaf=1: LET yaf=11
7040 LET t$="Cu ajutorul lui se pot rea-
liza:" +CHR$ 13+CHR$ 13+"-graficul da-
telor introduse +CHR$ 13+"-graficul func-
tiei: y=f(x) " +CHR$ 13+"-abaterea me-
die patratice " +CHR$ 13+"-interpolari."
7045 GO SUB 8100
7050 GO SUB 8200
7055 BORDER 0: PAPER 0: INK 6: C
LS: LET xaf=0: LET yaf=0: LET t
$="PD V 0.1 ? 1988 IAN CULESCU MI
HAI"
7060 INK 7: GO SUB 8100: INK 6
7065 PAUSE 20: LET xaf=1: LET yaf
=3
7070 LET t$="Introducerea datelor se va
face" +CHR$ 13+"oricum, programul or-
donindu-le in" +CHR$ 13+"ordinea cres-
catoare a x-ului."
7075 GO SUB 8100
7080 PAUSE 20: LET xaf=1: LET yaf
=9
7085 LET t$="Graficele se pot desena se-
parat" +CHR$ 13+"sau suprapus, infunctie
de op-" +CHR$ 13

```



```

+"tiunile din meniul de comenzi. "
7090 GO SUB 8100
7095 PAUSE 20
7100 LET xaf=1: LET yaf=15
7105 LET t$="Interpolarea se va face dupa
"+CHR$ 13+"polinoamele lui LAGRANGE
ptr. x"+CHR$ 13+"intre minimul si maxi-
mul dintre"+CHR$ 13+"x-urile introduse."
7110 GO SUB 8100
7115 GO SUB 8200: RETURN
8100 PRINT AT yaf,xaf:
8105 FOR i=1 TO LEN t$
8110 IF t$(i)=CHR$ 13 THEN PRINT
CHR$ 8": GO TO 8120
8115 PRINT INK 2;" ";: BEEP .001 ,20:
PRINT CHR$ 8;t$(i);
8120 NEXT i: RETURN
8200 LET t$="APASATI O TASTA PT.
A CONTINUA."
8205 GO SUB 8400
8210 PAUSE 0: RETURN
8245 IF n$=" " THEN PRINT #1;AT 0
,12;"NUME INVALID": PAUSE 0: GO
TO 6140
8250 LET pz=1: GO SUB 9100: BEEP
.1,7: SAVE "PDATA"+n$
8300 FOR z=0 TO 10
8305 BORDER 7*RND: BEEP .005,32*
RND
8310 NEXT z: BORDER 0: RETURN
8400 FOR i=1 TO LEN t$
8405 IF t$(i)>CHR$ 31 THEN PRINT
#1;AT 1,0; INK 5;t$(TO i): BEE
P .001,20: GO TO 8415
8410>PRINT #1;AT 1,0; INK 5;t$(
TO i+1): BEEP .001,20: LET i=i+1
8415 NEXT i: RETURN
8460 NEXT i: RETURN
8500 LET x$=STR$ x
8505 IF LEN x$ < 10 THEN RETURN
8410 LET pp=0: LET pe=0: FOR z=1
TO LEN x$
8515 IF x$(z)="E" THEN LET pe=z
8520 IF x$(z)="." THEN LET PP=z
8525 NEXT z
8530 IF pe=0 AND pp=0 THEN RETURN
8535 IF pe=0 THEN LET x$=x$(TO
pp)+x$(pp+1 TO 8): RETURN
8540 IF pp=0 THEN RETURN
8545 LET x$=x$(TO pp)+x$(pp+1 TO
(pe-1 AND pe-pp < 4)+(5 AND pe-
pp > 3))+x$(pe TO ): RETURN
9000 OVER 0: INVERSE 0: GO SUB 8
300: BORDER 0: PAPER 0: INK 6: CLS
9005 PRINT TAB 8;"PD V 0.1 ? 19
88" '' TAB 7; INK 2;"IANCULESCU M
IHAI."
9010 PRINT '' TAB 14;"MENU"
9015 PLOT 25,131: DRAW 86,0: PLO
T 143,131: DRAW 86,0: DRAW 0,-13 1:

```

```

DRAW -202,0: DRAW 0,131
9020 PRINT AT 8,7;"INTRODUCERE
DATE.";AT 10,7;"GRAFICUL DATELO
R.";AT 12,7;"GRAFICUL FUNCTIEI."
;AT 14,7;"ABATERE PARTRATICA.";AT
16,7;"INTERPOLARE.";AT 18,7;"OPE
RATII I/O.";AT 20,7;"EXPLICATII."
9025 FOR z=113 TO 16 STEP -16
9030 PLOT 53,z: DRAW 147,0: DRAW
0,-10: DRAW -147,0: DRAW 0,10
9035 NEXT z
9040 LET pz=1: LET dpz=0: LET c$
="00000000"
9045 PRINT AT 6+2*pz,5; INK 2;">"
9050 IF INKEY$="a" AND pz < 7 THEN
LET dpz=1
9055 IF INKEY$="q" AND pz > 1 THEN
LET dpz=-1
9060 IF INKEY$=" " AND c$ < "0000
000" THEN RETURN
9065 IF INKEY$=CHR$ 13 THEN GO S
UB 9100: GO TO 9050
9070 IF dpz=0 THEN GO TO 9050
9075 PRINT AT 6+2*pz,5;" "
9080 LET pz=pz+dpz: LET dpz=0
9085 BEEP .02,0: GO TO 9045
9100 FOR z=104-2*(pz-1)*8 TO 112-2*
(pz-1)*8
9105 PLOT 53,z: DRAW OVER 1;146,0
9110 NEXT z
9115 LET c$(pz)=( "0" AND c$(pz)=
"1 ")+("1" AND c$(pz)="0")
9120 BEEP .1,7: RETURN
9200 PRINT AT 6+2*pz,7; OVER 1;
FLASH 1;"
9205>RETURN
999 SAVE "PD V 0.1"+STR$ ver: S
AVE "PD CHSET"CODE 640007,800: VE
RIFY " ": VERIFY "CODE: RETURN

```

6.4. Elemente de proiectare asistată de calculator

"Decimus" este un program cu ajutorul căruia se poate realiza proiectarea unor circuite electronice.

În cadrul programului sînt figurate 130 simboluri ale unor elemente electronice uzuale (tranzistori, diode, rezistențe etc.) care pot fi amplasate în orice configurație prestabilită. Schema obținută se poate salva pe casetă magnetică pentru o utilizare (modificare) ulterioară sau se poate reproduce cu ajutorul imprimantei.

Programele de proiectare asistată de calculator au fost realizate de elevii: Pitic Dan, Bășulescu Ilie, Petrescu Mircea, Deac Sebastian, Singer Harold, Dumitrescu Romeo, sub îndrumarea ing. Stoica Adrian și a st. Stănescu Nicolae.

PROGRAMUL 17

```

4 CLEAR 39999: LOAD " " CODE 40
000
5 GO SUB 9900: PAPER 7: BORDE
R 7: CLS: GO SUB 4500: PRINT AT
10,0;INK2;m$;INK 4;PAPER 6;FLASH 1;
BRIGHT 1;AT 10,0; OVER 1;u$: FOR n=
1 TO 4: BEEP .1,50: NEXT n: GO SUB
8000
7 POKE 23658,255
30 REM
31 REM DECIMUS
32 REM
500 LET Q=2: LET out=0: CLS: GO
SUB 4020: PRINT AT 10,0; PAPER 2;
INK 6; FLASH 1; "1"; INVERSE 1;
" "2": PRINT AT 10,2; INK 1; ".
.....ZX Spectrum,HC-85";AT
12,2;"..... TIM-S cu RCD-
9335";AT 17,2;" ALEGETI MODUL DE
IMPRIMARE" " "
510 LET B$=INKEY$: IF B$=" " THE
N GO TO 510
520 IF B$="1" THEN LET out=1: L
ET M=1: GO TO 550
530 IF B$="2" THEN LET out=2: L
ET M=2: GO TO 550
540 GO TO 500
550 GO SUB 799
700 LET in=0: CLS: LET q=2: GO
SUB 4020: PRINT AT 10,0; PAPER
2; INK 6; FLASH 1;"1"; INVERSE 1
;" "2": PRINT AT 10,2; INK 1; "...
..... tastatura spectrum";AT 12,2;
"..... kempston Joy s. tick";AT 17,2;
"PENTRU INFORMATII TASTATI
H" " "
701 LET e=CODE INKEY$: IF e=49
THEN LET m=1: GO TO 800
702 IF e=50 THEN LET m=2: GO TO
800
703 LET in=in+1: IF in=7 THEN L
ET in=0
705 PRINT AT 17,2; INK in; OVER 1;
u$;AT 18,2; INK in; OVER 1;u$;AT
19,2; INK in+1;u$: BEEP .001 ,50: GO
TO 701
799 PRINT AT 18,2;u$;AT 19,2;u$;AT
8+m*2,14; INK 1; OVER 1; FLASH 1;
BRIGHT 1;"
": FOR n=1 TO 10: BEEP .1,n: NEXT
n: CLS: RETURN
800 PRINT AT 18,2;u$;AT 19,2;u$; ;AT
8+m*2,14; INK 1; OVER 1; FLASH 1;
BRIGHT 1;"
": FOR n=1 TO 10: BEEP .1,n: NEXT
n: CLS: GO SUB 4000
1001 IF m=1 THEN LET t=CODE INKE
Y$
1002 IF m=2 THEN LET k=IN 31
1020 GO SUB 2000: POKE 23606,0:

```

```

POKE 23607,60: GO SUB 6000: GO T 0
1001
2005 IF t=0 OR k=0 THEN RETURN
2006 IF t=13 OR k=16 THEN GO SUB
2900
2010 IF t=80 OR k=8 THEN LET cr=
cr-1: IF cr<3 THEN LET cr=3
2011 IF t=76 OR k=4 THEN LET cr=
cr+1: IF cr>20 THEN LET cr=20
2012 IF t=88 OR k=1 THEN LET cc=
cc+1: IF cc>30 THEN LET cc=30
2013 IF t=90 OR k=2 THEN LET CC=
cc-1: IF cc<1 THEN LET cc=1
2099 GO SUB 2500: RETURN
2519 PRINT AT crr,ccc; OVER 1; INK 0;
PAPER 7;" "
2520 PRINT AT cr,cc; INK 7; PAPER 2;
OVER 1; FLASH 1;" "
2580 BEEP .001,55: LET crr=cr: LET
ccc=cc: RETURN
2901 DIM a$(1,3): LET p=0 :PRINT
AT 0,0; BRIGHT 1; PAPER 6; " "
;AT 1,0; "^^^ "
2950 LET z=CODE INKEY$: IF m=1 T
HEN LET w=CODE INKEY$
2951 IF m=2 THEN LET o=IN 31
3002 IF z>47 AND z<58 THEN LET p
=p+1: GO TO 3005
3003 IF (w=13 OR o=16) AND p>0 T
HEN GO TO 3009
3004 BEEP .001,68: GO TO 2950
3005 LET a$(1,p)=CHR$ z: PRINT AT 0;
p-1; BRIGHT 1; PAPER 6;a$(1, p)
3006 IF p=3 THEN GO TO 3009
3007 BEEP .1,56: GO TO 2950
3009 LET V=VAL a$(1,1 TO p)
3010 IF v>31 AND v<128 THEN POKE
23606,64: POKE 23607, 156: GO TO
3015
3011 IF v>143 AND v<163 THEN GO
TO 3015
3012 BEEP .2,-10: GO TO 2901
3025 LET v$=CHR$ v: PRINT AT cr,
cc,v$
3099 PRINT AT 0,0;" ";AT 1,0;"
": BEEP .15,36: RETURN
4010 LET cr=3: LET cc=1: LET t=1
00: LET k=100: LET w=100: LET o=
100
4011 LET crr=3: LET ccc=1
4012 LET q=4: GO SUB 4020: INK 4
4014 PLOT 2,2: DRAW 251,0: DRAW
0,155: DRAW -251,0: DRAW 0,155:
PLOT 5,5: DRAW 245,0: DRAW 0,149:
DRAW -245,0: DRAW 0,-149
4019 INK 0: GO SUB 2500: RETURN
4021 PRINT AT 0,q; INK 1;g$;AT 1
,q;h$;AT 0,q+9; INK 3;" DECIMUS
": RETURN
4510 LET m$=" "
opreste
banda

```



```

4511 LET u$="
      ": LET g$="
      ": LET h$="
      ": RETURN
6001 LET s=CODE INKEY$: IF s < >71
AND s < >72 AND s < >74 AND s < >32A
ND s < >73 THEN RETURN
6010 IF s=71 THEN PRINT AT cr,cC
; PAPER 7; BRIGHT 0; OVER 1;" ":
BEEP 1,30: GO TO 6013
6011 GO TO 6020
6013 RANDOMIZE USR 23760: INPUT
"NUME SCHEMA(MAX.10 CAR.)?" ;Y$:
IF LEN Y$>10 THEN LET Y$=Y$( TO
10)
6014 IF Y$=" " THEN LET Y$="*"
6015 RANDOMIZE USR 23772: SAVE y
$SCREEN$
6020 IF s=73 THEN PRINT AT cr,cC
; PAPER 7; OVER 1;" ": BEEP 1,30
: GO SUB 8700: COPY
6025 IF s=72 OR s=104 THEN RANDO
MIZE USR 23760: CLS: BEEP 1,50:
GO SUB 8500: RANDOMIZE USR 2377
2: RETURN
6030 IF s=74 THEN BEEP 1,30: RAN
DOMIZE USR 23760: CLS: GO TO 60
32
6031 >GO TO 6040
6032 INPUT "INCARCARE SCHEMA (Y/
N)?;Y$: IF Y$="Y" OR Y$="y" THE
N GO TO 6034
6033 CLS: GO SUB 4000: RETURN
6034 INPUT "NUME SCHEMA(MAX.10 C
AR.)?" ;Y$: IF LEN Y$>10 THEN LET
Y$=Y$( TO 10)
6035 LOAD Y$CODE 41030: RANDOMIZ
E USR 23772: RETURN
6040 IF s=32 THEN GO SUB 7500
6099 GO SUB 2500: RETURN
7048 PLOT xx+1,yy: DRAW 1,1: DRAW
0,2: DRAW -1,1: DRAW -1,-1: DRAW
0,-2: RETURN
7049 PLOT xx+1,yy: DRAW 0,4: DRAW
-1,-1: RETURN
7050 PLOT xx+2,yy: DRAW -2,0: DRAW
2,2: DRAW 0,1: DRAW -1,1: DRAW -1,
-1: RETURN
7051 PLOT xx,yy: DRAW 1,0: DRAW 1,1:
DRAW -1,1: DRAW -1,0: DRAW 1,0:
DRAW 1,1: DRAW -1,1: DRAW -1,0:
RETURN
7052 PLOT xx+2,yy: DRAW 0,4: DRAW
-2,-2: DRAW 0,-1: DRAW 1,0: RETURN
7053 PLOT xx,yy: DRAW 1,0: DRAW 1,1:
DRAW -1,1: DRAW -1,0: DRAW 0,2:
DRAW 2,0: RETURN
054 PLOT xx+1,yy+2: DRAW 1,-1:
DRAW -1,-1: DRAW -1,1: DRAW 0,2:
DRAW 1,1: DRAW 1,0: RETURN
7055 PLOT xx,yy: DRAW 0,1: DRAW 2,2:
DRAW 0,1: DRAW -2,0: RETURN

```

```

7056 PLOT xx,yy+1: DRAW 1,-1: DRAW
1,1: DRAW -2,2: DRAW 1,1: DRAW 1,
-1: RETURN
7057 PLOT xx,yy: DRAW 1,0: DRAW 1,1:
DRAW 0,2: DRAW -1,1: DRAW -1,-1:
DRAW 1, 1: RETURN
7065 PLOT xx,yy: DRAW 0,3: DRAW 1,1:
DRAW 1,-1: DRAW 0,-1: DRAW -1,0:
DRAW 1,-1: DRAW 0,-1: RETURN
7067 PLOT xx+2,yy: DRAW -1,0: DRAW
-1,1: DRAW 0 2: DRAW 1,1: DRAW 1,0:
RETURN
7068 PLOT xx,yy: DRAW 0 4: DRAW 1,0:
DRAW 1,-1: DRAW 0,-2: DRAW -1,-1:
RETURN
7069 PLOT xx+2,yy: DRAW -2 0: DRAW
0,2: DRAW 2,0: DRAW -2,0: DRAW 0,2:
DRAW 2,0: RETURN
7070 PLOT xx,yy: DRAW 0,2: DRAW 2,0:
DRAW -2,0: DRAW 0 2: DRAW 2,0:
RETURN
7073 PLOT xx,yy: DRAW 2,0: DRAW -1,0:
DRAW 0,4: DRAW 1,0: DRAW - 2 0:
RETURN
7074 PLOT xx,yy+1: DRAW 1,-1: DRAW
1,1: DRAW 0,3: RETURN
7076 PLOT xx+2,yy: DRAW -2,0: DRAW
0 4: RETURN
7077 PLOT xx,yy: DRAW 0,4: DRAW 1 -1:
DRAW 1 1: DRAW 0 -4: RETURN
7080 PLOT xx,yy: DRAW 0 4: DRAW 1,0:
DRAW 1,-1: DRAW -1,-1: RETURN
7082 PLOT xx,yy: DRAW 0,4: DRAW 1,0:
DRAW 1,-1: DRAW -1,-1: DRAW 1,-1:
DRAW 0,-1: RETURN
7083 PLOT xx,yy: DRAW 1,0: DRAW 1 1:
DRAW -2 2: DRAW 1 1: DRAW 1 0:
RETURN
7084 PLOT xx+1,yy: DRAW 0,4: DRAW
1,0: DRAW -2,0: RETURN
7086 PLOT xx yy+4: DRAW 0,-3: DRAW
1,-1: DRAW 1,1: DRAW 0,3: DRAW 0 -1:
RETURN
7501 DIM c$(1,3): PRINT AT 0,0: PAPER
3; INK 7; BRIGHT 1;" " ;A T 1 0;
"^^^"; LET xx=cc*8+1: LET yy=169
-(cr*8): LET p=0
7519 LET z=CODE INKEY$: IF (z>47
AND z<58) <OR (z>66 AND z<71) OR
z=65 OR z=74 OR z=77 OR z=86 OR
z=73 OR z=76 OR z=80 OR (z>81 AND
z<85) THEN GO TO 7560
7550 IF p=3 OR (z=13 AND p>0) THEN
GO TO 7580
7551 BEEP .001 35: GO TO 7510
7560 IF p<3 THEN LET p=p+1
7561 LET c$(1 p)=CHR$ z: PRINT AT 0,p
-1; INK 7; PAPER 3; BRIGHT 1;c$
(1 p): BEEP .1,56: GO TO 7550

```



```

7581 FOR n=1 TO p: LET v=CODE c$
(1,n): GO SUB 7000 +v: LET xx=xx+4:
NEXT n
7599 PRINT AT 0 0;" "; AT 1 0;"
": BEEP .15,38: RETURN
8011 DATA 39,35,33,33,33,66,76,240,147,141,
145,161,193,128,14,0,21,26,0,127,0,26,21,16,80,
176,0,252,0,176,80,16,0,0,16,146,84,84,56,16
8012 DATA 0,66,90,90,219,90,66,0,0,49,74,49
2,9,0,0,0,0,140,82,140,123,0,0,0,0,49,74,49,239,
0,63,0,0,140,82,140,123,0,252,0,16,2,55,112,32,
200,28,255,16
8013 DATA 255,255,192,192,192,192,192,192,
255,255,3,3,3,3,3,3,3,3,3,3,255,255,192,192,
192,192,192,192,255,255,255,255,0,0,0,0,0,3,3,
3,3,3,3,3,0,0,0,0,0,255,255,192,192,192,192,
192,192,192,192
8020 RESTORE 8011: FOR n=USR "a"
TO USR "s"+7: READ d: POKE n,d:
NEXT n: RETURN
8499 STOP
8500 PRINT : LET SCRS=1: LET J=0
: FOR n=32 TO 162 STEP 5: LET J=
J+1: FOR o=0 TO 4: IF n+o<=162 T
HEN PRINT TAB 5*o;n+o;: POKE 236
07,156: POKE 23606,64: PRINT CHR
$(o+n);CHR$ 32;: OPKE 23607,60:
POKE 23606,00: NEXT o: PRINT "
"
8502 IF J=10 OR (SCRS=3 AND J=7)
THEN PRINT #1;AT 1,0;"I pt. impr
imare <<orice>pt.cont.": LET B$=I
NKEY$: IF B$=" " THEN GO TO 8502
8510 IF B$="I" OR B$="i" THEN PO
KE 57989,2: POKE 57990,2: POKE 57991,
0: PRINT #1;AT 1 0;"
": LET B$
=" "": COPY
8552 IF J=10 THEN CLS: LET J=0:
LET SCRS=SCRS+1: PRINT
8553 NEXT n
8556 GO SUB 8600
8558 RETURN
8600 CLS
8605 >PRINT "Cursor sus=P " " "
Jos=L " " " stinga=X " " "
dreapta=Z " " "Salvare ecran=G
" " "Imprimare ecran=I " " "Incarcar
e/Stergere ecran=J " " "Informatii
=H " " "Coduri componente=ENTER " " "
"Text (1 la 3 simboluri)=SPACE "
8609 PRINT #1;AT 1,0;"I pt. imprimare
<orice>pt.cont."

```

```

8610 LET b$=INKEY$: IF b$=" " THEN
GO 8610
8611 IF b$="I" OR b$="i" THEN LET
b$=" ": PRINT #1;AT 1,0;"
": COPY
8620 RETURN
8700 IF OUT=1 THEN RETURN
8710 IF OUT=2 THEN PRINT #1;AT 0
,0;"11×1 21×2 31×3 42×1 52×2";AT
1,0;"62×3 73×1 83×2 93×3 "
8720 LET B$=INKEY$: IF B$=" " OR
B$<"1" OR "9"<B$ THEN GO TO 8720
8730 IF "1"<=B=B$ AND B$<"3" THEN
POKE 57989,1
8731 IF "4"<=B$ AND B$<="6" THEN
POKE 57989,2
8732 IF "7"<=B$ AND B$<="9" THEN
POKE 57989,3: LET outx=1
8733 IF B$="1" OR B$="4" OR B$="
7" THEN POKE 57990,1
8734 IF B$="2" OR B$="5" OR B$="
8" THEN POKE 57990,2
8735 IF B$="3" OR B$="6" OR B$="
9" THEN POKE 57990,3
8736 POKE 57991,0
8737 PRINT #1;AT 0,0;"
"
8739 RETURN
9000 SAVE "DECIMUS" LINE 4
9001 POKE 23736,181
9010 SAVE "SIMBOL"CODE 40000,1030
9900 LET PK=PEEK 23658
9910 IF INT (PK/16)*2=INT (PK/8) THEN
POKE 23658,(PK+8)
9920 RETURN

```

6.5. Elemente de gestiune

În continuare se prezintă un program în dBASE pentru listarea sub formă de raport a participanților la tabăra de calculatoare și a temelor abordate de ei. Este necesară formarea unei liste separate de teme pentru a nu lista de două ori o temă abordată de mai mulți participanți în colectiv. Programul este memorat în fișierul LISTEM.CMD și se apelează de sub dBASE cu "do listem".

Aplicația prezentată a fost realizată de elevii: Ostace Ana Maria, Bărbulescu Bogdan, Breckner Hannelore, Colosi Dan, sub îndrumarea st. Lutic Mircea.

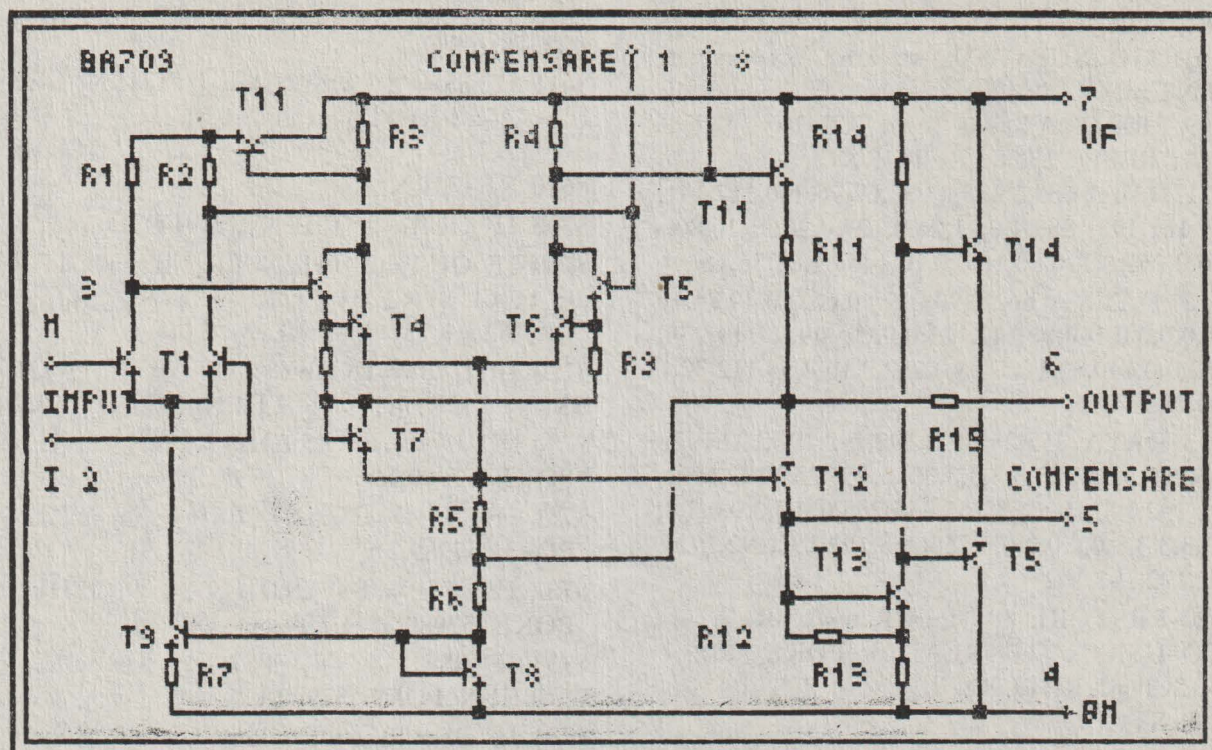


Fig. 18 Imagine aferentă programului 17

PROGRAMUL 18

STRUCTURE FOR FILE: B:TABERE.DBF

NUMBER OF RECORDS: 00130

DATE OF LAST UPDATE: 00/00/00

PRIMARY USE DATABASE

FLD	NAME	TYPE	WIDTH	DEC
001	NUME	C	020	
002	PREN	C	012	
003	LIC	C	020	
004	JUD	C	010	
005	CLS	N	002	
006	TEMA	C	002	
007	NOTAT	N	002	
008	NOTACS	N	002	
009	NOTAJ	N	002	
010	NOTAP	N	002	
011	SERIA	N	001	
** TOTAL **			00114	

DISPLAY STRUCTURE

REPORT FROM F1

COPY FIELDS TEMA TO TEME

USE TEMA

SORT ON TEMA TO TEME1

USE TEME1

STORE TEMA TO TEM

DO WHILE .NOT. EOF

SKIP

IF TEMA=TEM

DELETE

ELSE

STORE TEMA TO TEM

ENDIF

ENDDO

DELETE RECORD 1

PACK

REPORT FORM F2

USE

RETURN

PAGE NO. 00001

20/80/87

TABEL CU PARTICIPANȚII LA TABĂRA DE CALCULATOARE Tg. MUREȘ 1987

Nume	Prenume	Liceul	Județul	Cl
ABRAN	IOZSEF	L. BOLYAI FARKAS	MUREȘ	10
ABRAN	GYORGY	L. BOLYAI FARKAS	MUREȘ	11
ALIONTE	CRISTIANA	L. M. F. I. INFORMATICA	BUCUREȘTI	11
ANDRAȘ	PETER	L. D. P. GROZA	HARGHITA	10
ANDREI	ADI	NAȚIONAL	IAȘI	10
APRODU	MARIAN	CHIMIE	OLT	12

BABOTA	RADU	L M F 2 INFORMATICĂ	CLUJ	11
BĂDULESCU	CĂTĂLIN	DIMITRIE CANTEMIR	BUCUREȘTI	10
BĂRBULESCU	BOGDAN	SPIRU HARET	BUCUREȘTI	10
BĂSULESCU	ILIE	TRAIAN	MEHEDINȚI	12
BAZAC	TITUS	L M F 1 INFORMATICA	BUCUREȘTI	11
BEJU	CĂTĂLIN	MATE FIZICĂ NR 4	BUCUREȘTI	10
BELI	DAN	MATE FIZICĂ NR 1	BUCUREȘTI	12
BERGEL	CRISTIAN	L. PAPIU ILARIAN	MUREȘ	11
BERWANGER	DIETMAR	L M F 1 INFORMATICĂ	TIMIȘ	9
BODEAN	ADRIAN	L M F 1 INFORMATICĂ	BRĂSOV	11
BOGDAN	CRISTIAN	I SLAVICI	ARAD	11
BOICU	MIHAI	MAT FIZ 1 INFO	BUCUREȘTI	11
BORDI	LEVENTE	BOLYAI FARKAȘ	MUREȘ	10
BOSNEAG	ARINA	L M F 1 INFORMATICĂ	BUCUREȘTI	10
BRATU	DANIEL	L. IND. 7	VÎLCEA	10
BRECKNER	HANNELORE	L M F 2 INFORMATICĂ	CLUJ	9
BRÎNZĂNESCU	OANA	MIHAI VITEAZUL	BUCUREȘTI	12
BUDUȘAN	DAN	IND NR 1	HUNEDOARA	10
BURGHEZ	LUCIAN	INFORMATICĂ	IAȘI	10
CÎMPAN	DUCU	ȘTEFAN CEL MARE	SUCEAVA	10
CIOCAN	GHEORGHE	L. IND. 1	NEAMȚ	9
CIOCOIU	MIHAI	L M F 1 INFORMATICĂ	BUCUREȘTI	10
CODREANU	RADU	L. N. BĂLCESCU	CLUJ	10
COLOȘI	DAN	L M F 2 INFORMATICĂ	CLUJ	11
CORICI	ADRIAN	L M F 1 INFORMATICĂ	TIMIȘ	9
COSMA	LUCIAN	L. PAPIU ILARIAN	MUREȘ	11
COSMA	HORIA	L. PAPIU ILARIAN	MUREȘ	10
COTOI	IULIAN	L. N. BĂLCESCU	DOLJ	11
CRIȘAN	OVIDIU	ION NECULCE	BUCUREȘTI	12
CUCUTEANU	CLAUDIU	INFORMATICĂ	IAȘI	10
DANILA	DANIEL	DIMITRIE CANTEMIR	BUCUREȘTI	11
DEAC	SEBASTIAN	L M F 1 INFORMATICĂ	TIMIȘ	11
DINCĂ	SORIN	L. N. BĂLCESCU	DOLJ	11
DINU	ALIN	L. N. BĂLCESCU	DOLJ	11
DOBOȘAN	COSTIN	L M F 1 INFORMATICĂ	TIMIȘ	10
DOMNARIU	CORINA	L M F 2 INFORMATICĂ	CLUJ	11
DONEANU	RODICA	GH. ȘINCAI	BAIA MARE	12
DRĂCULEȚ	CORNELIA	L. N. BĂLCESCU	DOLJ	10
DRĂGUȚ	AURELIAN	MATE FIZICĂ NR 1	ARGEȘ	10
DUMITRESCU	ROMEO	TRAIAN	MEHEDINȚI	12
ERSZENYI	ZOLTAN	ELECTRO MUREȘ	MUREȘ	12
FĂTULESCU	RADU	L M F GH. LAZĂR	BUCUREȘTI	9
FEODORU	OVIDIU	M. EMINESCU	BOTOȘANI	10
FILEZ	ZOLTAN	BOLYAI FARKAȘ	MUREȘ	10

PAGE NO. 00002
20/80/87

TABEL CU PARTICIPANȚII LA TABĂRA DE CALCULATOARE Tg. MUREȘ 1987

Nume	Prenume	Liceul	Județul	Cl
FLOAREA	ADRIAN	LIC. INDS. NR 11	GALAȚI	12
FLOREA	RADU	INFORMATICĂ	IAȘI	11
GANEA	CĂLIN	EMANUIL GOJDU	BIHOR	10
GASPAR	ANDREI	L M F 1 INFORMATICĂ	TIMIȘ	9
GHIBU	LAURA	GH. LAZĂR	SIBIU	10
GOLU	CRISTIAN	L. IND. 1	GORJ	11
GOLUBOVICI	CRISTIAN	L M F GH. LAZĂR	BUCUREȘTI	11
GONGEA	VALENTIN	MATE FIZICĂ	IALOMIȚA	10
GYORFI	VALENTIN	MAT FIZ ZALĂU	ZALĂU	12
GOYRGY	LASZLO	SALAMON ERNO	HARGHITA	11
IANCULESCU	MIHAI	L. N. BĂLCESCU	BUCUREȘTI	12
ILUȚIU	VASILE	L M F 2 INFORMATICĂ	CLUJ	9
INDREAS	IOAN	MATE FIZICĂ NR 4	BUCUREȘTI	10

IONESCU	TIBERIU	L. I. L. CARAGIALE	PRAHOVA	9
IORDĂCHESCU	SORIN	L. N. BĂLCESCU	DOLJ	10
JAKAB	TIBOR	L M F 1	COVASNA	
KIS	FERENC	EMANUIL GOJDU	BIHOR	19
KURJATKO	PETER	L. BOLYAI FARKAŞ	MUREŞ	10
LASCU	CRISTIAN	L M F 1 INFORMATICĂ	TIMIŞ	11
LORINCZ	ALEXANDRU	L M F 1 INFORMATICĂ	TIMIŞ	9
MAGDO	CSABA	L. BOLYAI FARKAŞ	MUREŞ	9
MAHLER	ANDREEA	L M F 1 INFORMATICĂ	BUCUREŞTI	9
MAIEREAN	VLAD	D. CANTEMIR	BUCUREŞTI	12
MANOLESCU	DRAGOŞ	L M F 1 INFORMATICĂ	BUCUREŞTI	10
MARCHITAN	MARIUS	ŞTEFAN CEL MARE	SUCEAVA	10
MARTA	BOGDAN	DIMITRIE CANTEMIR	BUCUREŞTI	10
MATYAS	ROBERT	SC. GEN. 13	MUREŞ	8
MOISE	ADRIAN	L. B. P. HAŞDEU	BUZĂU	11
MUREŞAN	MARIUS	L M F 2 INFORMATICĂ	CLUJ	9
MUREŞAN	HORAŢIU	GH. ŞINCAI	BAIA MARE	12
MUŞAT	CRÎŞAN	I. MINULESCU	OLT	10
NICHITA	FLORIN	L. M. VITEAZUL	PRAHOVA	11
ONAŞ	ADRIAN	ALEX. IOAN CUZA	CONSTANŢA	12
OPRIŞ	DORIN	MAT. FIZICĂ	IALOMIŢA	11
OSTACE	ANAMARIA	L M F 2 INFORMATICĂ	CLUJ	11
PANĂ	DIANA	MAT FIZ V. ALECSANDRI	GALAŢI	11
PĂTRAŞCU	ALEXANDRU	L. G. BACOVIA	BACĂU	10
PETEANU	RĂZVAN	L M F 1 INFORMATICĂ	TIMIŞ	11
PETRO	ŞTEFAN	C.D. LOGA	TIMIŞ	12
PITIC	DAN	L. N. BĂLCESCU	CLUJ	12
POPA	DAN	L. G. BACOVIA	BACĂU	11
POPA	CONSTANTIN	A. I. CUZA	CONSTANŢA	12
POPESCU	SORIN	L. IND. 1	GORJ	11
POPESCU	TIBERIU	L. I. L. CARAGIALE	PRAHOVA	9
POPESCU	BOGDAN	GH. LAZĂR	SIBIU	11
PRICOP	DORU	MATE. FIZICĂ	HUNEDOARA	10
RĂDĂULEANU	GHEORGHE	L. IND. 2	NEAMŢ	10
RITTER	CSABA	L M F 1	COVASNA	11
ROŞU	LEONARD	L. I. L. CARAGIALE	PRAHOVA	10

PAGE NO .00003
20/08/87

TABEL CU PARTICIPANŢII LA TABĂRA DE CALCULATOARE Tg. MUREŞ 1987

Nume	Prenume	Liceul	Judeţul	Cl
SĂCĂREA	CRISTIAN	INDUSTRIAL NR 2	MUREŞ	12
SĂLĂBAŞ	ELENA	L. N. BĂLCESCU	DOLJ	10
SANDOR	OVIDIU	L M F 1 INFORMATICĂ	TIMIŞ	10
SANDRU	CĂLIN	L. BOLYAI FARKAŞ	MUREŞ	8
SARION	VIRGIL	D. CANTEMIR	BUCUREŞTI	11
SINGER	HARALD	L M F 1 INFORMATICĂ	TIMIŞ	10
STAN	MUGUREL	R. GRECEANU	OLT	12
STANCA	MARIAN	MATE. FIZICĂ	IALOMIŢA	11
SUHASZ	BALINT	L. BOLYAI FARKAŞ	MUREŞ	11
SULTANA	RĂZVAN	L M F 1 INFORMATICĂ	BRAŞOV	9
TĂLPĂLARU	RĂZVAN	L M F GH. LAZĂR	BUCUREŞTI	11
TECUCEANU	BEATRICE	MAT. FIZ. NR. 1	BUCUREŞTI	0
TIBĂNESEU	DANIEL	INFORMATICĂ	IASI	12
TIMAR	CRISTIAN	L M F 1 INFORMATICĂ	BRAŞOV	9
TOADER	ŞTEFAN	N. BĂLCESCU	ARGES	12
TOCA	DOREL	EMANUIL GOJDU	BIHOR	10
TOMA	SILVIU	L. IND. 4	BRĂILA	10
TOMA	GRUIA	ION CREANGĂ	BUCUREŞTI	10
TOTH	LEVENTE	L. BOLYAI FARKAŞ	MUREŞ	9
TROCANU	LIVIU	L. B. P. HAŞDEU	BUZĂU	10
TUDOR	FLORIN	DIMITRIE CANTEMIR	BUCUREŞTI	12
TURDEAN	CĂLIN	L. PAPIU ILARIAN	MUREŞ	8

UNGUROIU	ILEANA	L. N. BĂLCESCU	VÎLCEA	10
VAGAI	MIHAI	DIMITRIE CANTEMIR	BUCUREȘTI	12
VASILESCU	ALIN	MAT. FIZ. INFORMATICA	HUNEDOARA	11
VASU	OVIDIU	L M F 1 INFORMATICA	BRAȘOV	10
VISCOTCHI	IONUȚ	L. IND. 4	BRĂILA	11
VOICU	RĂZVAN	L. I. NECULCE	BUCUREȘTI	11
VOICULESCU	RUXANDRA	L. I. NECULCE	BUCUREȘTI	10
ZAPOTINSCHI	RADU	MAT. FIZICĂ	HUNEDOARA	10

PAGE NO. 00003
20/08/87

TEMELE ABORDATE ÎN TABĂRA DE CALCULATOARE Tg. MUREȘ 1987

APROX SERIILOR DE TIP INTERPOLARE/EXTRA

BAZA DE DATE "TABĂRA" (J)

BIORITMUL

CALC. SUM. PRIM. N TERM. —PROGR ARIIM GEOM

CALC. ARIEI UNUI TRIUNGHI

CALC. VOL. PIRAMIDĂ

CALCUL INTEGRAL (J)

CALCUL MATRICIAL ȘI REZ DE ECUAȚII

CALCULUL UNEI EXPRESII

CONVERSIE DIN BAZA 10 ÎN BAZA B

DERIVAREA POLIGOAMELOR

DESCOPUNEREA ÎN FACTORI PRIMI (J)

DESCOMPUNEREA UNUI ÎNTREG ÎN FACT. PRIMI

DESENAREA UNUI CUB CU CERURILE ÎNSCRISE

DET. DRUM ÎNTR-UN LABIRINT

DET. DURITĂȚII METALELOR

DET. TRIUNGHIILOR DIN N PCT (J)

EC DE GR 2 ȘI TRANȘAREA GRAFICULUI

EDITOR 87 memorare texte

EFACTE GRAFICE ȘI DE SUNET

GĂSIREA RĂDĂCINIILOR REALE ALE EC.

GEN. NUMERELOR PRIME

GENERARE FIGURI

GENERAȚIE CELULE NERVOASE (J)

GRAFICA ABSTRACTĂ

GRAFICE DE FUNCȚII

IMAGINI ÎN OGLINZI ȘI LENTILE

ÎNFĂȘURĂTOARE CONVEXĂ A N PCT DIN PLAN

JOC "NIM"

JOC GO-BANG (5 ÎN LINIE)

LABIRINT

LECȚII ASISTATE DE CALCULATOR

LEGILE GAZULUI IDEAL

LOC. GEOMETRIC

MĂRIRE DE FRAZĂ ȘI CĂUT C. M. SCT. DRUM

MATRICE

OPERAȚII CU POLINOAME

POWER 17 LIBRĂRIE

PROBLEMA DAMELOR

PROBLEME GRAFICE

PROBLEME GRAFICE, FACTORIAL, REZ. DE ECUAT.

PROGNOSIS extrapolarea serii timp

PROGRAM DIDACTIC DE FIZ.

GEN. NR. PRIME

PAGE NO. 00002
20/08/87

TEMELE ABORDATE ÎN TABĂRA DE CALCULATOARE Tg. MUREȘ 1987

PROGRAM TEST DE VERIFICAREA CUNOȘT. LA LR

PROGRAM "SISTEM SOLAR"

REALIZAREA UNOR TRAIECTORII
REPR SPAȚIALA FCT DE 2 VAR REALE
REPREZ GRAFICĂ A UNEI FCT DE GR 2
REPREZENTAREA GR A 3 FCT SINUSOIDALE
REPREZENTAREA GRAFICĂ A FUNCȚIILOR
REZ. DE ECUAȚII ȘI PROBLEME GRAFICE
REZ. EC. DE GR. I ȘI PROBLEME GRAFICE
REZ. EC. DE GRAD N
REZ. POLIN. DE INTERPOLARE LAGRANGE (J)
REZOLV TRIUNGHI OARECARE
REZOLVAREA SISTEMELOR DE EC. ȘI CALC. MAT.
ROTIREA CORPURILOR ÎN SPAȚIU
ROTIRI, GEN. DE IMAGINI, GRAFICE
RUTINA LISTARE MEMORIE
SIMULAREA EXECUȚIEI UNUI PROGRAM BASIC
SIMULATOR DE SCHEME LOGICE
SIMULATOR LIMBAJ DE ASAMBL. ASSEX
SISTEM DE ECUAȚII
SISTEM DE N ECUAȚII CU N NECUNOSCUTE
SPAȚIU
SPEGL INTERPRETOR CU FACILITĂȚI GRAFICE
ST. ARUNCĂRII SUB UN UNGHI DAT ÎN CÎMP GR.
ST. FUNCȚIILOR DE GRADUL 2
TEST DE CONTROL LA FIZICĂ
TRATAREA EC. TRANSCENDENTE
TURNURILE DIN HANOI

