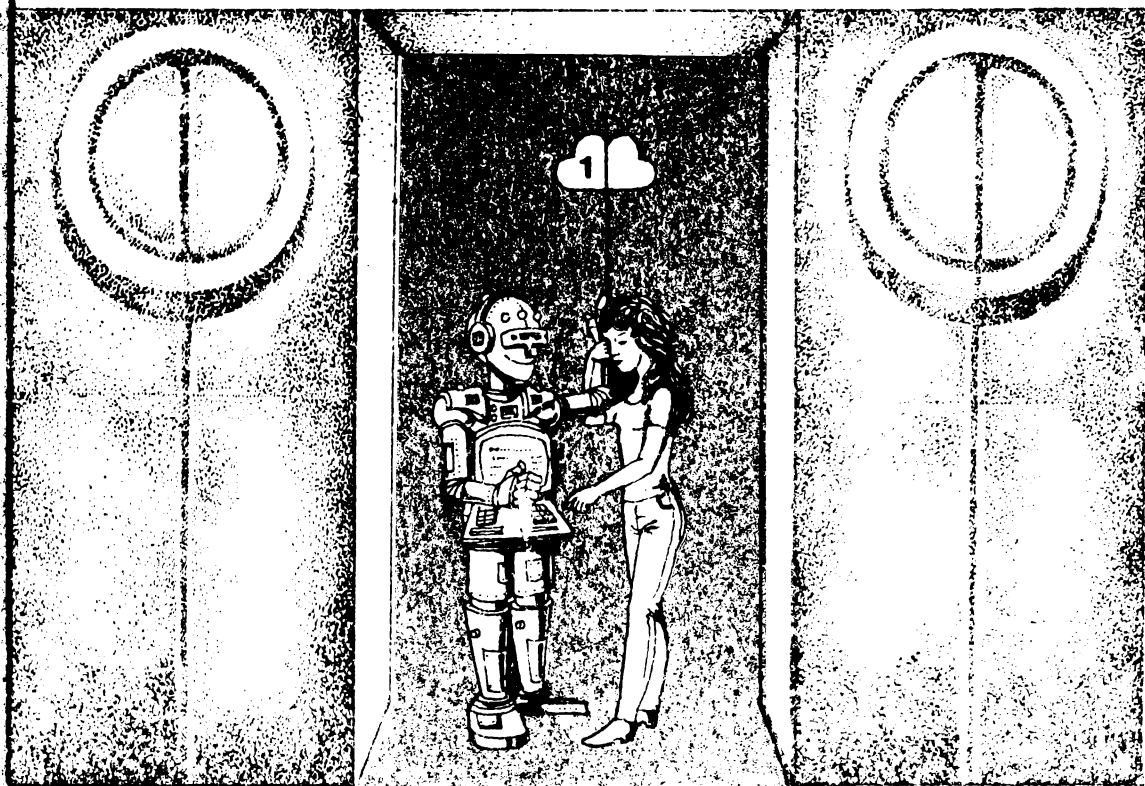


LIVIU DUMITRASCU

ÎNVĂȚĂM



MICROELECTRONICA INTERACTIVĂ

EDITURA TEHNICĂ

AUTOMATICA
INFORMATICA
ELECTRONICA
MANAGEMENT



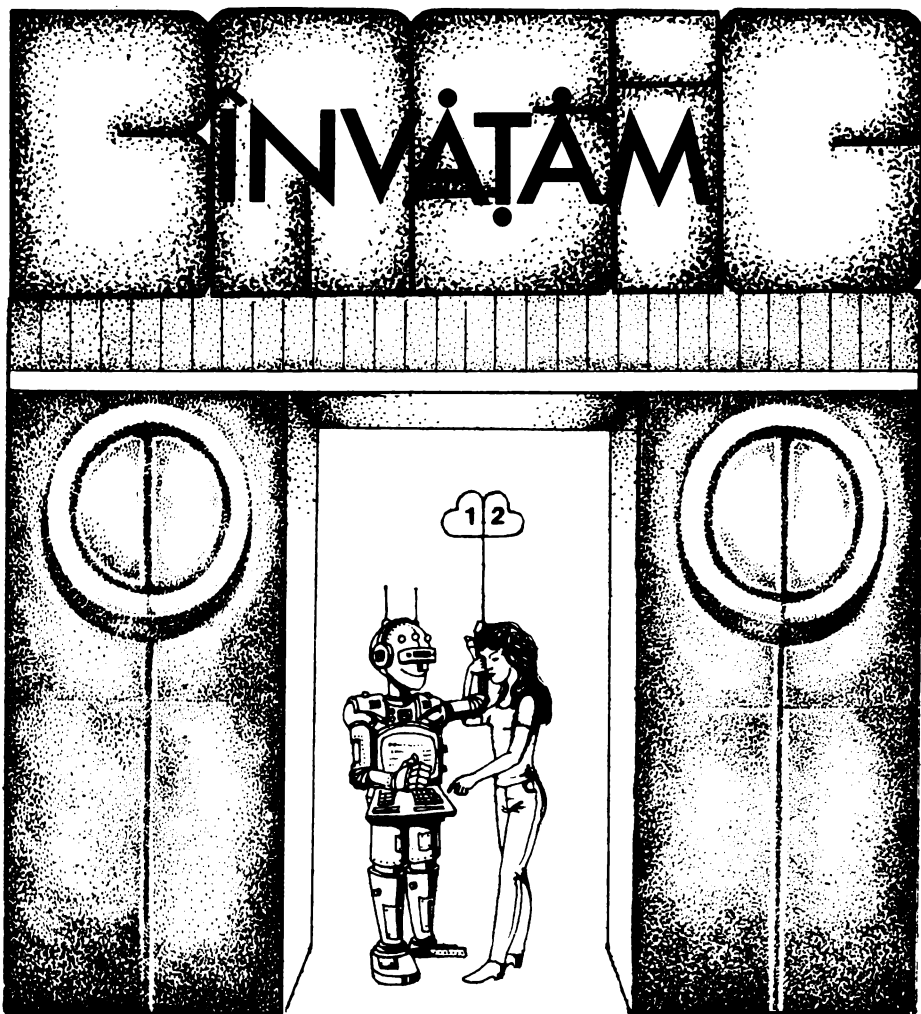
BIBLIOTECA DE

Automatică—Informatică—Electronică—Management

SERIA INIȚIERE

- E. VASILIU**
INIȚIERE ÎN DISPOZITIVELE SEMICONDUCTOARE
- D. STANOMIR**
INIȚIERE ÎN ELECTROACUSTICA
- W. TRUSZ**
ABC-UL REPARĂRII RADIORECEPTOARELOR
Traducere din lb. polonă (Ciclul ABC-uri)
- A. POPA**
ABC DE PROTECȚIA MUNCII (ciclul ABC-uri)
- MARGARETA DRĂGHICI**
INIȚIERE ÎN COBOL
- STELIAN NICULESCU**
INIȚIERE ÎN FORTRAN
- PAUL CONSTANTINESCU ȘI ZAHARIA NICOLAE**
INIȚIERE ÎN ORGANIZAREA ȘI PROIECTAREA SISTEMELOR DE CONDUCERE
- I. V. DUMITRESCU ș.a.**
INIȚIERE ÎN TELEPRELUCRAREA DATELOR
- I. CREȚU**
INIȚIERE ÎN ESTETICA PRODUSELOR (Ciclul ABC-uri)
- E. AISBERG**
ABC DE RADIO ȘI TELEVIZIUNE
Traducere din limba franceză
- J. D. WARNIER, B. MI FLANAGAN**
INSTRUIRE ÎN PROGRAMARE
Traducere din limba franceză
- I. H. BERNHARD, B. KNUPPERTZ**
INIȚIERE ÎN TIRISTOARE
Traducere din limba germană
- W. DEPPEERT, K. STOLL**
INIȚIERE ÎN PNEUMOAUTOMATICA
Traducere din limba germană
- E. VASILIU**
INIȚIERE ÎN RADIOELECTRONICĂ CUANTICĂ
- V. POPESCU**
INSTRUIRE PROGRAMATĂ ÎN CALCULATOARE NUMERICE
- ȘT. BIRLEA**
INIȚIERE ÎN CIBERNETICA SISTEMELOR INDUSTRIALE
- I. PAPADACHE**
AUTOMATIZĂRI INDUSTRIALE, INIȚIERE, APLICAȚII
- ȘT. NICULESCU**
FORTRAN, INIȚIERE ÎN PROGRAMARE STRUCTURATĂ
- J. FORRESTER**
PRINCIPIILE SISTEMELOR: TEORIE ȘI AUTOINSTRUIRE PROGRAMATĂ
Traducere din lb. engleză — S.U.A.
- P. DRANSFIELD, D. F. HABER**
INSTRUIRE PROGRAMATĂ ÎN METODA LOCULUI RĂDĂCINILOR
Traducere din lb. engleză — S.U.A.
- R. BĂRSAN**
DISPOZITIVE ȘI CIRCUITE INTEGRATE CU TRANSFER DE SARCINĂ
- D. RODDY**
INIȚIERE ÎN MICROELECTRONICĂ
Traducere din lb. engleză
- NICULESCU Cl., IOSIF M.**
INIȚIERE ÎN COMUNICAȚIILE PRIN FIBRE OPTICE
- CSABAI DĂNIEL**
TEHNICA SONORIZĂRII (traducere din lb. maghiară)
- MITROFAN GH., PFLANZER G.**
INIȚIERE ÎN TELEVIZIUNEA ÎN CULORI
- RADU NEGOESCU**
INIȚIERE ÎN ELECTRONICA BIOMEDICALĂ
- RADU NEGOESCU**
INSTRUMENTAȚIA ELECTRONICĂ BIOMEDICALĂ
- A. PETRESCU ș.a.**
TOTUL DESPRE... CALCULATORUL PERSONAL 2MIC
- L. DUMITRAȘCU ș.a.**
ÎNVĂȚĂM FORTRAN... CONVERSIND CU CALCULATORUL
- L. DUMITRAȘCU**
ÎNVĂȚĂM COBOL... CONVERSIND CU CALCULATORUL

Dr. ing. LIVIU DUMITRAȘCU



MICROELECTRONICA INTERACTIVĂ

TOTUL DESPRE... BASIC
ÎN 14 CONVERSAȚII ȘI 7 SINTEZE

1

EDITURA TEHNICĂ



BUCUREȘTI, 1989

LIVIU DUMITRAȘCU a absolvit în 1970 Institutul Politehnic din București. În anul 1986 a susținut teza de doctorat în specialitatea automatizarea instalațiilor petroliere. Este șef de lucrări la Institutul de Petrol și Gaze Ploiești. A activat timp de aproape zece ani la C.E.P.E.C.A. din cadrul Academiei „Ștefan Gheorghiu” unde a predat îndeosebi limbaje de programare în cadrul cursurilor pentru formarea specialiștilor în informatică. A absolvit în Franța cursuri de specializare în informatică și tehnică de calcul. A participat la realizarea în colectiv a unor produse-program și sisteme informatice. A publicat singur sau în colaborare la Editura Academiei R. S. România și Editura Tehnică mai multe articole și cărți, o carte editându-se și în limba engleză. Cărțile sale din același ciclu de la Editura Tehnică s-au bucurat de o mare popularitate. Domenii de interes: generarea automată a programelor prin tabele de decizie, calculatoare personale, microbaze de date, inteligență artificială.

Prefață: ec. NICOLAE BADEA DINCĂ

Recenzii: dr. ing. NICOLAE ȚĂPUȘ

prof. dr. ing. ADRIAN PETRESCU

Redactor: ing. PAUL ZAMFIRESCU

Ideea graficii de copertă și vignete:

ing. GHEORGHE LUCHIAN

**Execuția coperții: RADU GHEORGHIAN,
SIMONA DUMITRESCU, BOGDAN ILIE**

Desene: arh. CONSTANTIN MIHĂESCU

Tehnoredactor și machetare: MARIA TRĂSNEA

ISBN 973-31-0010-2

ISBN 973-31-0011-0

**Coli de tipar: 39. Bun de tipar: 12. VI. 1989.
C.Z. 621.38**

**Tiparul executat sub com. 260/1988 la
Întreprinderea Poligrafică „Crișana”
Oradea, str. L. Sălăjan nr. 105
Republica Socialistă România**

PREFAȚĂ

...Să facem totul pentru înarmarea clasei muncitoare, a intelectualității, a întregii națiuni, în primul rînd a tineretului, cu cele mai înalte cunoștințe din toate domeniile...

NICOLAE CEAUȘESCU

(din cuvîntarea la Plenara Consiliului Național al Oamenilor Muncii, 13 iulie 1989)

Am acceptat solicitarea Editurii Tehnice de a scrie cîteva cuvinte de bun sosît acestei lucrări, cel puțin din următoarele 4 motive: – sînt lucrător științific la Institutul de Tehnică de Calcul și Informatică (I.T.C.I.) care are numeroase și variate preocupări în domeniul larg pe care îl intersectează cartea; – sînt (co)autor al unor cărți de profil, apărute și în curs de apariție; – sînt consilier al redacției de specialitate și – nu sînt nici coleg, nici prieten al autorului, așa încît puteți crede cu seninătate în obiectivitatea mea de prim-cititor, care nicum – vă asigur – nu ar fi fost perturbată.

Oricum aș privi, separat din fiecare din cele patru ipostaze sau integrat, răspunsul meu la această editare este favorabil. Să mă explic:

– I.T.C.I. are ca obiective principale cercetarea științifică și ingineria tehnologică în domeniu, elaborarea și instalarea unor sisteme informatice, organizarea, dezvoltarea și desfășurarea industriei naționale de programe ș.a; organizăm cursuri teoretice și practice de formare, perfecționare și specializare a personalului ce lucrează în informatică și a utilizatorilor de tehnică de calcul; cooperăm – și prin unități distribuite teritorial – cu producătorii și beneficiarii de tehnică de calcul, desfășurînd cercetare și proiectare pentru diferite echipamente, pentru sisteme de operare, limbaje, programarea aplicațiilor etc.; asigurăm întreținerea, dezvoltarea și funcționarea Bibliotecii Naționale de Programe; motive, credem, suficiente, de a saluta apariția a cît mai multor cărți bune în domeniu, fie că ele sînt realizate de specialiștii institutului, fie, ca în acest caz, în alte nuclee de învățămînt-cercetare-producție; mai remarcăm, în contextul prefeței, preocupările speciale ale I.T.C.I. în informatica educațională; studiile de impact al calculatoarelor în educație, elaborarea și reproducerea industrială a programelor informatice pentru diverse discipline, cercetări pentru instruirea asistată pe calculatoare personale în BASIC și LOGO, realizarea de laboratoare didactice-fixe și mobile pentru predarea informaticii la diferite nivele, organizarea olimpiadelor de profil ș.a.; laboratorul mobil de instruire – INFOBUS – este apreciat de multe instituții de învățămînt din țară și va putea fi promovat în cadrul programelor UNESCO;

– de curînd a apărut o lucrare de automatică și informatică aplicată, la care am adus o oarecare contribuție privind metodologia și tehnicile de programare, iar la porțile cititorilor va bate curînd un „ABC de calcul electronic” realizat de un colectiv de la I.P.B., I.T.C.I., Fabrica de calculatoare, C.N.O.P. etc., adresat cu deosebire tineretului, care se completează de minune cu conținutul prezentei cărți de programare;

– noul ciclu generic „Învățăm microelectronica interactivă” pe care Editura Tehnică îl inițiază este într-adevăr mult mai generos decît pot cuprinde aceste prime volume, care nu se preocupă de o multitudine de aspecte constructive, funcționale ș.a. ale echipamentelor electronice interactive și nici de componente specializate ale acestora ci doar de comunicarea cu ele, prin intermediul unui limbaj consacrat conversațiilor om-mașină; apreciînd această adevărată „enciclopedie BASIC” am pus la dispoziția redacției documentații de mare actualitate;

– cititorul avizat mai remarcă atît stilul original cît și organizarea funcțională a cărții, în două volume cu planuri juxtapuse, în care se intersectează 10 direcții judicioase, ce asigură un ansamblu de manuale practice de instruire și aplicare a programării în BASIC, cu codificări pe numeroase clase de echipamente, cu exemple selecționate și cu un bogat conținut de îndreptar ce extinde limitele stricte ale limbajului (microprocesoare, sisteme de operare, aplicații tipice, alte limbaje de nivel înalt prezentate comparativ etc.).

În esență, felicitînd pe autor și redacția de informatică și tehnică de calcul a Editurii Tehnice pentru această realizare, recomand unor cercuri largi de cititori, utilizatori și creatori de tehnică de calcul din toate ramurile economiei noastre să studieze aceste prețioase volume.

ec. NICOLAE BADEA DINCĂ

Director adjunct științific I.T.C.I.

N.R. Culoarea în carte s-a dorit funcțională, nu doar decorativă: temele din conversații și soluțiile lor, modulele tip de analiză și proiectare structurată a programelor din exemple, trimiterile în vol. 1 la imaginile programelor din vol. 2, sinteza 20, comenzile și instrucțiunile explicate din sinteze, chiar și... vignetele simbolice și unele titluri evidențiate stau mărturie.



Celor mai dragi tineri,
ai mei, Daniel și Andrei

Pași printre... conversații și sinteze.

Un dialog autor—editor—cititor (I)

□ Mai întâi... cititorul, "le maître". Vă explic titlul cu pătrățel, mărturisindu-vă — discret și modest — că, pe cînd începeam să scriu cartea, am dat răspunsul corect la o întrebare eternă: "ce a fost mai întâi, oul sau găina?", e drept într-un **caz** al meu, particular, carte/cititor; mă veți întreba, poate, "în ce fel, și, chiar de; nu sînteți curioși, vă voi spune, deși dezvălui — să nu zic vorbă mare — un scump secret: în cazul dat, am procedat simplu (ca bonjour), **tăind nodul gordian** și, ce să vezi, ... a **șipat la tăiat**; imediat am dedus logic "e cititorul, nu cartea" (e drept, și cartea poate țipa, cînd e anapoda, dar... să lăsăm altora calea asta). Și, ca să nu fie o șaradă, să vă descriu **cazul**. Discutam, animat, cu redactorul, planul cărții și nimerim, ca din întîmplare, într-o frumoasă librărie, unde — fatalitate?! — ce auzim: "— N-aveți cumva cartea «Învățăm BASIC... conversînd cu...?»" "— Nu", vine răspunsul sec. "— De ce oare, că a fost promisă în altă carte a aceluiași autor, «Învățăm COBOL... etc.?»". "— Nu știm, mai încercați prin vară... peste trei ani. Cărțile informatice se scriu mai greu și apar mai rar". Mă înroșesc în mine (pentru toți) și intru în... conversație, să văd dacă cititorul știe ce caută și cum ar vrea să fie, dac-ar fi. Am **tăiat**, astfel, nodul... de care vă vorbeam, iar cititorul a **șipat la mine** (q.e.d. — sau, ceea ce era de demonstrat — cititor **da**, carte **ba**).

"— Cum să nu știu, dom'le, zice, dar nu cumva dumneavoastră sînteți niscaiva autori sau editori?". "— Nu sînt (încă autor...) zic, în gîndul meu), dar dacă am fi toți trei ce ai vrea să facem?". "— Păi, mai întâi o **copertă** frumoasă, cu un **titlu** atrăgător, vorba lui Vlahuță: "La titlu să iei seama: sonor, bombastic, vag / Și cu ceva macabru să ne lovești din prag; / Că nu-i zor nicidecum / Ca titlul și povestea să aibă același drum", apoi un **cuprins**... cuprinzător, pe înțelesul **tuturor** (tuturor e un fel de a spune că și pentru mine — care sînt muncitor specialist într-o secție în care s-au instalat terminale conversaționale — și pentru fiul meu, elev la informatică și candidat la facultate, și pentru profesorii lui... măcar). Ce mai, cînd om lua-o în mină, **cartea să vorbească singură** (că autorii sînt la treburile lor); să plecăm, deci, de la simplu la complicat și să nu batem cîmpii decît în locuri știute, pe care cine vrea să le poată sări; fără prea multă teorie, dar nici în doi perii, să fie la obiect, cu programe alese, să ne ducă de la particular la general și odată ajunși aici să știm oricînd să facem cale-n-toarsă. Ca să nu ne plictisim, să aibă culoare, să fie veselă, incitantă și, mai ales, **să ne pună la treabă**... materia cenușie, cerîndu-ne mereu să dezlegăm (sau să legăm) cite ceva. Fie vorba între noi, planul cărților anterioare din ciclul, cu conversații și sinteze, n-a fost rău, l-am putea păstra, făcîndu-l mai consistent". Așa zise cititorul și se duse, așteptînd încrezător rezultatul.

Îngîndurat ("...era oare omul meu?! Da!" mi-am zis), mi-am adus aminte că și redactorul (autor al unui BASIC și școlit informatic prin Franța,

ca și subsemnatul) mi-a vorbit de un personaj – șef și executant în noul oficiu de calcul al tipografiei orădene – pe care l-a întâlnit prima oară lucrind de zor (după o ușă cu "strict oprit") la un terminal, cu "Învățăm COBOL..." pe masă, și care, întrebat pe neașteptate, a răspuns: "... E o carte **practică, învăț și lucrez după ea**, dar (era și un dar) ... ar fi fost bine să aibă și ..., dar dumneavoastră cine sînteți, ... nu știți să citiți ... etc." De acești stăpîni (și prieteni) ai noștri – cititorii – căutăm să ascultăm și acum!

Am trecut, deci, cu redactorul, la proiectarea cărții, ghidîndu-ne nu numai după auzite ci și după altele, cum ar fi (bine-ar fi!) planul, viu în suflet, al grădinilor Palatului din Versailles (ici o arteziană, colo o Diană, ici un potir, colo un satir, ici un copac, colo un lac ... totul, însă, riguros, cu perspectivă carteziană).

Apoi, multe manuscrise, iterații și, în fine, cartea. Pe un plan unitar al fiecăreia din cele 14 conversații (cu exemple, teme, soluții), "distilat" în cursul editării, am conceput **exemple și programe de instruire**, care s-au încheat rapid într-o **concepție structurată** de analiză-proiectare-programare în BASIC, redată sistematic, ca un laitmotiv, prin **documentații tipice** (diagrame de structură, pseudocod, scheme logice). În paralel, programele conversațiilor au fost **codificate în variantele de BASIC** pe calculatoare personale, personal-profesionale, minicalculatoare, calculatoare medii; **cititorul a învățat comenzile și instrucțiunile uzuale**, salvarea, depanarea și documentarea programelor, structurile fundamentale de programare (secvența, selecția, iterația), crearea, actualizarea, consultarea și listarea fișierelor etc. S-au rezolvat **mai toate tipurile de probleme** adecvate limbajului: calcule numerice, alfanumerice, vectoriale, matriciale, lucrul cu fișiere secvențiale, cu acces direct, cu fișiere masive virtuale, grafică bi- și tridimensională, lucrul cu meniuri, ferestre, culori, muzică, simularea mișcărilor, animație, conversațiile încheindu-se cu un studiu de caz complex. Variante ale manuscriselor, programelor și ale volumului 1 au fost supuse, înainte și în timpul editării, unor **testări** îndelungi ... cu public, mai ales la I.P.G. Ploiești, I.R.E. Ploiești, I.P.R.S. Băneasa.

Sintezele, ce ocupă volumul 2, au fost elaborate ulterior (cu excepția sintezei 20, cu programele exemplurilor din conversații) ținînd seama de observațiile din testări/recenzări și după consultarea unei bogate bibliografii, în bună parte recomandată de redacție. **Ele sint construite în alt mod decît conversațiile**, constituind un mediu compact de consolidare, completare și regăsire a cunoștințelor, cît și de extindere a lor asupra componentelor, echipamentelor, software-ului de sistem, aplicațiilor, programării, în general, și programării în BASIC, în special. **Pașii (II și III) printre conversații și sinteze** (pag. 606, v.1 și pag. 8, v.2) lămuresc, pe de o parte, cele zece direcții ce se intersectează în carte, iar, pe de altă parte, tratează noutăți ale domeniului, cunoscute în ultimele faze editoriale.

Mulțumiri (măcar ale mele) transmit tuturor. Dar, mai ales, următorilor: prof. dr. ing. Ioan Dumitrescu, prof. dr. ing. Stelian Dumitrescu, conf. dr. Miron Oprea, prof. dr. Eugeniu Niculescu-Mizil, dr. ing. Adrian Davidoviciu, dr. ing. Valerie Marinescu, ing. Ion Trandafir, Constanța Figaro, Florența Costea, dr. ing. Alexandru Ioachim, mat. Cristian Marinouiu, ing. Victor Stoicea, ing. Marius Manea, ing. Dan Drăgan, prog. Geo Voican, soției mele Livia Dumitrașcu, ec. Claudia Dulgheru, Leonida Băietu, conducerii Editurii Tehnice și, de fapt, întregului său colectiv, cum și Poliigrafiei orădene „Crișana” (ing. șef Vladimir Kagan, șef producție Rozalia Farkaș și colectivul respectiv de maiștri și muncitori tipografici).

Aștept verdictul cititorilor!

SUMAR

pentru volumele 1 și 2

VOLUMUL 1

Prefață	V
* Pași printre conversații și sinteze. Un dialog autor–editor–cititor (I)	VII
Sumar pentru volumele 1 și 2	IX
Cuprins detaliat volumul 1	X

Totul despre BASIC în 14 conversații și . . .

CONVERSAȚIA 1	3
CONVERSAȚIA 2	33
CONVERSAȚIA 3	67
CONVERSAȚIA 4	155
CONVERSAȚIA 5	209
CONVERSAȚIA 6	277
CONVERSAȚIA 7	371
CONVERSAȚIA 8	393
CONVERSAȚIA 9	423
CONVERSAȚIA 10	441
CONVERSAȚIA 11	479
CONVERSAȚIA 12	515
CONVERSAȚIA 13	581
CONVERSAȚIA 14	593
* Pași printre conversații și sinteze. Un dialog autor–editor–cititor (II)	606

VOLUMUL 2

Sumar pentru volumele 1 și 2	V
Cuprins detaliat volumul 2	VI
* Pași printre conversații și sinteze. Un dialog autor–editor–cititor (III)	VIII

. . . 7 sinteze

SINTEZA 15	2
SINTEZA 16	12
SINTEZA 17	66
SINTEZA 18	141
SINTEZA 19	160
SINTEZA 20	197
SINTEZA 21	286

CUPRINS DETALIAT

VOLUMUL 1

Prefață	V
Pași printre... conversații și sinteze (I). Un dialog autor—editor—cititor	VII
Sumar	IX

Totul despre BASIC în 14 conversații și . . .

CONVERSAȚIA 1. Editarea unui text (titlul cărții) cu calculatorul. Cel mai simplu program în BASIC. Metode pentru introducerea, adăugarea și corectarea interactivă a unei linii program. Comenzi BASIC: RUN și LIST. TESTE pentru cititor	3
EXEMPLELE 1-PC, m, M, F	4
<input type="checkbox"/> Imaginea programului BASIC-aMIC	4
<input type="checkbox"/> Imaginea programului BASIC-PRAE	13
<input type="checkbox"/> Imaginea programului BASIC HC-85, TIM S, SPECTRUM	15
<input type="checkbox"/> Imaginea programului BASIC-AMSTRAD	21
<input type="checkbox"/> Imaginea programului BASIC-COMMODORE	25
<input type="checkbox"/> Imaginea programului BASIC-80, GW-BASIC pentru FELIX PC	26
<input type="checkbox"/> Imaginea programului BASIC-PLUS, pentru INDEPENDENT, CORAL	27
<input type="checkbox"/> Imaginea programului ABASIC, pentru FELIX C	29
TEMA 1	30
<input type="checkbox"/> Întrebări de control din text	30
CONVERSAȚIA 2. Calculul ariei unui rezervor sferic de rază dată. Constante și variabile numerice. Expresii numerice. Instrucțiunile LET și PRINT. Comenzi BASIC: SCR, NEW, TESTE și APLICAȚII pentru cititor	33
EXEMPLELE 2-PC, m, M, F	34
<input type="checkbox"/> Citeva operații fundamentale ale calculatoarelor și descrierea lor în BASIC-aMIC	34
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-PRAE	45
<input type="checkbox"/> Particularități ale programării în limbajul BASIC HC-85, TIM S, SPECTRUM	47
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-AMSTRAD	49
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-COMMODORE	53
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-80	55
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-PLUS	57
<input type="checkbox"/> Particularități ale programării în limbajul ABASIC	60
TEMA 2	61
<input type="checkbox"/> Răspundeți prin DA sau NU la	61
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc din	62
<input type="checkbox"/> Scrieți cite un program BASIC pentru	62
<input type="checkbox"/> Calculul cantităților totale trimestriale de jucării fabricate de întreprinderea U	63
<input type="checkbox"/> Calculul consumului mediu anual al unui articol	64
SOLUȚIA TEMEI 2	64
<input type="checkbox"/> Răspunsurile sint	64
<input type="checkbox"/> Cuvintele lipsă sint	64
<input type="checkbox"/> Programele BASIC sint	64
<input type="checkbox"/> Programul pentru calculul cantităților totale de jucării fabricate de întreprinderea U este	66
<input type="checkbox"/> Programul pentru calculul consumului mediu anual este	66
CONVERSAȚIA 3. Proiectarea și realizarea unui program pentru calculul ariei și volumului unui rezervor sferic a cărui rază poate lua diferite valori. Metodică de analiză, proiectare și realizare a programelor BASIC. Introducerea dinamică a datelor.	

Instrucțiunea INPUT. Documentarea programelor. Instrucțiunea REM. Salvarea unui program BASIC pe medii magnetice. Depanarea unui program. APLICAȚII și TESTE pentru cititor		67	
EXEMPLELE 3-PC, m, M, F		68	
<input type="checkbox"/> Metodică de analiză, proiectare și realizare a programelor	68	<input type="checkbox"/> Codificarea în limbajul BASIC-AMSTRAD	94
<input type="checkbox"/> Codificarea în limbajul BASIC-aMIC	74	<input type="checkbox"/> Codificarea în limbajul BASIC-COMMODORE	110
<input type="checkbox"/> Codificarea în limbajul BASIC-PRAE	81	<input type="checkbox"/> Codificarea în limbajul BASIC-80	116
<input type="checkbox"/> Codificarea în limbajul BASIC-HC-85, TIM S, SPECTRUM	89	<input type="checkbox"/> Codificarea în limbajul BASIC-PLUS	124
		<input type="checkbox"/> Codificarea în limbajul ABASIC...	129
TEMA 3		134	
<input type="checkbox"/> Răspundeți prin DA sau NU la...	134	toare cantităților trimestriale de jucării (pentru un articol) fabricate de întreprinderea U	143
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc din...	135	<input type="checkbox"/> Calculul procentelor corespunzătoare consumurilor trimestriale ale unui articol	146
<input type="checkbox"/> Scrieți cite un program BASIC pentru	138		
<input type="checkbox"/> Calculul procentelor corespunzătoare			
SOLUȚIA TEMEI 3		148	
<input type="checkbox"/> Răspunsurile sint	148	<input type="checkbox"/> Programul pentru calculul procentelor corespunzătoare consumurilor trimestriale ale unui articol este	153
<input type="checkbox"/> Cuvintele lipsă sint	148		
<input type="checkbox"/> Programele BASIC sint	148		
<input type="checkbox"/> Programul pentru calculul procentelor corespunzătoare cantităților trimestriale de jucării este	150		
CONVERSAȚIA 4. Proiectarea și realizarea unui program pentru calculul cantității de lichid din nouă rezervoare cilindrice echilaterale a căror rază variază de la 2 la 10 m, cu pasul de un metru. Constante și variabile alfanumerice. Structura de iterație PENTRU. Bucla FOR-NEXT. "Bunicul" lui FOR-NEXT. Editarea valorilor numerice. Instrucțiunea PRINT USING. Modificatorul FOR. APLICAȚII și TESTE pentru cititor...		155	
EXEMPLELE 4-PC, m, M, F		156	
<input type="checkbox"/> De la problemă la program	156	<input type="checkbox"/> Codificarea în limbajul BASIC-AMSTRAD	179
<input type="checkbox"/> Analiza problemei	156	<input type="checkbox"/> Codificarea în limbajul BASIC-COMMODORE	187
<input type="checkbox"/> Proiectarea programului	158	<input type="checkbox"/> Particularități ale programării în limbajul BASIC-80	188
<input type="checkbox"/> Codificarea în limbajul BASIC-aMIC	168	<input type="checkbox"/> Particularități ale programării în limbajul BASIC-PLUS	192
<input type="checkbox"/> Codificarea în limbajul BASIC-PRAE	174	<input type="checkbox"/> Particularități ale programării în limbajul ABASIC	199
<input type="checkbox"/> Codificarea în limbajul BASIC-HC-85, TIM S, SPECTRUM	176		
TEMA 4		201	
<input type="checkbox"/> Răspundeți prin DA sau NU la...	201	toare cantităților trimestriale de jucării pentru 500 de articole fabricate de întreprinderea U	207
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc din...	204	<input type="checkbox"/> Calculul procentelor corespunzătoare consumurilor trimestriale pentru 1000 de articole	207
<input type="checkbox"/> Scrieți cite un program BASIC pentru	204		
<input type="checkbox"/> Calculul procentelor corespunzătoare			
SOLUȚIA TEMEI 4		207	
<input type="checkbox"/> Nu răspundem	207	<input type="checkbox"/> Se vor adăuga la programul din tema precedentă următoarele instrucțiuni...	208
<input type="checkbox"/> Nu răspundem	207		
<input type="checkbox"/> Programele BASIC sint	207		

CONVERSAȚIA 5. Proiectarea și realizarea unui program BASIC pentru calculul cantității de lichid dintr-un număr variabil de rezervoare, ale căror raze pot lua numai valori pozitive. Variabile indexate numerice. Vectori de date. Instrucțiunea DIM. Introducerea statică a datelor. Instrucțiunile READ și DATA. Structura de iterație CIT TIMP. Bucla WHILE-WEND. Structura de selecție. Instrucțiunile IF-THEN și IF-THEN-ELSE. Subprograme. Instrucțiunile GOSUB și RETURN. JOCURI, APLICAȚII și TESTE pentru cititor	209
EXEMPLELE 5-PC, m, M, F	210
<input type="checkbox"/> De la problemă la program	210
<input type="checkbox"/> Analiza problemei	210
<input type="checkbox"/> Proiectarea programului	212
<input type="checkbox"/> Codificarea în limbajul BASIC-aMIC	218
<input type="checkbox"/> Codificarea în limbajul BASIC-PRAE	230
<input type="checkbox"/> Codificarea în limbajul BASIC-HC-85, TIM S, SPECTRUM	238
<input type="checkbox"/> Codificarea în limbajul BASIC-AMSTRAD	243
<input type="checkbox"/> Codificarea în limbajul BASIC-COMMODORE	250
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-80	252
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-PLUS	259
<input type="checkbox"/> Particularități ale programării în limbajul ABASIC	262
TEMA 5	267
<input type="checkbox"/> Răspundeți prin DA sau NU la...	267
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc din...	268
<input type="checkbox"/> Scrieți cite un program BASIC pentru...	269
<input type="checkbox"/> Calculul procentelor corespunzătoare cantităților trimestriale de jucării pentru un număr variabil de articole fabricate de întreprinderea U	271
<input type="checkbox"/> Calculul abaterii în procente a consumului realizat față de cel normat pentru un număr variabil de articole	271
SOLUȚIA TEMEI 5	271
<input type="checkbox"/> Nu răspundem	271
<input type="checkbox"/> Nu răspundem	271
<input type="checkbox"/> Programele BASIC sint...	271
<input type="checkbox"/> Pentru scrierea programului s-au definit	274
<input type="checkbox"/> Pentru scrierea programului s-au definit	275
CONVERSAȚIA 6. Proiectarea și realizarea unui program pentru editarea raportului săptămânal privind livrările zilnice de lichide. Variabile indexate alfanumerice. Instrucțiuni matriciale. Funcțiile DEF FN și RANDOMIZE. Instrucțiunea PRINT AT. Programe utile pentru matematică și fizică. JOCURI, APLICAȚII și TESTE pentru cititor	277
EXEMPLELE 6-PC, m, M, F	278
<input type="checkbox"/> De la problemă la program	278
<input type="checkbox"/> Analiza problemei	278
<input type="checkbox"/> Proiectarea programului	282
<input type="checkbox"/> Codificarea în limbajul BASIC-aMIC	297
<input type="checkbox"/> Codificarea în limbajul BASIC-PRAE	311
<input type="checkbox"/> Codificarea în limbajul BASIC-HC-85, TIM S, SPECTRUM	313
<input type="checkbox"/> Codificarea în limbajul BASIC-AMSTRAD	318
<input type="checkbox"/> Codificarea în limbajul BASIC-COMMODORE	325
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-80	326
<input type="checkbox"/> Particularități ale programării în limbajul BASIC-PLUS	337
<input type="checkbox"/> Particularități ale programării în limbajul ABASIC	342
TEMA 6	353
<input type="checkbox"/> Răspundeți prin DA sau NU la...	353
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc din...	354
<input type="checkbox"/> Scrieți cite un program BASIC pentru...	356
<input type="checkbox"/> Calculul abaterii cantității totale trimestriale de jucării (în procente) fabricate de întreprinderea U, față de cantitatea medie trimestrială	360
<input type="checkbox"/> Calculul abaterii consumurilor trimestriale raportate față de consumurile trimestriale normate ale mai multor articole	363

SOLUȚIA TEMEI 6			363
<input type="checkbox"/> Nu răspundem	363	<input type="checkbox"/> Programul pentru calculul abate-	
<input type="checkbox"/> Nu răspundem	363	rării consumurilor trimestriale rapor-	
<input type="checkbox"/> Programele BASIC sint...	363	tate față de consumurile trimes-	
<input type="checkbox"/> Programul pentru calculul abate-		triale normate ale mai multor ar-	369
rării cantității totale trimestriale de		ticole este...	
jucării, în procente, față de can-	367		
titatea medie trimestrială este...			
CONVERSAȚIA 7. Proiectarea și realizarea unui program BASIC pentru crearea unui fișier secvențial de livrări. Instrucțiuni BASIC AMSTRAD, COMMODORE, BASIC-80, BASIC-PLUS, ABASIC pentru crearea fișierelor de organizare secvențială. Programe utile pentru gestionarea fondurilor de date în fișiere. JOCURI, APLICAȚII și TESTE pentru cititor			371
EXEMPLELE 7-PC, m, M, F			372
<input type="checkbox"/> De la problemă la program	372	<input type="checkbox"/> Particularități ale programării în	
<input type="checkbox"/> Specificații de programare	373	limbajul BASIC-80	381
<input type="checkbox"/> Documentația de proiectare	376	<input type="checkbox"/> Particularități ale programării în	
<input type="checkbox"/> Codificarea în limbajul BASIC-		limbajul BASIC-PLUS	382
AMSTRAD	376	<input type="checkbox"/> Particularități ale programării în	
<input type="checkbox"/> Codificarea în limbajul BASIC-		limbajul ABASIC	386
COMMODORE	378		
TEMA 7			388
<input type="checkbox"/> Răspundeți prin DA sau NU la...	388	<input type="checkbox"/> Proiectarea și realizarea unui	
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc		program pentru crearea unui fi-	
din...	388	șier de jucării	391
<input type="checkbox"/> Scrieți cite un program BASIC		<input type="checkbox"/> Proiectarea și realizarea unui	
pentru...	390	program pentru crearea unui fi-	
		șier de consumuri	391
SOLUȚIA TEMEI 7			391
<input type="checkbox"/> Nu răspundem	391	<input type="checkbox"/> Programul BASIC este...	391
<input type="checkbox"/> Nu răspundem	391	<input type="checkbox"/> Programul BASIC este...	391
<input type="checkbox"/> Nu răspundem	391		
CONVERSAȚIA 8. Proiectarea și realizarea unui program BASIC ghidat de un meniu pentru actualizarea, consultarea și listarea unui fișier secvențial de livrări. Instrucțiunile INKEY\$, GET, ERASE, OPENIN, CLOSEIN, WRITE# și ON... GOTO. Funcția EOF. Fișiere cu acces direct. Depanarea unui program. JOCURI, APLICAȚII și TESTE pentru cititor			393
EXEMPLELE 8-PC, m, M, F			394
<input type="checkbox"/> De la problemă la program	394	<input type="checkbox"/> Crearea și întreținerea unui fi-	
<input type="checkbox"/> Specificații de programare	394	șier de livrări în acces direct în	
<input type="checkbox"/> Documentația de proiectare	398	limbajul BASIC-80	414
<input type="checkbox"/> Codificarea în limbajul BASIC-		<input type="checkbox"/> Particularități ale programării în	
AMSTRAD	405	limbajul BASIC-PLUS	417
<input type="checkbox"/> Codificarea în limbajul BASIC-		<input type="checkbox"/> Particularități ale programării în	
COMMODORE	410	limbajul ABASIC	418
<input type="checkbox"/> Particularități ale programării în			
limbajul BASIC-80	413		
TEMA 8			419
<input type="checkbox"/> Răspundeți prin DA sau NU la...	419	<input type="checkbox"/> Proiectarea și realizarea unui	
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc		program (ghidat de un meniu)	
din...	419	pentru întreținerea unui fișier sec-	
<input type="checkbox"/> Scrieți cite un program BASIC		vențial de jucării	422
pentru...	420		

<input type="checkbox"/>	Proiectarea și realizarea unui program (ghidat de un meniu)	pentru întreținerea unui fișier secvențial de consumuri	422
CONVERSAȚIA 9. Proiectarea și realizarea unui program BASIC-PLUS ghidat de un meniu pentru crearea, actualizarea și listarea unui fișier masiv virtual de livrări. Instrucțiunile DIM, OPEN... AS VIRTUAL și CLOSE. JOCURI, APLICAȚII și TESTE pentru cititor			
			423
EXEMPLUL 9-M			
			424
<input type="checkbox"/>	De la problemă la program	<input type="checkbox"/>	Codificarea în limbajul BASIC-PLUS
			425
TEMA 9			
			437
<input type="checkbox"/>	Răspundeți prin DA sau NU la	rea și întreținerea unui fișier masiv virtual de jucării	439
<input type="checkbox"/>	Inlocuiți cuvintele care lipsesc din	<input type="checkbox"/>	Proiectarea și realizarea unui program BASIC-PLUS pentru crearea și întreținerea unui fișier masiv virtual de consumuri
			440
<input type="checkbox"/>	Scrieți câte un program BASIC pentru		
			438
<input type="checkbox"/>	Proiectarea și realizarea unui program BASIC-PLUS pentru crea-		
			440
SOLUȚIA TEMEI 9			
			440
<input type="checkbox"/>	Răspunsurile sint	<input type="checkbox"/>	Nu răspundem
			440
<input type="checkbox"/>	Cuvintele lipsă sint	<input type="checkbox"/>	Nu răspundem
			440
<input type="checkbox"/>	Programele BASIC sint		
			440
CONVERSAȚIA 10. Proiectarea și realizarea programului principal pentru crearea și întreținerea unui fișier masiv virtual de livrări (adăugare, consultare, numărare, ștergere, modificare) care apelează mai multe subrutine al căror format sursă se cunoaște. Instrucțiuni BASIC-PLUS pentru prelucrarea fișierelor masive virtuale de tip șir. Depanarea unui program. JOCURI, APLICAȚII și TESTE pentru cititor			
			441
EXEMPLUL 10-M			
			442
<input type="checkbox"/>	De la problemă la program	<input type="checkbox"/>	Codificarea în limbajul BASIC-PLUS
			452
TEMA 10			
			474
<input type="checkbox"/>	Răspundeți prin DA sau NU la	virtual de jucării cu structură modificată	478
<input type="checkbox"/>	Inlocuiți cuvintele care lipsesc din	<input type="checkbox"/>	Proiectarea și realizarea unui program BASIC-PLUS pentru crearea și întreținerea fișierului masiv virtual de consumuri cu structură modificată
			478
<input type="checkbox"/>	Scrieți câte un program BASIC pentru		
			475
<input type="checkbox"/>	Proiectarea și realizarea unui program BASIC-PLUS pentru crearea și întreținerea fișierului masiv		
			478
CONVERSAȚIA 11. Proiectarea și realizarea unui program BASIC pentru sortarea unui fișier secvențial și a unui fișier masiv virtual de livrări prin metodele: extracție, inserție, SHELL, indexată. Funcții standard pe șiruri de caractere. Instrucțiunea ON...GOSUB. Funcția SWAP. JOCURI, APLICAȚII și TESTE pentru cititor			
			479
EXEMPLELE 11-PC, m, M, F			
			480
<input type="checkbox"/>	De la problemă la program	<input type="checkbox"/>	Particularități ale programării în limbajul BASIC-80
			502
<input type="checkbox"/>	Specificații de programare	<input type="checkbox"/>	Particularități ale programării în limbajul BASIC-PLUS
			505
<input type="checkbox"/>	Documentația de proiectare	<input type="checkbox"/>	Particularități ale programării în limbajul ABASIC
			512
<input type="checkbox"/>	Codificarea în limbajul BASIC-AMSTRAD		
			494
<input type="checkbox"/>	Codificarea în limbajul BASIC-COMMODORE		
			501

TEMA 11			512
<input type="checkbox"/> Răspundeți prin DA sau NU la...	512	niu pentru sortarea fișierului secvențial de jucării	514
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc din...	513	<input type="checkbox"/> Proiectarea și realizarea unui program BASIC-PLUS ghidat de un meniu pentru sortarea fișierului masiv virtual de consumuri	514
<input type="checkbox"/> Scrieți cite un program BASIC pentru...	513		
<input type="checkbox"/> Proiectarea și realizarea unui program BASIC ghidat de un me-			
CONVERSAȚIA 12. Reprezentarea livrărilor de lichide din trei rezervoare cu ajutorul histogramelor. Tehnici de trasare a histogramelor în 2D și 3D. Programarea culorilor. Instrucțiunile grafice WINDOW, VIEWPORT, MOVE, DRAW, RMOVE, RDRAW, PLOT, POINT, CIRCLE. APLICAȚII și TESTE pentru cititor			
			515
EXEMPLELE 12-PC, m			516
<input type="checkbox"/> Elemente de grafică interactivă	516	HC-85, TIM S, SPECTRUM. Trasarea histogramelor prin metoda "punct cu punct"	543
<input type="checkbox"/> Primitive grafice	520	<input type="checkbox"/> Trasarea histogramelor în 2D prin metoda "linie cu linie" cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM	554
<input type="checkbox"/> Să desenăm un roboțel cu calculatoarele aMIC și PRAE	525	<input type="checkbox"/> Trasarea histogramelor în 3D cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM	558
<input type="checkbox"/> Să desenăm un cilindru cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM	532	<input type="checkbox"/> Trasarea histogramelor în 2D și 3D cu microcalculatoarele M118 și TPD	565
<input type="checkbox"/> Culori și grafice	538		
<input type="checkbox"/> Generarea unui rezervor-display cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM. Trasarea histogramelor cu ajutorul caracterelor mozaic	542		
<input type="checkbox"/> Generarea a trei rezervoare-display cu calculatoarele AMSTRAD și			
TEMA 12			572
<input type="checkbox"/> Răspundeți prin DA sau NU la...	572	lunare de jucării fabricate de întreprinderea U	573
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc din...	572	<input type="checkbox"/> Reprezentarea în 3D a histogramei procentelor corespunzătoare consumurilor trimestriale ale unui articol	573
<input type="checkbox"/> Scrieți cite un program BASIC pentru...	573		
<input type="checkbox"/> Trasarea în 2D și 3D a histogramelor corespunzătoare cantităților			
SOLUȚIA TEMEI 12			574
<input type="checkbox"/> Nu răspundem	574	<input type="checkbox"/> Nu răspundem	580
<input type="checkbox"/> Nu răspundem	574	<input type="checkbox"/> Nu răspundem	580
<input type="checkbox"/> Programele BASIC sint...	574		
CONVERSAȚIA 13. Simularea livrărilor de lichide din trei rezervoare cu ajutorul imaginilor animate. Simboluri grafice definite de utilizator. Sprite-uri COMMODORE. Simboluri grafice în mișcare. JOCURI, APLICAȚII și TESTE pentru cititor			
			581
EXEMPLELE 13-PC			582
<input type="checkbox"/> Caractere definite de utilizator pe calculatoarele HC-85, TIM S, SPECTRUM și AMSTRAD	582	<input type="checkbox"/> Programarea imaginilor animate pe calculatorul aMIC	588
<input type="checkbox"/> Generarea sprite-urilor pe calculatorul COMMODORE	583	<input type="checkbox"/> Animație la nivel de caracter pe calculatorul AMSTRAD	590
TEMA 13			591
<input type="checkbox"/> Răspundeți prin DA sau NU la...	591	<input type="checkbox"/> Scrieți cite un program BASIC pentru...	591
<input type="checkbox"/> Înlocuiți cuvintele care lipsesc din...	591		

<input type="checkbox"/> Reprezentarea animată a dinamicii producției de jucării la întreprinderea U	592	<input type="checkbox"/> Reprezentarea animată a consumurilor trimestriale ale unui articol	592
CONVERSAȚIA 14. Proiectarea și realizarea unui simulator pentru o stație de livrare produse lichide. Animație, tehnica WINDOW, semnalizare optică și acustică, reprezentarea în 3D			
			593
TEMA 14			
			594
<input type="checkbox"/> Cerințe de prelucrare	594	<input type="checkbox"/> Cerințe de rezolvare	595
SOLUȚIA TEMEI 14			
			596
<input type="checkbox"/> Proiectarea programului	596	<input type="checkbox"/> Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM	597
* Pași printre conversații și sinteze. Un dialog autor—editor—cititor (II)			606

**TOTUL DESPRE BASIC
ÎN 14 CONVERSAȚII ȘI...**

CONVERSAȚIA 1

Editarea unui text (titlul cărții) cu calculatorul.
Cel mai simplu program în BASIC. Metode pentru
introducerea, adăugarea și corectarea interactivă
a unei linii program. Comenzi BASIC: RUN și LIST

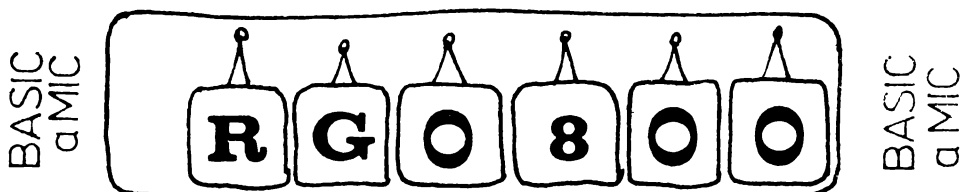


□ Imaginea programului BASIC-aMIC

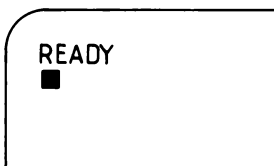
Aprobare pentru un program-READY

Iată cât de simplu se pot întâlni două bucurii. Una a dumneavoastră care doriți să învățați BASIC... conversind cu calculatorul și una a noastră, care ne-am angajat să vă ajutăm, scriind această carte accesibilă tuturor vîrstelor. Cît de prietenoase sînt calculatoarele vă veți convinge chiar din prima conversație, pe care vă invităm să o abordați urmind cu atenție și întocmai textul. Apoi veți vedea și singuri ce va urma.

Treziți, vă rugăm, geniul electronic adormit! Introduceți de la claviatura aMIC-ului dvs. următoarea succesiune de tastări: **RGØ8ØØ***. Acțio-



nați apoi tasta **[RETURN]** și urmăriți ce se afișează pe ecranul televizorului:



Mesajul **READY** indică faptul că sistemul așteaptă să interveniți. El spune de fapt "sînt gata, este rindul dvs. să faceți ceva" sau, altfel zis: "am înregistrat comanda și sînt gata să primesc informațiile de la tastatură".

Pătrățelul de sub **READY** care se deplasează pe ecran în timp ce dvs. tastați litere/cifre/caractere speciale (șir de caractere) este cursorul. El vă permite să știți exact unde va fi tipărit următorul caracter. El spune de fapt "aici este locul unde va apare pe ecran următorul caracter". Dacă apăsați pe bara de jos cursorul se va deplasa fără să tipărească nimic.

Introducerea unei linii program

Modul de lucru cu calculatorul personal aMIC (pentru a-l determina să facă ceva pe cont propriu) este acela de a scrie pentru el un program, de a introduce acest program în calculator și de a determina apoi calculatorul să execute programul introdus. Un program este o listă de instrucțiuni pe care calculatorul le înțelege și le execută singur.

* Caracterul "Ø" reprezintă cifra zero.

Cu aMIC-ul puteți conversa în BASIC, recunoscut pentru ușurința cu care poate fi înțeles, învățat și folosit.

Înainte ca orice program să poată fi parcurs sau executat, el trebuie introdus în memoria calculatorului, fie direct de la claviatură, fie cu ajutorul unei casete sau al unei dischete.

În această conversație vom considera că programele sînt introduse direct de la claviatură. Programele pot fi introduse în calculator numai atunci cînd acesta se află în stare aptă de operare. Aceasta înseamnă că introducerea unui program poate începe numai după apariția pe ecran a mesajului **READY** urmat de simbolul cursor.

Pentru dvs., „experții”, care deja ați parcurs prima conversație, aceste lucruri vă sînt familiare. Pentru aceia dintre dvs. care acum ați început să conversați, să încercăm împreună introducerea și execuția unui program BASIC care să afișeze titlul cărții pe care o lecturați:

```
“INVATAM BASIC . . .
```

```
. . . CONVERSIND CU CALCULATORUL”
```

Să introducem, pentru început, prima linie a programului care să editeze doar “INVATAM BASIC . . .”. Tastați următoarea linie:

```
10 PRINT “INVATAM BASIC . . .”
```

Nu acționați încă tasta **[RETURN]**!

Ștergerea caracterelor unei linii program

Dacă faceți o greșeală este ușor să o corectați, mult mai ușor decît la o mașină de scris. Folosiți tasta **[DEL]**. Atunci cînd ați greșit, ștergeți cu **[DEL]** totul înapoi pînă la caracterul greșit (inclusiv), după care îl retastați. Acum, examinați cu atenție ceea ce ați tipărit:

a) după cuvîntul **PRINT**, ați inclus totul între ghilimele?

b) există cumva ghilimele suplimentare?

Dacă totul este în ordine, apăsați pe **[RETURN]**.

Număr de linie

Priviți, vă rugăm, acest program BASIC. Să ne obișnuim cu „imaginea” lui. El este alcătuit dintr-o singură linie, numerotată cu 10. Un program BASIC este compus dintr-o serie de instrucțiuni ce se execută într-o anumită secvență (ordine). Din acest motiv, liniile programului se vor numera. Deoarece în BASIC-aMIC noi nu putem scrie decît o instrucțiune pe fiecare linie, putem enunța prima regulă:

Regula 1. Într-un program BASIC orice linie trebuie să înceapă cu un număr de linie.

Numerele de linie sînt numere întregi pozitive, cuprinse între zero și un număr destul de mare. Cel mai mare număr de linie posibil pentru calculatorul aMIC este 32767.

Editare șir de caractere: PRINT

Instrucțiunea **PRINT**, în engleză “tipărește”, ce urmează după numărul de linie (10) realizează editarea șirului de caractere: “INVATAM

BASIC..." plasat între ghilimele. Un șir de caractere poate să includă cifre (0; 1; 9), litere (A; B; ...) sau caractere speciale (+; -; *; ↑; ?; ...). Când ghilimelele definesc începutul și sfârșitul unui șir de caractere, ele nu pot fi folosite drept caractere în cadrul șirului.

Din acest exemplu rețineți o nouă regulă:

Regula 2. Fiecare număr de linie trebuie să fie urmat de o instrucțiune.

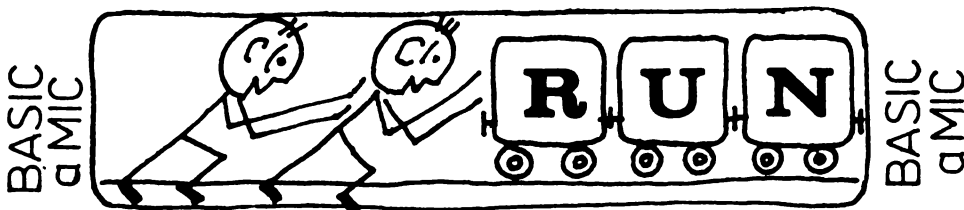
Remarci

- La calculatorul personal aMIC nu aveți nevoie de tasta [SHIFT] pentru obținerea caracterelor majuscule, deoarece scrierea se face numai cu litere mari. Această tastă este necesară pentru tipărirea semnelor desenate deasupra tastelor (prin apăsarea simultană a tastei [SHIFT] și a tastei cu semnul dorit).
- Când tipăriți caractere sau spații, cursorul se mișcă spre dreapta.
- Calculatorul, singur, nu are de unde să știe când ați terminat o linie program; trebuie ca dvs să-i spuneți acest lucru. La aMIC semnalul de terminare constă în apăsarea tastei [RETURN].
- Dacă, totuși, găsiți o greșeală după ce ați tastat o linie de program și ați apăsat [RETURN] nu puteți folosi pentru corecție tasta [DEL]. Deocamdată, vă recomandăm retastarea întregii linii peste ea însăși.
- În BASIC este posibilă "scrierea peste" orice linie a programului prin simpla tipărire a numărului de linie și introducerea versiunii revizuite a instrucțiunii.

Comanda de execuție RUN

Să nu uităm de simbolul cursor. Pare a vă întreba: "Bine, ce urmează?".

Un program – chiar și unul construit dintr-o singură linie – nu este de prea mare folos înainte de a fi executat. În BASIC, un program este executat imediat după ce se tipărește comanda **RUN** și se apasă pe tasta [RETURN].



Deci, tastați **RUN** și apăsați pe [RETURN]. Dacă nu ați făcut nici o greșeală, pe ecranul televizorului veți putea citi:

```

RUN
INVATAM BASIC...
ERROR 01 IN LINE 10
READY

```

Felicitări! Primul dvs. program în BASIC este acum executat, dar...

Un mesaj de eroare

Nu vă alarmați! Pentru a înțelege ce s-a întâmplat vă propunem să analizăm împreună rezultatul execuției programului introdus. Când ne-am adresat calculatorului cu instrucțiunea **PRINT** urmată de "ÎNVĂȚĂM BASIC . . ." i-am cerut să reproducă șirul de caractere plasat între ghilimele. După cum constatați, pe ecranul televizorului s-a tipărit: ÎNVĂȚĂM BASIC dar ghilimelele nu au fost afișate! Pe linia imediat următoare a apărut însă ceva ciudat, un mesaj de eroare spunem noi:

ERROR 01 IN LINE 10

Acest mesaj este rezultatul unei erori interne săvârșite de noi. Ne ajută să aflăm când am realizat un lucru greșit, eronat sau un lucru bun dar la timpul nepotrivit. Codul 01 din linia 10 (de altfel, singura linie din program) ascunde ceva! Ce anume? Consultând lista cu mesaje de erori (vol. 2, pag. 2) a aMIC-ului, în dreptul codului 01 găsim: „programul nu se termină cu instrucțiunea **END** sau **STOP**”.

Aici este momentul oportun pentru introducerea unei instrucțiuni BASIC speciale, instrucțiunea **END**.

Unde se sfârșește programul? Instrucțiunea **END**

Sfârșitul programului este ultima instrucțiune pe care dorim s-o execute calculatorul.

Este probabil evident că instrucțiunea **END** spune calculatorului că s-a atins punctul final al programului respectiv. În mod normal, executarea unui program ce nu conține instrucțiunea **END** produce același rezultat ca și în cazul când programul conține instrucțiunea **END**.

Foarte mulți își fac o obișnuință din a încheia fiecare program cu instrucțiunea **END**, deși nu este întotdeauna necesar să se procedeze astfel. Regulile ce dictează utilizarea lui **END** în majoritatea dialectelor BASIC impun ca **END** să fie ultima instrucțiune din program. De obicei, lui **END** i se dă numărul 99 sau 999 sau 9999, în funcție de cel mai mare număr de linie acceptat de calculator.

Interpretorul BASIC-aMIC cere ca programul să se termine obligatoriu cu una din instrucțiunile **END** sau **STOP**. În caz contrar, calculatorul va afișa mesajul de eroare cu codul 01, ceea ce s-a și întâmplat, de altfel, în cazul programului nostru.

Cum să extindem un program?

Din cauza apariției mesajului de eroare cu codul 01, sîntem nevoiți să lărgim vechiul program (existent în memorie) cu o nouă linie, cea a instrucțiunii **END**. În BASIC fiecare linie a programului trebuie să aibă un număr, calculatorul executînd programul în deplină concordanță cu ordinea numerelor liniilor, începînd cu cel mai mic număr de linie și terminînd cu cel mai mare număr de linie. Cum **END** trebuie să fie ultima instrucțiune din program, vă propunem să tastați:

99 **END**

Verificați cu atenție dacă ați introdus corect noua linie, după care tastați **[RETURN]**. Dacă totul este bine, tastați **RUN** și apăsați pe **[RETURN]**. Sperăm că, de această dată, ați obținut un rezultat fără erori.



```

RUN
INVATAM BASIC...
READY
■

```

Felicitări! Primul dvs. program v-a reușit! După această experiență este bine să rețineți următoarele două reguli importante.

Regula 3. Calculatorul execută un program în deplină concordanță cu ordinea numerelor liniilor, începând cu cel mai mic număr și terminând cu cel mai mare număr de linie.

Regula 4. Liniile programului nu trebuie introduse neapărat în ordine crescătoare. Calculatorul le va executa însă în ordine crescătoare.

Mai multe despre comanda de execuția RUN

Acest program simplu, cu două linii, poate fi executat ori de câte ori dorim, tipărind comanda **RUN** și apăsând pe tasta **[RETURN]**. Încercați să executați programul de trei ori sau de câte ori doriți. Veți obține, evident, același rezultat: INVATAM BASIC . . . , o singură dată.

După studierea acestei mici demonstrații, este bine să remarcați că atâta timp cât programul rămâne intact în memoria calculatorului, el poate fi executat ori de câte ori dorim.

După cum ați putut constata, spre deosebire de liniile program, comanda **RUN** se introduce fără număr de linie, iar după apăsarea tastei **[RETURN]** calculatorul execută imediat această comandă. Liniile de program sînt precedate de un număr de linie, iar după acționarea tastei **[RETURN]** calculatorul reține doar numărul de linie și linia respectivă, fără ca instrucțiunea (entitatea) din linia respectivă să fie executată.

Revenind la instrucțiunea PRINT

Cînd ne-am adresat calculatorului cu instrucțiunea **PRINT** i-am impus reproducerea șirului de caractere inclus între ghilimele. Vă puteți întreba, pe bună dreptate, dacă expresia plasată între ghilimele nu trebuie să aibă neapărat un sens! Răspunsul este nu! Ea poate fi orice combinație de caractere, inclusiv spații, cu o singură excepție – ghilimelele.

Instrucțiunea

```
10 PRINT "1+2=4"
```

"merge" la fel de bine ca și instrucțiunea

```
10 PRINT "INVATAM BASIC . . ."
```

Deci, chiar dacă expresiile cuprinse între ghilimele nu au nici un sens, calculatorul va tipări exact ceea ce i se indică!

Primul test

Să părăsim pentru un moment programul BASIC și să ne concentrăm atenția asupra unui mic test. Verificați dacă puteți răspunde la întrebările puse, înainte de a citi explicațiile noastre.

1. Se consideră următorul program BASIC (pe care vă rugăm să nu-l introduceți în calculator).

```
10 PRINT "LA MULTI ANI FRUMOSI!"
20 END
```

a) Precizați numerele de linie ale programului.

```
. . . . .
```

```
10
```

```
20
```

b) Precizați instrucțiunea care indică sfârșitul programului.

```
. . . . .
```

```
END
```

c) Care este rolul ghilimelelor din cadrul instrucțiunii **PRINT** (linia 10).

Ghilimelele spun calculatorului că șirul de caractere începe cu litera L și se termină cu caracterul special "!"

d) Ce se va afișa pe ecranul televizorului în momentul execuției programului?

```
. . . . .
```

```
LA MULTI ANI FRUMOSI!
```

```
READY
```

```
■
```

e) Dacă ați executa programul de o mie de ori, ce ați constata?

```
. . . . .
```

Se obține același rezultat.

f) Precizați rezultatul execuției programului anterior în cazul cind ați introduceți mai linia 10.

```
. . . . .
```

```
LA MULTI ANI FRUMOSI!
```

```
ERROR 01 IN LINE 10
```

```
READY
```

```
■
```

2. Cum va răspunde calculatorul aMIC în urma execuției următoarelor programe?

a) 10 PRINT " "

```
30 END
```

```
. . . . .
```

Un spațiu, urmat de mesajul **READY**.

b) 10 PRINT "5-3"

```
20 END
```

```
. . . . .
```

5-3, urmat de mesajul **READY**.

c) 10 PRINT "AB#"

```
20 END
```

```
. . . . .
```

```
AB#
```

d) 10 PRINT "2>3"

```
20 END
```

```
. . . . .
```

```
2>3 (!)
```

INIT șterge ecranul

O cantitate excesivă de informații tipărite la televizor vă poate deruta la un moment dat. În consecință, se impune o curățire a ecranului, fără

a afecta însă programul din memoria calculatorului. Acest lucru se poate realiza, în BASIC-aMIC, cu instrucțiunea **INIT**.

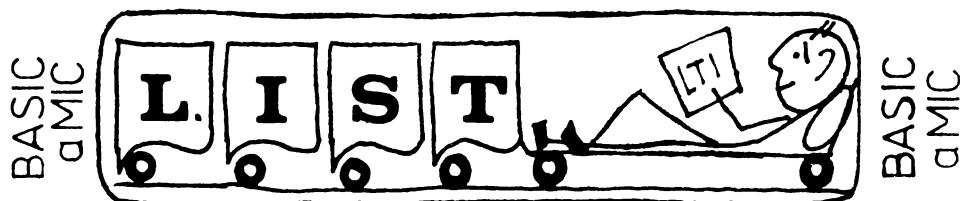
Este oare prea târziu pentru a adăuga această instrucțiune? Nu, bineînțeles că nu. Nu este niciodată prea târziu! Instrucțiunea **INIT** poate fi inserată în programul nostru înaintea liniei 10 prin tipărirea liniei

5 INIT [RETURN]

O nouă linie a programului poate fi inserată în orice loc în cadrul programului și în orice moment, atâta timp cât sistemul este **READY**. Dvs. trebuie să alegeți pur și simplu un număr de linie, astfel încât instrucțiunea introdusă să "cadă" bine, adică acolo unde trebuie. În cazul nostru, ideea a fost de a introduce operația de ștergere a ecranului la începutul programului, așa că numărul de linie poate fi orice cifră între 0 și 9 inclusiv (noi am ales 5). Faptul că numărul de linie ce precede instrucțiunea **INIT** este mai mic decât numărul de linie ce precede pe **PRINT** ne dă certitudinea că **INVATAM BASIC . . .** va fi întotdeauna tipărit după curățirea ecranului și înainte de a se fi executat instrucțiunea **END**.

Cînd scrieți un program BASIC nu este nevoie să introduceți liniile de program în secvență, adică în ordine crescătoare. Calculatorului nu îi pasă dacă instrucțiunile programului i se transmit într-o "ordine" aiurea. El urmărește numărul liniei și nu ordinea în care am introdus noi instrucțiunile.

Pentru a vă lămuri singuri, cereți calculatorului să afișeze întreg programul rezident în memorie.



Cu această ocazie, faceți cunoștință cu o nouă comandă BASIC – **LIST**.

Revederea liniilor din program – comanda **LIST**

Comanda **LIST** permite afișarea programului introdus de către dvs. (tot programul sau numai anumite secvențe) în memoria calculatorului.

Tastați:

LIST [RETURN]

Ce constatați?

```
LIST
5 INIT
10 PRINT "INVATAM BASIC..."
99 END
```

Comanda **LIST** a confirmat faptul că sistemul a acceptat inserarea liniei 5 (și desigur 99). În plus, remarcați modul în care au fost afișate liniile programului – în ordinea crescătoare a numerelor de linie.

Transmiteți calculatorului comanda **RUN**, tastați **[RETURN]** și analizați noul rezultat obținut.

Este altceva, nu-i așa? Într-adevăr, instrucțiunea **INIT** a șters cu "butonetele" tot ecranul dar nu și programul dvs. Dacă nu sunteți convingeți, dați din nou comanda **LIST**.

O sugestie

Faptul că într-un program pot fi scrise noi linii explică de ce am ales pentru instrucțiunile **PRINT** și **END** numerele de linie 10 și 99. Este întru-totul posibilă scrierea programelor cu numere de linie 1, 2, 3 și așa mai departe dar, procedând astfel, nu am mai fi putut lăsa loc liber pentru introducerea de noi linii; și veți rămâne surprinși când veți vedea cât de des veți dori să introduceți într-un program noi linii. De aceea, vă recomandăm să lăsați spațiu între fiecare număr de linie. Numerotarea liniilor cu multipli de 10 este probabil cea mai obișnuită practică, dar și intervalele de 5 sau 20 sînt la fel de bune.

Cînd linia program depășește 30 de caractere

Pentru aceia dintre dvs. care v-ați amintit că ați lăsat robinetul deschis acasă, ne putem opri aici (înainte de a pleca vă rugăm, totuși, să "opriți" calculatorul și televizorul pentru a nu consuma energie electrică!). Alții dintre dvs. au devenit deja pasionați de programare și vor dori să-și continue instruirea cu programarea afișării următorului rînd al titlului cărții "... CONVERSIND CU CALCULATORUL". (Noi nu ne asumăm responsabilitatea dacă întîrziati la masa de seară!).

Pentru a tipări cel de-al doilea șir de caractere ("... CONVERSIND CU CALCULATORUL") va trebui să introducem o nouă linie în programul rezident în memoria calculatorului:

20 PRINT "... CONVERSIND CU CALCULATORUL"

Incepeți, vă rugăm, tastarea instrucțiunii. Ați ajuns la situația din fig. de mai jos?

```
20 PRINT "... CONVERSIND CU CAL
CULATORUL"
```

Nu vă alarmați! Continuați să introduceți și restul caracterelor, după care nu uitați să apăsați pe **[RETURN]**.

Ce s-a întîmplat? După cum ați putut constata linia program nu coincide cu linia de pe ecranul televizorului.

Lungimea liniei de pe ecranul televizorului este de 30 de caractere, în cazul nostru pînă la litera L inclusiv. Restul caracterelor (atenție la noua poziție a cursorului) se introduc automat pe linia următoare, fără a fi nevoie să mai tastăm **[RETURN]**.

Fizic, linia program reprezintă totalitatea caracterelor cuprinse între simbolul emis de calculator (cursor) și terminatorul unei linii **[RETURN]** și nu coincide neapărat cu o linie de pe ecranul televizorului. O linie program poate avea maximum 252 de caractere.

Nu sînteți curioși să vedeți ce ați realizat? Dacă da, tastați **RUN** și **[RETURN]**. Trebuie să obțineți următoarea imagine:

```

INVATAM BASIC...
... CONVERSIND CU CALCULATORUL
READY
■
  
```

Cum vi se par cele două șiruri de caractere pe care le-a afișat calculatorul dvs.? Nu este așa că sînt prea apropiate?

Vă sugerăm să le distanțăm puțin, cu un rînd. Pentru aceasta va trebui să "umblăm" din nou în program și să introducem tot o instrucțiune **PRINT**, dar, de data aceasta, fără nimic altceva în plus!

Apare o problemă – plasarea lui **PRINT!** Apelați în acest caz la comanda care permite revederea liniilor de program, comanda **LIST**.

Editarea unei linii albe: **PRINT**

Tastați **LIST** și apoi **[RETURN]**. Acum vă este mult mai ușor să localizați coordonatele noii instrucțiuni **PRINT** – între liniile 10 și 20. Realizați următoarea succesiune de tastări:

```

15 PRINT [RETURN]
LIST [RETURN]
RUN [RETURN]
  
```

Sperăm că ați obținut o imagine similară celei de mai jos.

```

INVATAM BASIC...

... CONVERSIND CU CALCULATORUL
  
```

Nu-i așa că textul afișat este mult mai lizibil? Totul se datorează instrucțiunii **PRINT** din linia 15. Spunem **PRINT**, dar nu se indică nimic, deci

nu se tipărește nimic, astfel încât se introduce automat un spațiu între cele două linii (10 și 20). Deci, acesta este modul de introducere a unei linii cu spațiu.

Să reținem, în consecință, următoarele noi reguli:

Regula 5. Fiecare instrucțiune PRINT imprimă cite o linie pe suprafața display-ului (televizorului).

Regula 6. În cazul în care linia program nu încapă pe o linie a display-ului (televizorului) se va continua pe linia următoare.

Regula 7. Instrucțiunea PRINT fără nici o opțiune imprimă o linie albă.

Aplicație. Modificați programul anterior, astfel încât pe ecranul televizorului să apară, cu spațierea necesară (v. copertă) și numărul volumului (1, de exemplu) cit și numele editurii – EDITURA TEHNICA. Atenție la aliniere!

Observație. Într-o conversație ulterioară vom relua această aplicație, solicitându-vă și imaginea color a copertii!

TEST

Modificați programul BASIC realizat mai înainte astfel încât el să afișeze următorul text modificat (atenție la noile alinieri):


```

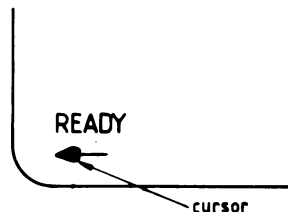
                                INVATAM BASIC . . .
. . . . .CONVERSIND CU CALCULATORUL dMIC
                                                1
                                EDITURA TEHNICA
  
```

Imaginea programului BASIC-PRAE

vol. 2, pag. 198

Obținerea lui READY

Apăsați o tastă oarecare, de preferință cea cu o săgeată pe ea  (retur de car) pe care o vom nota în continuare cu [CR]. Admirați cit doriți emblema de pe ecran și apăsați, la întâmplare, o tastă. Dacă imaginea obținută nu conține mesaje de eroare tastați [CR]. Calculatorul va afișa numărul de octeți disponibili în BASIC-PRAE: 7050 octeți ((7050 BYTES FREE) sau 39818 octeți (39818 BYTES FREE), în funcție de capacitatea memoriei calculatorului dvs. (16 Ko RAM respectiv 48 Ko RAM).



Remarcați cum pe ecran și-au făcut deja apariția **READY** și cursorul – săgeata din colțul din stînga jos care se “agită” tot timpul.

Introducerea și corectarea interactivă a unei linii

Este momentul să introduceți, în aceeași manieră, ca la, calculatorul personal aMIC, prima linie din program. Verificați cu atenție ceea ce s-a tipărit pe ecran.



```
10 PRINT "INVATAM BASICCCC
```

În programul nostru s-a strecurat o greșeală. Ținând ceva mai mult de o secundă degetul pe tasta **[C]**, ea s-a repetat de mai multe ori. Se impune, așadar, ștergerea ultimelor trei caractere și înlocuirea acestora cu trei puncte.

La calculatorul PRAE ștergerea unui caracter din fața cursorului se realizează cu tasta **[DEL]** (**SHIFT** și **Ø**). Spre deosebire însă de aMIC, caracterul nu se șterge de pe ecran, ci este afișat între două semne "\".

Să corectăm împreună linia 10, utilizând pentru ștergerea ultimelor trei caractere tasta **[DEL]**.

Apăsați tasta **[DEL]** și remarcați cum, pentru prima dată, pe ecran a apărut "\C", ceea ce înseamnă că s-a șters litera C (ultima literă). La a doua și a treia apăsare a aceleiași taste au reapărut literele C. Tastați punctul. Remarcați cum pe ecran a apărut mai întâi semnul "\", avertizându-ne că operația de ștergere s-a terminat. Tastați în continuare, de două ori, punctul, apoi ghilimelele.

În momentul când ați terminat de tastat linia program este bine să înștiințați și calculatorul de intenția dvs., prin simpla apăsare a tastei **[CR]**. Vi se pare complicat? Dacă aveți de operat mai multe corecturi și imaginea liniei de program este prea "încărcată", tastați **[CTRL]** și **[R]**.

Pentru editarea în condiții de lizibilitate mărită a titlului complet al lucrării, tastați aceleași instrucțiuni ca și la aMIC.

```
15 PRINT [CR]
```

```
20 PRINT "... CONVERSIND CU CALCULATORUL" [CR]
```

Pentru a vă convinge că, într-adevăr cele două instrucțiuni au fost acceptate, cereți o confirmare lansând, ca și la aMIC comanda **LIST**. Tastați în ordine **RUN**, **[CR]** și analizați rezultatul. Nici o problemă, nu-i așa?

Remarcați pînă în acest punct portabilitatea (de la aMIC la PRAE) a celor două instrucțiuni **PRINT** și a comenzilor **RUN** și **LIST**.

Puteți executa programul ori de câte ori doriți. Rezultatul va fi însă mereu același.

Urmează, în sfârșit, operația de ștergere a ecranului. În BASIC-PRAE acest lucru se poate realiza cu instrucțiunea **CLS**.

Aplicație. Simulați cu creionul și hîrtia răspunsul calculatorului PRAE în urma execuției următorului program:

10 PRINT "INVATAM BASIC..."	[CR]	30 PRINT "PERSONAL PRAE"	[CR]
15 PRINT	[CR]	RUN	[CR]
20 PRINT "... CU CALCULATORUL"	[CR]	INVATAM BASIC...	
25 PRINT	[CR]	... CU CALCULATORUL	
		PERSONAL PRAE	

□ Imaginea programului BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 198

Utilizarea claviaturii

Puneți în stare de funcționare calculatorul personal: HC-85, TIM S, SPECTRUM și faceți cunoștință cu prima imagine (color sau alb-negru) care apare pe monitor (exemplu: H.C.85 I.C.E. FELIX). Deși sinteți nerăbdători să introduceți cit mai repede programul, vă rugăm să mai așteptați puțin. Motivul este unul singur: tastatura! Spre deosebire de celelalte calculatoare cu care ați lucrat pînă acum, tastatura lui HC-85, TIM S, SPECTRUM este profesională, iar cunoașterea ei este o consecință a lucrului intens cu calculatorul.

Să începem prin a învăța s-o utilizăm. Două taste sînt cele mai importante: **[CAPS SHIFT]** și **[SYMBOL SHIFT]**. Identificați-le fără întîrziere pentru o mică demonstrație. Mențineți apăsată tasta **[CAPS SHIFT]** și acționați orice tastă alfabetică. Rezultatul! Toate literele care apar pe ecran sînt scrise cu majuscule (v. poziția 1 din fig. 1.1-a). Mențineți apăsată tasta **[SYMBOL SHIFT]** și acționați, la întîmplare, orice tastă a claviaturii (în afară de **[ENTER]**, **[BREAK SPACE]** sau **[CAPS SHIFT]**). Veți vedea cum pe ecran apar simbolurile din poziția 2 (v. figura 1.1-b) sau cele scrise în roșu (v. SPECTRUM) sub fiecare din tastele care au fost acționate (de exemplu, o săgeată verticală pentru tasta H). Apăsați acum pe o tastă alfabetică oarecare, fără a acționa nici una din tastele **[SHIFT]**. Veți obține instrucțiunile scrise în poziția 3 (v. figura 1.1-c) sau pe cele scrise în alb (v. SPECTRUM) **DIM, IF, NEXT** etc. care permit scrierea programelor BASIC.

Pentru a obține instrucțiunile scrise în poziția 4 (v. figura 1.1-d) sau pe cele scrise cu verde (v. SPECTRUM) deasupra tastelor, acționați cele două taste **[SHIFT]**, după care le eliberați. Acționați, de exemplu, tasta D și veți obține pe ecran cuvîntul **DATA**.

Cuvintele scrise în poziția 5 (v. fig. 1.1-e) sau cele scrise cu roșu (v. SPECTRUM) sub fiecare tastă se obțin prin apăsarea simultană a celor două taste **[SHIFT]**, eliberînd apoi tasta **[CAPS SHIFT]** (albă la SPECTRUM), în timp ce tasta **[SYMBOL SHIFT]** (roșie la SPECTRUM) se menține apăsată, perioadă în care se apasă pe tasta dorită. De exemplu, obțineți cuvîntul **BRIGHT** pornind de la tasta B, **ATTR** cu tasta L și **VERIFY** cu tasta R (v. figura 1.1-e, f, g).

Observație. Aceasta este valabil pentru toate tastele, în afară de funcțiile **EDIT, ENTER, GRAPHICS, DELETE**.

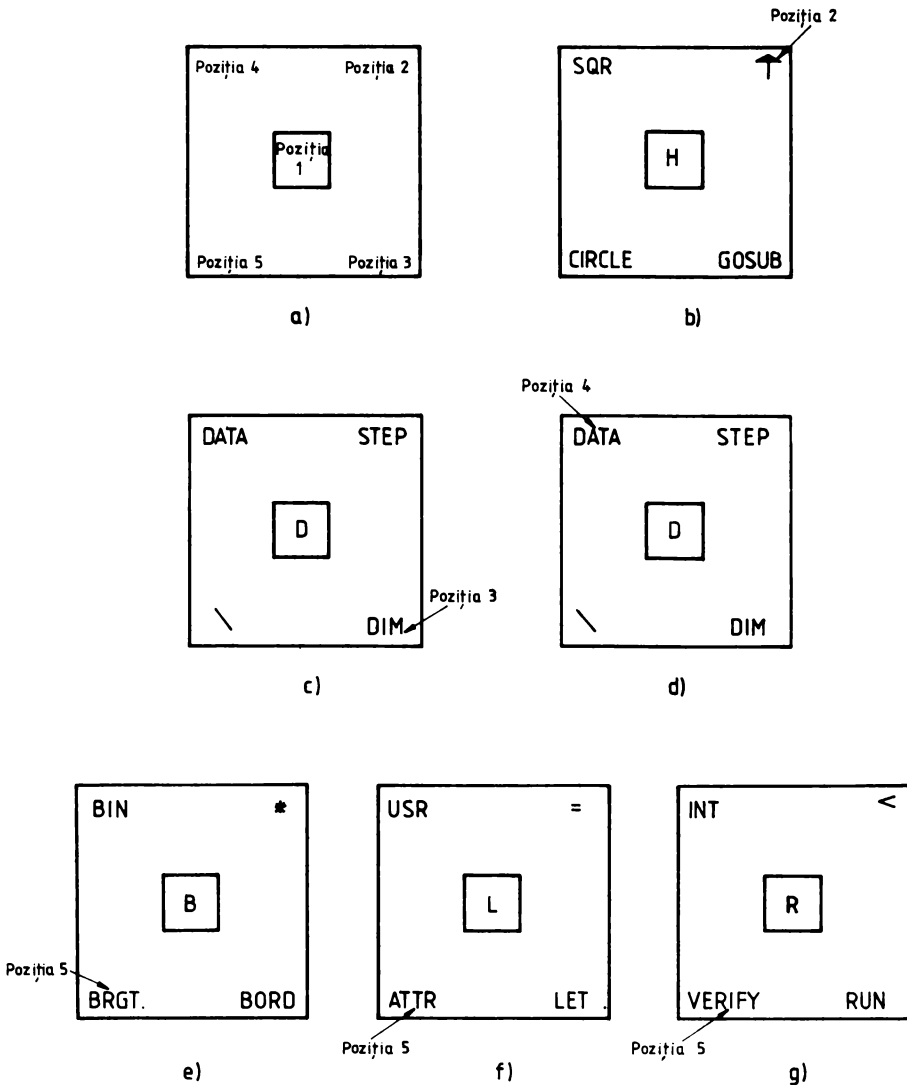


Fig. 1.1. Obținerea funcțiilor și comenzilor cu ajutorul tastelor calculatorului personal HC-85: a) pozițiile unei taste a calculatorului personal HC-85; b) obținerea simbolurilor din poziția 2 [SYMBOL SHIFT] [orice tastă] (v. restricții); c) obținerea instrucțiunilor din poziția 3; d) obținerea instrucțiunilor din poziția 4: se acționează simultan [SYMBOL SHIFT], [CAPS SHIFT], se eliberează [SYMBOL SHIFT] și [CAPS SHIFT] și se acționează tasta care conține instrucțiunea dorită; e, f, g) obținerea funcțiilor din poziția 5: se acționează simultan [SYMBOL SHIFT], [CAPS SHIFT], se ține apăsată tasta [SYMBOL SHIFT] și se acționează tasta care conține funcția dorită.

În figura 1.2 (a, b, c, d, e) sînt prezentate tastele calculatorului personal TIM S corespunzătoare celor din figura 1.1 (b, c, d, e, f, g).

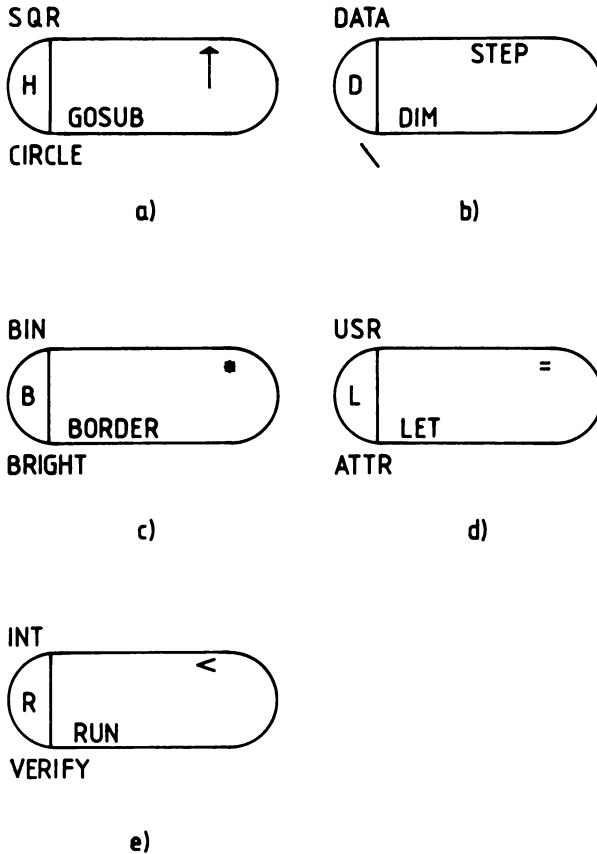


Fig. 1.2. Obținerea funcțiilor și comenzilor cu ajutorul tastelor calculatorului personal TIM S.

EDIT

Această tastă este utilizată atunci cînd se dorește modificarea conținutului liniilor unui program. Deplasînd cursorul (simbolul " $>$ ") și menținînd apăsată tasta **[CAPS SHIFT]**, tasta **[1/EDIT]** permite scrierea liniei de editat la baza ecranului; tastele **[5]** și **[8]** permit deplasarea cursorului în lungul liniei pentru eventualele modificări; sensul deplasării este reprezentat prin săgeata situată deasupra fiecăreia din aceste taste.

ENTER

Această tastă servește validării liniei de program pe care ați editat-o; această linie se atașează astfel în continuarea programului, în par-

tea de sus a ecranului. Dumneavoastră puteți utiliza această instrucțiune numai după ce ea a fost introdusă în memoria calculatorului, tastind **RUN** pentru a putea fi executată.

GRAPHICS

Dacă apăsați pe tasta [**CAPS SHIFT**] iar după aceea tastați 9, veți vedea cum cursorul se va transforma în G. Acționați asupra tastelor numerice și observați ce se afișează pe ecran. Mențineți tasta [**CAPS SHIFT**] apăsată și acționați tastele numerice. Veți vedea cum pe ecran apar simbolurile grafice „complementare” acelorora pe care le-ați obținut mai înainte fără a menține apăsată tasta [**CAPS SHIFT**]. În conversațiile următoare vom vedea și alte utilizări ale modului **GRAPHICS**.

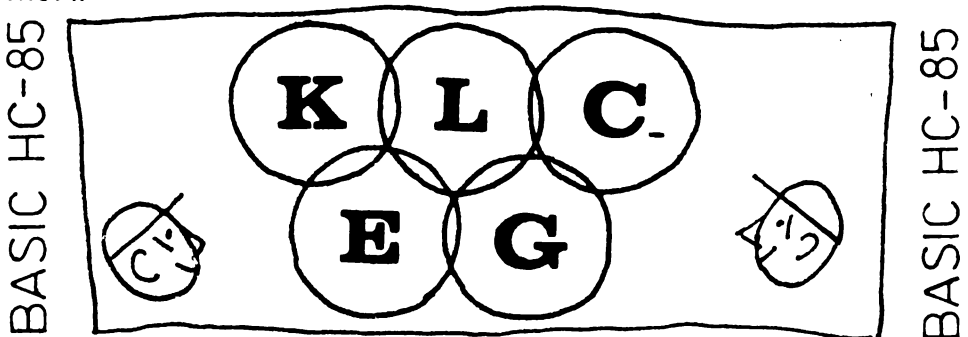
DELETE

După cum indică și numele, tasta permite ștergerea caracterelor nedorite. Apăsați pe [**CAPS SHIFT**], apoi pe tasta O; veți vedea cum linia asupra căreia lucrați va fi ștersă, caracter cu caracter. Nu veți avea nevoie să mențineți apăsată tasta [**CAPS SHIFT**] decît dacă [**DELETE**] va trebui să devină auto-repetitivă.

Desigur, nu trebuie să vă neliniștiți dacă această primă descriere a claviaturii nu v-a relevat imediat toate misterele. Utilizarea intensivă a calculatorului dvs. și descoperirea progresivă a facilităților lui în funcție de nevoile dvs. vă vor permite ca, într-un timp scurt, claviatura să vă devină familiară.

Cinci moduri de lucru: K, L, C, E, G

Este foarte important să înțelegem chiar din prima conversație modul de lucru al calculatorului HC-85, TIM S, compatibil cu SINCLAIR SPECTRUM.



Ați dat binețe simbolului "K" din colțul din stînga jos al ecranului dvs.?

[K]

El este cursorul și vă rugăm, să-i acordați toată atenția. Să nu credeți că el va avea tot timpul "înfățișarea" lui **K**. Nu! Deseori se schimbă în **L**, apoi în **C** și nu puține sînt cazurile cînd poartă chipul lui **E** sau **G**.

Modul K sau modul cuvînt-cheie (în engleză **KEYWORD**=cuvînt cheie) apare numai atunci cînd se așteaptă de la dvs. o comandă sau linie program. După cum constatați, modul **K** reprezintă chiar situația noastră și, în consecință, vă propunem să nu îl părăsim.

Tastați prima linie din program:

10 PRINT "Învățăm BASIC ..."

Nu este chiar atît de complicat pe cît s-ar părea.

Pentru introducerea numărului de linie (10) acționați tastele 1 și 0. Apăsăți apoi pe tasta [P]. Să urmărim ce s-a întimplat între timp pe ecran! Calculatorul a afișat:

10 PRINT L

transformînd, după cum remarcăți, cursorul în **L**, adică "literă".

Modul L (în engleză **LETTER**=literă) este un alt mod de lucru al calculatorului; de regulă, el alternează cu modul **K**.

Facem precizarea că la acționarea unei taste în modul **L** se afișează simbolul principal înscris pe tastă. Dacă se tastează o literă ea va fi tipărită cu minuscule.

Să ne continuăm tastarea cu acționarea simultană a tastelor [**SYMBOL SHIFT**] și [P]. Observați cum pe ecran s-au afișat ghilimelele:

10 PRINT " L "

Tot în modul **L** vom introduce și "Învățăm", cu precizarea că primul caracter se va tasta cu majuscule.

Apăsăți simultan [**CAPS SHIFT**] și [I], după care tastați restul caracterelor, inclusiv spațiul **SPACE** ce urmează cuvîntului introdus.

10 PRINT "Învățăm L "

În continuare, vom afișa cu majuscule cuvîntul **BASIC**. Modul în care se realizează scrierea cu litere mari este **modul C** (în engleză **CAPITALS**=capitale), o variantă a modului **L**. Trebuie deci schimbată înfățișarea cursorului din **L** în **C**. Tastați simultan [**CAPS SHIFT**] și [2]. Acționați în continuare asupra tastelor care conțin caracterele **B**, **A**, **S**, **I** și **C**. Remarcați cum, din acest moment, ne aflăm cu cursorul în modul **C**.

10 PRINT "Învățăm BASIC C "

Urmează să introduceți cele trei puncte prin apăsarea simultană a tastelor [**SYMBOL SHIFT**] și [M]. Nu uitați ghilimelele! ([**SYMBOL SHIFT**], [P]).

10 PRINT "Învățăm BASIC ... " C

Înainte de a apăsa pe [**ENTER**] verificați cu atenție corectitudinea celor introduse. Dacă aveți "neplăceri" folosiți tasta [**DELETE**] ([**CAPS SHIFT**], [0]). Ea permite ștergerea unui caracter din stînga cursorului.

După ce ați eliminat și eventualele caractere eronate ("scăpate" la tastare), apăsați pe [**ENTER**] și remarcați cum între numărul de linie (10) și **PRINT** s-a intercalat simbolul ">" pe care îl vom numi cursorul programului.

```
10>PRINT "Învățăm BASIC..."
```



Simbolul ">" marchează, de regulă, linia curentă. Este vorba în principal de ultima linie introdusă.

Observație. Pentru urcarea și coborîrea simbolului ">" puteți utiliza tastele [↑] respectiv [↓].

Nu-i așa că dialogul pe care îl purtați acum cu HC-85, TIM S, SPECTRUM a devenit incitant? Pînă și tastele par a fi mai prietenoase. Vă propunem să introducem în continuare cea de-a doua linie a programului:

```
20 PRINT "... conversînd cu calculatorul" [ENTER]
```

Tastați:

R [ENTER]

și priviți cum pe ecranul monitorului s-au afișat, fără spațiu între ele, cele două rînduri care alcătuiesc titlul lucrării.

```
Învățăm BASIC...
... conversînd cu calculatorul
```



Pentru tipărirea titlului cărții în condiții de lizibilitate mărită, tastați în continuare următoarele comenzi și instrucțiuni BASIC:

```
LIST [ENTER]
5 CLS [ENTER]
LIST [ENTER]
```

```
15 PRINT [ENTER]
LIST [ENTER]
RUN [ENTER]
```

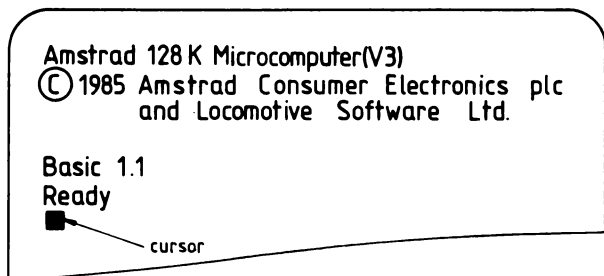
Observație. Modul **E** (în engleză EXTENDED=extins) este utilizat pentru a obține caractere noi, în special alte comenzi. Modul **G** (în engleză GRAPHICS =grafic) este folosit în programarea grafică. Atât modul **E** cât și modul **G** vor face obiectul unor conversații viitoare.

□ Imaginea programului BASIC-AMSTRAD

vol. 2, pag. 198

Introducere în BASIC-AMSTRAD

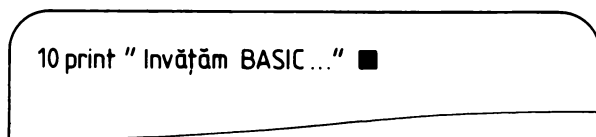
Remarcați cum, odată cu "trezirea" AMSTRAD-ului, pe ecranul monitorului dvs. se afișează cu repeziciune următorul text:



Ceea ce surprinde în mod plăcut la acest calculator este dorința lui imediată de conversație, fără nici un alt protocol. El este gata (**READY**) de a primi în ospeție programul dvs. Pentru a vă obișnui cu noua gazdă începeți prin a introduce prima linie:

10 print "Învățăm BASIC..."

Ați obținut și dvs. această imagine?



Dacă nu, înseamnă că la tastare s-au strecurat greșeli. Vă puteți folosi în acest caz de tasta **[DEL]** care permite ștergerea oricărui caracter din stînga cursorului. Nu este nimic complicat!

Să mai rămînem totuși la această imagine și pentru motivul unor explicații suplimentare, dacă considerați că ele vă sînt utile.

Ce vi se pare mai nou, atât la tipărirea instrucțiunii **PRINT** cât și la tipărirea anumitor caractere din cadrul mesajului?

Deoarece atât litera **I** de la începutul mesajului cât și literele ce alcătuiesc cuvîntul **BASIC** au fost tastate cu **[SHIFT]**, ele au fost reproduse, evident, cu majuscule. Cît privește instrucțiunea **PRINT**, ea a fost tipărită cu minuscule, întrucît la tastare nu am "forțat" **[SHIFT]**-ul.

Observație. Pentru introducerea unei instrucțiuni în BASIC-AMSTRAD puteți utiliza indiferent ce caractere doriți: MAJUSCULE sau minuscule.

Dacă sînteți mulțumiți de modul în care ați introdus prima linie, tastați **[RETURN]** sau **[ENTER]** și apoi executați programul cu:

run **[RETURN]**

Veți vedea cum pe ecran a apărut titlul lucrării:

```
Invățăm BASIC...
Ready
■
```

Observație. Dacă doriți, puteți înlocui **PRINT** cu semnul întrebării:

10 ? "Invățăm BASIC..." **[RETURN]**

Cum vi se pare AMSTRAD-ul? Vă place să conversați împreună? Ne-ar surprinde un răspuns negativ. De aceea, vă și invităm să tastați în continuare următoarea succesiune de instrucțiuni și comenzi:

```
15 ? [RETURN]
20 ? "... conversînd cu calculatorul" [RETURN]
list [RETURN]
run [ENTER]
```

Dacă ați urmărit atent dialogul cu calculatorul AMSTRAD, ați remarcat cum la comanda **list** transmisă de dvs., el a răspuns prin a tipări cu majuscule instrucțiunile **PRINT** pe care dvs. le-ați introdus cu semnul întrebării.

Pentru ștergerea ecranului, introduceți instrucțiunea:

5 cls **[RETURN]**

În continuare se prezintă "textul" conversației cu calculatorul AMSTRAD; remarcați cum, la ultima comandă **RUN**, pe ecranul monitorului șters de instrucțiunea **CLS**, s-a afișat ceea ce, de altfel, ne-am propus: titlul cărții!

<pre>Ready 10 ? "Invățăm BASIC..." run Invățăm BASIC Ready 15 ? 20 ? "... conversînd cu calculatorul" list 10 PRINT "Invățăm BASIC..." 15 PRINT 20 PRINT "... conversînd cu calculatorul" Ready run</pre>	<pre>Invățăm BASIC... ... conversînd cu calculatorul Ready 5 cls list 5 CLS 10 PRINT "Invățăm BASIC..." 15 PRINT 20 PRINT "... conversînd cu calculatorul" Ready run Invățăm BASIC... ... conversînd cu calculatorul Ready ■</pre>
--	---

Trei moduri de afișaj

Calculatorul personal AMSTRAD dispune de trei moduri de afișaj: Mode 0, Mode 1 și Mode 2. În mod automat, calculatorul lucrează în Mode 1. Pentru a înțelege cele trei moduri de afișaj, vă propunem o mică demonstrație în trei pași.

Pasul 1. Apăsați pe orice tastă, de exemplu [1], pînă cînd obțineți două linii cu 1. Numărați aparițiile cifrei 1 pe una din linii. Ați găsit 40? Tastați în continuare [RETURN] sau [ENTER]. Veți obține un mesaj de eroare ((Syntax Error)). Nu vă alarmați! Am ales cel mai scurt drum pentru a reveni la **READY**.

Pasul 2. Tastați:
mode 0 [RETURN]

Veți observa cum pe ecranul monitorului se obțin caractere mult mai mari. Apăsați din nou pe tasta [1] pînă obțineți două linii pline. Vă rugăm să le numărați. Ați găsit 20 ? Tastați [RETURN].

Pasul 3. Tastați:
mode 2 [RETURN]

Apăsați pe tasta [1] pînă obțineți două linii pline. Dacă numărați cu atenție aparițiile acestei cifre într-una din linii veți găsi 80. Remarcați cum scrisul (corpul de literă), în comparație cu cazurile precedente, apare mult mai mic.

În concluzie:

Mode 0=20 coloane
Mode 1=40 coloane
Mode 2=80 coloane

Observație. Pentru a pune calculatorul la zero (**Mode 1**) apăsați pe [CONTROL] [SHIFT] și [ESC] în această ordine.

Aplicație. Executați programul BASIC creat anterior utilizînd toate cele trei moduri de afișaj, specifice calculatorului AMSTRAD.

a) Modul zero

```
mode 0 [RETURN]          20 ? "...conversînd cu calculatorul"
5 cls [RETURN]          [RETURN]
10 ? "Învățăm BASIC..." [RETURN]  run [RETURN]
15 ? [RETURN]           [CONTROL], [SHIFT], [ESC]
```

b) Modul unu

```
mode 1 [RETURN]         20 ? "...conversînd cu calculatorul"
5 cls [RETURN]         [RETURN]
10 ? "Învățăm BASIC..." [RETURN]  run [RETURN]
15 ? [RETURN]          [CONTROL], [SHIFT], [ESC]
```

c) Modul doi

```
mode 2 [RETURN]         20 ? "...conversînd cu calculatorul"
5 cls [RETURN]         [RETURN]
10 ? "Învățăm BASIC..." [RETURN]  run [RETURN]
15 ? [RETURN]          [CONTROL], [SHIFT], [ESC]
```

În BASIC-AMSTRAD cele trei moduri de afișaj pot fi obținute cu instrucțiunea **MODE**, similară comenzii **MODE**, cu singura deosebire că in-

strucțiunea **MODE**, ca de altfel orice instrucțiune BASIC, trebuie să fie precedată de un număr de linie.

În cele ce urmează, vă propunem să testați și facilitățile acestei instrucțiuni, foarte des utilizate în programarea pe calculatorul AMSTRAD, după cum urmează:

3 mode 0 [RETURN]	3 mode 1 [RETURN]
5 cls [RETURN]	list [RETURN]
10 ? "Învățăm BASIC..." [RETURN]	run [RETURN]
15 ? [RETURN]	3 mode 2 [RETURN]
20 ? "...conversind cu calculatorul" [RETURN]	list [RETURN]
run [RETURN]	run [RETURN]

Observație. Pentru a vă reintoarce în **MODE 1** nu uitați să tastați **[CONTROL]**, **[SHIFT]**, **[ESC]** în această ordine.

În cele ce urmează se prezintă textul îmbunătățit al conversației purtate cu calculatorul AMSTRAD.

Ready	10 ? "Învățăm BASIC..."
3 mode 0	15 ?
5 cls	20 PRINT "...conversind cu calculatorul"
	run

Invatam BASIC...
...conversind cu calculatorul

Ready	10 PRINT "Învățăm BASIC..."
3 mode 1	15 PRINT
list	20 PRINT "...conversind cu calculatorul"
3 MODE 1	Ready
5 CLS	run

Învățăm BASIC...
...conversind cu calculatorul

Ready	10 PRINT "Învățăm BASIC..."
3 mode 2	15 PRINT
list	20 PRINT "...conversind cu calculatorul"
3 MODE 2	Ready
5 CLS	run



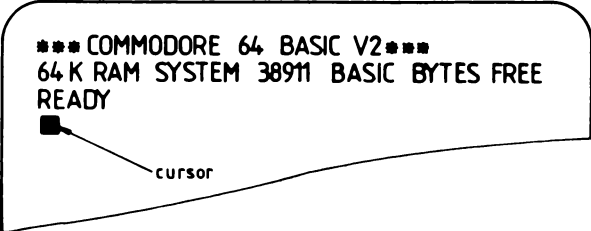
Învățăm BASIC...
...conversînd cu calculatorul

□ Imaginea programului BASIC-COMMODORE

vol. 2, pag. 198

Accesul la calculator

Odată cu "trezirea" calculatorului COMMODORE, pe ecranul monitorului se afișează mesajul:



*** COMMODORE 64 BASIC V2***
64 K RAM SYSTEM 38911 BASIC BYTES FREE
READY

cursor

Sinteți pregătit pentru introducerea programului? Cunoașteți bine tastatura calculatorului COMMODORE? Să începem prin a tasta prima linie:

```
10 PRINT "ÎNVĂȚĂM BASIC..." [RETURN]
```

De notat că **PRINT**, cea mai utilizată instrucțiune în BASIC poate fi înlocuită cu semnul întrebării (?).

Tastați instrucțiunile și comenzile următoare, apăsînd de fiecare dată pe tasta **[RETURN]**.

```
RUN [RETURN]
```

```
15 PRINT [RETURN]
```

```
20 PRINT "... CONVERSÎND
```

```
CU CALCULATORUL"
```

```
[RETURN]
```

```
LIST [RETURN]
```

```
RUN [RETURN]
```

```
5 PRINT CHR$(147) [RETURN]
```

```
RUN [RETURN]
```

Instrucțiunea **PRINT CHR\$(147)** din linia 5 servește la ștergerea ecranului, dar nu și a programului. Pentru ștergerea unui caracter eronat dintr-o linie, veniți cu cursorul pe locul dorit (**[SHIFT]**, **[CRSR]**), și acționați **[SHIFT]**, **[INST]**.

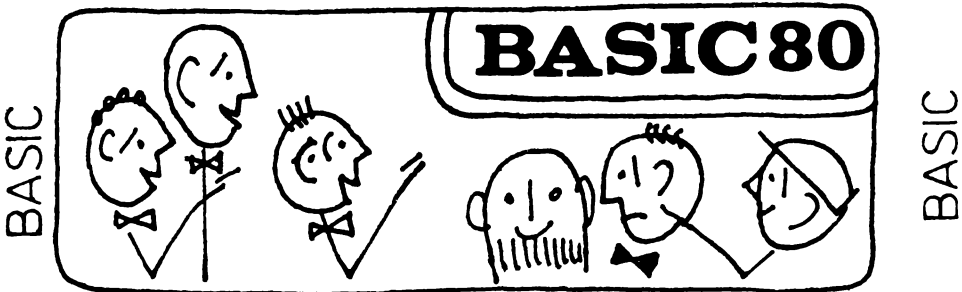
□ Imaginea programului BASIC-80*, GW-BASIC pentru Felix PC

vol. 2, pag. 199

Operare, comenzi și instrucțiuni

Aveți în fața dvs. un microcalculator românesc: M118, JUNIOR, TPD. Înțelegem emoția conversației cu aceste calculatoare dar nu aveți de ce să vă fie... teamă. V-ați acomodat cu tastatura? Ați identificat **[SHIFT]**-ul, **[RETURN]**-ul și celelalte taste mai importante? Aveți definiți toți pașii pe care trebuie să-i parcurgeți singuri? Dacă nu, să-i listăm împreună: 1) încărcarea interpretorului BASIC-80; 2) obținerea lui **READY**; 3) introducerea primei linii a programului; 4) ștergerea (dacă este cazul) a caracterelor unei linii program; 5) execuția programului; 6) extinderea programului; 7) listarea programului; 8) ștergerea ecranului.

Și-acum, la muncă!

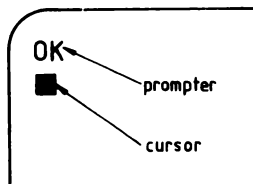


Introduceți în unitatea de floppy discheta care conține interpretorul BASIC-80. Tastați comanda CP/M:

A>BASIC 80 [RETURN]

și așteptați apariția pe ecranul monitorului a mesajului „OK” (omologul lui **READY**). Nu ignorați prezența acestuia! Din acest moment puteți introduce prima linie program:

10 PRINT "Învățăm BASIC..." [RETURN]



Dacă în momentul tastării liniei program ați remarcat apariția unor caractere eronate, există posibilitatea corectării lor (v. conversația 3, comanda **EDIT**). Pentru introducerea acestei linii în memoria calculatorului tastați **[RETURN]**.

* BASIC-80 asemănător cu GW-BASIC, de pe Felix PC (IBM PC). Acesta este tratat în detaliu în vol. 2, pag. 66. Observația este comună programelor BASIC-80 din întreaga lucrare.

Este bine să rețineți că în BASIC-80 numărul liniei este cuprins între 0 și 65529 iar o linie program poate avea maximum 255 caractere. Cit privește instrucțiunea **END**, prezența acesteia nu este obligatorie.

Executați programul cu comanda:

RUN [RETURN]

și analizați primul rezultat.

În vederea extinderii programului cu cele două instrucțiuni **PRINT**, tastați în continuare

15 **PRINT [RETURN]**

20 **PRINT "... conversind cu calculatorul" [RETURN]**

Listați programul cu comanda

LIST [RETURN]

și introduceți instrucțiunea

5 PRINT CHR\$(24) [RETURN]

pentru ștergerea ecranului. Nu uitați denumirea "buretelui de ștergere" în BASIC-80 – **CHR\$(24)** ce se atașează unei instrucțiuni **PRINT**.

În continuare, puteți urmări întreaga conversație purtată în BASIC-80 cu microcalculatoarele românești M118, JUNIOR, TPD.

OK

10 **PRINT "Învățăm BASIC ..."**

RUN

Învățăm BASIC ...

OK

15 **PRINT**

20 **PRINT "... conversind
cu calculatorul"**

LIST

10 **PRINT "Învățăm BASIC ..."**

15 **PRINT**

20 **PRINT "... conversind
cu calculatorul"**

OK

RUN

Învățăm BASIC ...

... conversind cu calculatorul

OK

5 **PRINT CHR\$(24)**

LIST

5 **PRINT CHR\$(24)**

10 **PRINT "Învățăm BASIC ..."**

15 **PRINT**

20 **PRINT "... conversind
cu calculatorul"**

OK

RUN

Învățăm BASIC ...

... conversind cu calculatorul

OK

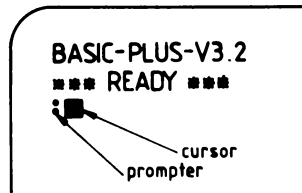
**Imaginea programului BASIC-PLUS,
pentru INDEPENDENT, CORAL**

vol. 2, pag. 199

READY, prompter, cursor

Pentru a pune "la treabă" calculatoarele INDEPENDENT și CORAL trebuie să obțineți mai întâi: ***** READY ***** și simbolul „ : ” pe care îl vom numi **prompter**. **Prompterul** reprezintă modul de exprimare al celor două calculatoare: "mergeți înainte, faceți ceva!" (în engleză, to prompt=a împinge, a îndemna). Spre deosebire de mesajul **READY**, prompterul apare de fiecare dată când sistemul așteaptă o nouă linie de program, chiar dacă

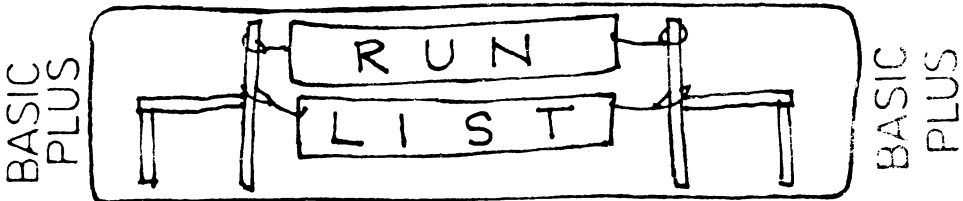
linia respectivă se întâmplă să fie prima sau nu. Remarcați cum, sub forma unei linii subțiri, și-a făcut apariția și cursorul.



Deci, în rezumat, cînd calculatoarele INDEPENDENT și CORAL afișează pe ecran **READY**, ele spun de fapt: "am înregistrat comanda și sîntem gata să acceptăm o nouă linie de program". Simbolul „:” denumit **prompter** se identifică cu "sînt pregătit să accept o nouă linie de program", iar simbolul cursor are semnificația știută: "aici este locul unde următorul caracter va apare pe ecran".

Singuri printre ... comenzi și instrucțiuni

Cum și calculatoarele INDEPENDENT și CORAL se "hrănesc" tot cu programe, considerăm oportună introducerea de la tastatură a programului nostru. De această dată vă vom lăsa singuri, invitîndu-vă să tastați următoarea succesiune de instrucțiuni și comenzi:



```
10 PRINT "ÎNVĂȚĂM
    BASIC ..." [RETURN]
99 END [RETURN]
RUN [RETURN]
15 PRINT [RETURN]
```

```
20 PRINT "... CONVERSÎND
    CU CALCULATORUL" [RETURN]
LIST [RETURN]
RUN [RETURN]
```

Prezentăm în continuare întreaga conversație:

```
: 10 PRINT "ÎNVĂȚĂM BASIC ..."
: 99 END
: RUN
INVĂȚĂM BASIC ...
END OF PROGRAM AT LINE 99
: 15 PRINT
: 20 PRINT "... CONVERSÎND
    CU CALCULATORUL"
: LIST
```

```
10 PRINT "ÎNVĂȚĂM BASIC ..."
15 PRINT
20 PRINT "... CONVERSÎND CU
    CALCULATORUL"
99 END
: RUN
ÎNVĂȚĂM BASIC ...
... CONVERSÎND CU
    CALCULATORUL
END OF PROGRAM AT LINE 99
```

Cum vi s-a părut începutul acestei ... aventuri? Chiar dacă nu ați avut "probleme", câteva precizări sînt totuși necesare:

Reguli

- Ștergerea caracterelor unei linii program se realizează prin acționarea tastei [DELETE] (RUB OUT). Fiecare acționare determină ștergerea unui caracter.
- Semnalul de terminare a unei linii program constă în apăsarea tastei [RETURN].
- Listarea unui program (sau a unei secțiuni din el) se realizează cu comanda LIS[T] (T este opțional).
- Execuția unui program existent în memoria internă se realizează cu comanda RUN.
- O linie BASIC se poate continua pe mai multe linii terminal. Marcarea continuării liniei program curente pe linia terminal următoare se indică prin simbolul "&" [RETURN] plasat în penultima poziție a liniei care se continuă.

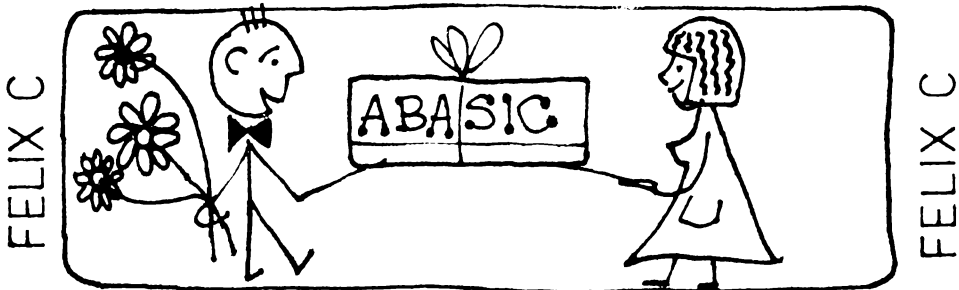
□ Imaginea programului ABASIC, pentru FELIX C

vol. 2, pag. 199

Comanda ABASIC

Interpretorul BASIC sub ARIEL, cunoscut sub numele de ABASIC poate fi lansat de către un utilizator ARIEL (aflat în stare EDITOR) prin comanda:

A [BASIC] [/**<nume fișier>**]/[**<nr. linie>**]]



unde, **<nume fișier>** (dacă este prezent) reprezintă numele fișierului în care se găsește programul BASIC, iar **<nr. linie>** (dacă este prezent) indică linia din fișierul de intrare – program de la care se începe execuția programului.

Linia ABASIC

Linia ABASIC conține maximum 80 de caractere, provenite din articolele unui fișier de intrare sau de la terminal. Numărul de linie precede instrucțiunile, indiferent dacă linia ABASIC provine dintr-un fișier sau de la terminal. Numărul de linie ABASIC poate lua valori cuprinse între 1 și 32 000. Numim program ABASIC secvența de linii ABASIC terminată printr-o instrucțiune **END** sau prin sfârșit de fișier de intrare-program, ordonată în mod crescător după numărul de linie, indiferent de ordinea în care au fost introduse liniile.

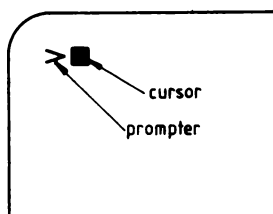
Moduri de lucru în ABASIC

Există trei moduri de lucru în ABASIC: 1) **execuție imediată** (ABASIC așteaptă sau prelucrează o instrucțiune imediată sau o comandă de la terminal); 2) **execuție program** (ABASIC execută un program încărcat, din-

tr-un fișier de intrare sau terminal, ca urmare a unei comenzi sau instrucțiuni); 3) **schimb de date cu terminalul** (programul scrie/citește date la/de la terminal). Trecerea de la un mod de lucru la altul se face prin: comanda **A [BASIC]**, instrucțiunile **STOP**, **END**, apariția unor erori netratate, **BREAK**, comenzile **RUN** și **CON**, orice instrucțiune de scriere/citire etc.

Introducerea și execuția unui program ABASIC

Linia program ABASIC poate fi tastată la terminal numai după apariția prompterului ">". Pentru ca linia să fie introdusă în memoria calculatorului, se va acționa tasta [RETURN]. Pentru listarea și execuția programului folosiți comenzile **LIST** respectiv **RUN**. În continuare, se prezintă imaginea programului ABASIC executat pe calculatorul FELIX C-512.



Observații

- În ABASIC instrucțiunea **PRINT** poate fi înlocuită cu semnul întrebării (?).
- Pentru ștergerea ecranului utilizați instrucțiunea **CLEAR\$** sau **CHR\$ (H1b)**.

> 10 ? "INVĂȚĂM BASIC ..."	10 PRINT "INVĂȚĂM BASIC ..."
> 90 END	15 PRINT
> RUN	20 PRINT "... CONVERSIND
INVĂȚĂM BASIC ...	CU CALCULATORUL"
STOPPED AT LINIE 90	90 END
>	>
> 15 ?	> RUN
> 20 ? "... CONVERSIND	INVĂȚĂM BASIC ...
CU CALCULATORUL"	... CONVERSIND CU
> LIST	CALCULATORUL
	STOPPED AT LINE 90
	>

TEMA 1

Întrebări de control din text

Răspundeți, ajutându-vă de textul nostru, la următoarele întrebări:

1) Există numeroase posibilități de comunicare între om și calculator. Precizați care din următoarele variante permit introducerea unui program în calculator: a) ecranul de televizor; b) claviatura; c) joystick.

2) Numiți cel puțin două funcțiuni îndeplinite de numărul de linie al unui program BASIC.

3) Definiți următorii termeni:

a) prompter; b) cursor; c) șir de caractere.

4) Care este semnificația apariției mesajului **READY** pe monitor? Care este diferența dintre **READY** și simbolul cursor/prompter?

5) Care este diferența dintre o comandă și o instrucțiune BASIC?

6) Descrieți modul în care este posibil să corectați o instrucțiune/comandă înainte de a fi apăsat pe **[RETURN]**.

7) Descrieți procedura pentru ștergerea completă a unei linii din program.

8) Descrieți procedura pentru inserarea unei noi linii de program într-un program BASIC existent.

9) Descrieți procedura pentru schimbarea unei linii de program existente, după ce a fost deja introdusă în memoria calculatorului.

10) Se consideră următorul program BASIC:

```
1 PRINT "A"           180 PRINT "A + B"
10 PRINT "B"          9999 END
```

După cum observați, în acest exemplu numărul de linie poate lua valori de la 1 la 9999. Limita superioară diferă de la calculator la calculator. Precizați această valoare atât pentru calculatorul dvs. cât și pentru calculatoarele CORAL, FELIX C, HC-85, AMSTRAD, aMIC și PRAE.

11) Numerele din fața liniilor nu sînt lipsite de importanță. Ele precizează calculatorului ordinea în care trebuie folosite instrucțiunile din program. Nu este necesar ca aceste numere să se succedă în secvența 1, 2, 3, 4, ... Este mult mai bine și chiar se recomandă ca numerotarea liniilor să se facă din 10 în 10 așa cum de altfel am procedat și noi. Aceasta este o metodă comună de secvențare. Apoi, dacă dorim adăugarea unor noi linii, ele pot fi mult mai ușor introduse printre cele deja existente. Precizați cite linii pot fi introduse între liniile 30 și 40 ale următorului program.

```
10 PRINT "LINIA 10"    40 PRINT "LINIA 40"
20 PRINT "LINIA 20"    50 PRINT "LINIA 50"
30 PRINT "LINIA 30"    99 END
```

12)

```
20 PRINT                5 CLS
80 PRINT "DIMINETI CU FERESTRE" 10 PRINT "DIMINEATA DE SEPTEMBRIE"
   DESCHISE"
```

Acesta este un program simplu ce cuprinde linii. Fiecare linie începe cu Încercuiți numărul de linie al instrucțiunii pe care calculatorul o va executa prima dată. Precizați funcțiunea acestei instrucțiuni.

13) Dacă vă aflați în fața calculatorului personal PRAE imediat după "trezirea" acestuia și doriți să introduceți acest scurt program:

```
5 CLS                20 PRINT
10 PRINT "TEST BASIC" 30 PRINT "BASIC PRAE"
```

procedați astfel:,,,,,,,,,

Să presupunem că ați introdus programul și în continuare doriți să-l și executați. Tastați comanda și apăsați pe tasta Dacă nu

CONVERSAȚIA 2

Calculul ariei unui rezervor sferic de rază dată.
Constante și variabile numerice. Expresii numerice.
Instrucțiunile LET și PRINT. Comenzi BASIC: SCR,
NEW. TESTE și APLICAȚII pentru cititor



EXEMPLELE 2 – PC, m, M, F

□ Cîteva operații fundamentale ale calculatoarelor și descrierea lor în BASIC-aMIC

vol. 2, pag. 199

Operații numerice

Poate să realizeze aMIC-ul operații matematice? Da, poate! Aritmetica elementară este ușoară pentru el. Poate efectua chiar calcule matematice complicate dar și efortul de programare este mai mare.

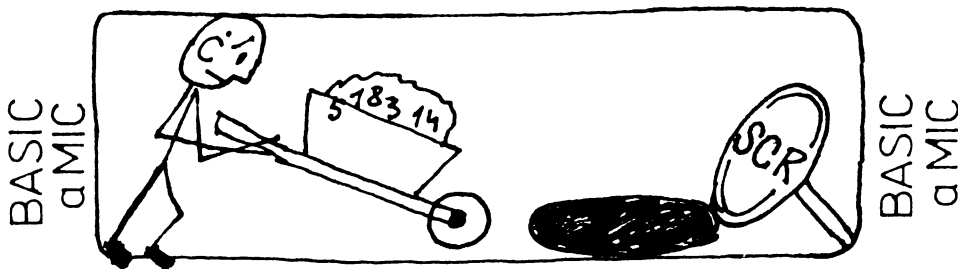
BASIC-ul folosește cele patru operații matematice fundamentale: adunarea (+); scăderea (-); înmulțirea (*); împărțirea (/) și include, de asemenea, un număr de funcții exponențiale, trigonometrice etc.

Semnul "=" este și el necesar. BASIC-ul este puțin particular în ceea ce privește utilizarea acestui semn. O expresie aritmetică de genul $5*8-3$ poate fi efectuată numai dacă se află în partea dreaptă a semnelui de egalitate. Partea stîngă a semnelui egal este pentru „nume variabilă”. Acesta este numele pe care îl dăm rezultatului expresiei aritmetice. Pare ciudat ceea ce v-am spus, dar, după cum veți vedea peste cîteva pagini, este foarte simplu.

Observație. Nu putem folosi "X" pentru operația de înmulțire. Din nefericire, folosirea lui "X", care este de fapt o literă, pentru înmulțire, s-a stabilit mult mai demult. Folosim litere pentru alte scopuri, iar utilizarea lui "*" este mult mai puțin generatoare de confuzii pe care calculatorul nu le tolerează.

Comanda SCR

Inițial, trebuie să eliminăm programul existent în memoria calculatorului. Cel mai drastic mod de distrugere este executarea comenzii **SCR**.



Tastați:

SCR [RETURN]

Ce efect are comanda **SCR**? **SCR** șterge toate programele existente în memorie și inițializează toate variabilele cu zero. Este puțin cam dur acest procedeu, dar este o cale bună pentru a porni din nou totul de la zero. Introducerea comenzii **SCR** mai rezolvă și unele "treburi casnice" cum ar fi plasarea lui **READY** și a cursorului în colțul din stînga sus.

Tastați:

LIST [RETURN]

pentru o confirmare din partea calculatorului a faptului că în memorie nu mai există nici un program.

Program BASIC pentru calculul ariei unui rezervor sferic

Vom folosi în continuare calculatorul pentru rezolvarea unei probleme simple: calculul ariei unui rezervor sferic. Vă mai reamintiți cum se calculează aria unei sfere, căci despre ea este vorba? Se înmulțesc: patru cu trei și paisprezece și cu pătratul razei. Cam dificilă această exprimare, nu vi se pare? Să alegem pentru arie și rază litere, astfel încît să putem rescrie textul ca pe o instrucțiune BASIC-aMIC:

20 A=4 * PI * R * R

unde A reprezintă aria rezervorului sferic, PI este 3,14 iar R=raza rezervorului. 20 este numărul liniei de program.

Observație. Atenție la utilizarea lui " * " pentru înmulțire. Pentru operația de ridicare la putere folosiți " ↑ "

Instrucțiunile:

20 A=4 * PI * R * R

și

20 A=4 * PI * R ↑ 2

sînt echivalente. Vom prefera în continuare ultima formă de scriere.

Ce semnificație are linia 20 pentru calculator? În momentul execuției acestei linii, calculatorul ia valoarea lui R (trebuie s-o cunoască!), o ridică la pătrat, înmulțește rezultatul cu 4 și apoi cu 3,14 și, în final, atribuie noul rezultat variabilei A.

Observație. Nu putem inversa $4 * PI * R \uparrow 2 = A$. Această formă de scriere nu are nici un sens pentru calculator.

Pentru identificarea valorilor cunoscute (raza), cît și pentru cele pe care dorim să le obținem (aria), putem folosi oricare din cele 26 de litere (A-Z) ale alfabetului. Este bine, însă, să utilizați litere cu o anumită semnificație, în context cu problema de rezolvat (exemplu: R – rază, A – arie) etc.

În algebra obișnuită, literele alfabetului sînt deseori folosite pentru termeni care pot lua valori numerice variabile. Expresia $L+2$, de exemplu, poate fi egală cu orice număr L, plus 2.

BASIC-ul are o astfel de structură încît permite utilizatorului să atribuie valori numerice oricărei litere din alfabet.

Dacă aria unui rezervor sferic se calculează cu formula $4\pi R^2$, ne întrebăm cît este aria unui rezervor cu raza de 3 m? Pentru ca aMIC-ul să ne răspundă la această întrebare, introduceți următoarele linii de program:

10 R=3 [RETURN]
20 A=4 * PI * R ↑ 2 [RETURN]

Verificați cu atenție programul, după care tastați:

```
RUN [RETURN]
```

Ce ați obținut? Doar **READY**? Într-adevăr, dvs. aveți dreptate. Dar, de ce numai atât? Nu ați primit nici un răspuns de la calculator deoarece ați uitat să-i spuneți să afișeze valoarea ariei! Ne pare rău! În limbajul BASIC tipărirea valorilor numerice se realizează tot cu instrucțiunea **PRINT**. Tastați:

```
30 PRINT "A" [RETURN]  
RUN [RETURN]
```

Ați obținut 113.097, cât reprezintă aria rezervorului de rază 3 m? Nu!? Dar ce s-a întâmplat? Cum de nu ne-am dat seama de la început că am greșit? Deoarece în linia 30, A apare între ghilimele, la execuția programului, după cum știm foarte bine, se va tipări doar A – caracterul cuprins în interiorul ghilimelelor – și nimic mai mult.

Dacă dorim ca aMIC-ul să tipărească numai litera A, atunci o includem între ghilimele; dacă dorim însă valoarea variabilei A, ca în cazul nostru, nu mai punem ghilimele. Atenție mărită, deci!

Tastați:

```
30 PRINT A [RETURN]  
RUN [RETURN]
```

și priviți cum pe ecranul televizorului a apărut valoarea mult așteptată: 113.097.

Observație. Valorile reale se scriu cu punct zecimal (nu cu virgulă).

Dacă doriți să afișați pe aceeași linie atât valoarea lui A cât și... câteva cuvinte, de exemplu: "A=", tastați:

```
30 PRINT "A=", A [RETURN]
```

Observație. O instrucțiune se poate înlocui, prin tipărirea noii instrucțiuni cu același număr de linie, cu vechea instrucțiune – lucru pe care l-am dorit și noi.

```
SCR  
10 R=3  
20 A= 4*PI*R^2  
RUN  
READY  
30 PRINT "A"  
RUN  
A  
READY  
30 PRINT A  
RUN  
113.097  
READY  
30 PRINT "A=", A  
RUN  
A=      113.097  
READY  
■
```

Fig. 2.1. Programul conversației plus rezultate.

Introduceți din nou comanda **RUN** și spuneți-vă părerea în legătură cu rezultatul. Pare mult mai explicit, nu-i așa?

În fig. 2.1 se prezintă imaginea întregii conversații pe care dvs. ați purtat-o cu calculatorul aMIC.

Variabile numerice

În procesul descrierii programului BASIC pe care l-ați urmărit au fost introduse noi elemente BASIC, cum ar fi: variabile, instrucțiunea de atribuire și instrucțiunea **PRINT** de editare a valorilor variabilelor numerice.

Variabila numerică poate fi orice literă din alfabet de la A–Z. În general, PC-urile permit folosirea a două litere pentru denumirea variabilelor numerice, de exemplu AB, AC, AE și așa mai departe. Alte calculatoare personale folosesc pentru reprezentare, pe lângă litere și o cifră cuprinsă între 0 și 9: A1, A2, ș.a.m.d. Toate aceste posibilități pun în evidență nu-

Tabelul 2.1

A	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9
B	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9
C	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
D	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9
E	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9
F	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9
G	G0	G1	G2	G3	G4	G5	G6	G7	G8	G9
H	H0	H1	H2	H3	H4	H5	H6	H7	H8	H9
I	I0	I1	I2	I3	I4	I5	I6	I7	I8	I9
J	J0	J1	J2	J3	J4	J5	J6	J7	J8	J9
K	K0	K1	K2	K3	K4	K5	K6	K7	K8	K9
L	L0	L1	L2	L3	L4	L5	L6	L7	L8	L9
M	M0	M1	M2	M3	M4	M5	M6	M7	M8	M9
N	N0	N1	N2	N3	N4	N5	N6	N7	N8	N9
O	O0	O1	O2	O3	O4	O5	O6	O7	O8	O9
P	P0	P1	P2	P3	P4	P5	P6	P7	P8	P9
Q	Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
R	R0	R1	R2	R3	R4	R5	R6	R7	R8	R9
S	S0	S1	S2	S3	S4	S5	S6	S7	S8	S9
T	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
U	U0	U1	U2	U3	U4	U5	U6	U7	U8	U9
V	V0	V1	V2	V3	V4	V5	V6	V7	V8	V9
W	W0	W1	W2	W3	W4	W5	W6	W7	W8	W9
X	X0	X1	X2	X3	X4	X5	X6	X7	X8	X9
Y	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7	Y8	Y9
Z	Z0	Z1	Z2	Z3	Z4	Z5	Z6	Z7	Z8	Z9

mărul mare de alternative pentru reprezentarea variabilelor numerice. Folosirea simplă a literelor alfabetului de la A la Z ne furnizează 26 de variabile, combinațiile de la AA–ZZ oferă încă $26 \times 26 = 626$ variabile, iar folosirea combinațiilor de tipul AØ la Z9 încă 260.

În orice caz, aceasta nu înseamnă că sînteți liberi să folosiți 912 variabile numerice diferite în cadrul aceleiași program. Calculatorul dvs. deține o unitate de memorie ale cărei dimensiuni sînt adesea nesatisfăcătoare pentru înmagazinarea a 912 variabile diferite. Aveți libertatea de a alege dintre aceste 912 variabile, dar numărul de variabile pe care îl puteți folosi o dată este limitat.

Cît privește variabilele numerice utilizate în BASIC-aMIC, ele pot fi **reprezentate fie printr-o literă, fie o literă și o cifră**. Toate variabilele numerice utilizate în BASIC-aMIC sînt reale. Cele 286 de variabile numerice (simple) pe care le puteți utiliza în limbajul BASIC pe calculatorul personal aMIC sînt prezentate în tabelul 2.1.

Instrucțiunea de atribuire LET

Pentru a ilustra conceptul de variabilă cît și funcțiunile instrucțiunii de atribuire, imaginați-vă că, în cadrul programului anterior, există 26 de sertare. Fiecare sertar poate conține în orice moment cite un număr. Avem deja depozitate numere în citeva dintre sertare. De exemplu (vezi figura) 8 este în sertarul F, 2 este în sertarul B. Ce număr este în sertarul V? Dar în C? 20 este în sertarul . . . iar 12.5 este în sertarul . . .

A		J	9	T	20
B	2	K	-137	U	0
C	6	L	-3.14	V	50
D	7.9	M	100	W	8.25
E	-3	N	10	X	13
F	8	O	60	Y	0
G	80	P	20	Z	-15
H	70	Q	30		
I	12.5	R			
		S	50		

Folosiți creionul pentru a pune 8 în sertarul A. Cu alte cuvinte, scrieți cifra 8 în sertarul liber A. Puneți 14 în R. Puneți 3 în R. Dar, stați! Un sertar poate să conțină la un moment dat un singur număr. Înainte de a-l intro-

duce pe 3, trebuie să-l scoateți (ștergeți) din sertar pe 14, pe care l-ați introdus mai înainte.

A

8

R

3

Cind calculatorul "pune" un număr într-un sertar, el șterge automat ceea ce a avut anterior în sertar, așa cum ați procedat, de altfel, și dvs. Pentru a pune "3" în sertarul R veți șterge mai întâi ceea ce a fost anterior în sertar – "14".

Observație. Pătrunzind în interiorul calculatorului personal veți descoperi una din minunile tehnicii moderne – microprocesorul. Veți remarca, de asemenea, sertarele din structura "dulapurilor" ce poartă inscripția "RAM", "EPROM" etc.

Numim A, B, C, . . . , Z variabile. Numerele din A, B, C, . . . , Z reprezintă valorile lui A, B, C, . . . , Z.

Programul nostru folosește în linia 10 o instrucțiune de tip **LET** pentru a instrui calculatorul să "pună un număr într-un sertar" sau, altfel spus, să atribuie variabilei R valoarea 3.

În mod analog procedează și pentru instrucțiunea din linia 20, cu singura deosebire că întâi calculează valoarea expresiei din dreapta semnelui egal și după aceea o atribuie variabilei A.

Sintaxa instrucțiunii. În BASIC-ul Darmouth original, valorile variabilelor numerice sînt stabilite prin introducerea instrucțiunii **LET** a cărei sintaxă este:

Format general
LET <variabilă numerică> = <expresie>

Cuvîntul cheie **LET** semnaleză calculatorului că a fost stabilită o relație între o variabilă și un număr. În alte versiuni ale BASIC-ului, inclusiv cele utilizate pentru calculatorul personal aMIC, folosirea lui **LET** este facultativă. Expriarea **LET** va fi omisă din majoritatea exemplurilor BASIC-aMIC din această carte, deși ea va mai fi folosită uneori pentru a vă reaminti că sistemele mai vechi o folosesc.

În concluzie, ori de cîte ori introducem în memoria calculatorului o linie de forma: variabilă numerică, semnul egal (=) și apoi o valoare numerică, apăsînd ulterior pe **[RETURN]** spunem de fapt calculatorului să atribuie acea valoare numerică particulară acelei variabile numerice particulare. Altfel spus, în momentul execuției instrucțiunii **LET** se evaluează expresia din dreapta semnelui "=" după care, valoarea obținută se atribuie variabilei specificate în partea stîngă a semnelui "=".

TEST

În programul anterior, variabila din linia 10 este iar valoarea destinată variabilei A din linia 20 este

R
113.097

Particularizînd formatul instrucțiunii de atribuire **LET** pentru linia 20 constatați că <expresie> este "4 * PI * R ↑ 2" iar <variabilă numerică> este A,

ce reprezintă o zonă de memorie în care se plasează valoarea obținută (113.097) în urma evaluării expresiei de calcul a ariei rezervorului sferic.

Remarcă. O variabilă definită într-o expresie fără a fi fost definită în prealabil, primește automat valoarea zero.

Evaluarea expresiei are loc în conformitate cu următoarele **reguli**:

- se evaluează expresiile cuprinse între paranteze;
- în absența parantezelor, ordinea este: operatori unari (exemplu: +A; -A), ridicarea la putere, înmulțire și împărțire, adunare și scădere;
- dacă operatorii sînt de aceeași prioritate, evaluarea se face întotdeauna de la stînga la dreapta.

Constante numerice. Numerele 4; 3.14 și 2 (exponentul) reprezintă constante numerice BASIC (v. linia 20). Valoarea acestora este fixă în cadrul unui program – ea nu se modifică în timpul execuției programului. Limbajul BASIC recunoaște trei formate de reprezentare a constantelor numerice: format întreg (exemplu: 4; 2), format real (exemplu: 3.14) și format exponențial (vezi Conversația 5).

Observație. Constanta intrinsecă **PI** poate fi utilizată oriunde sînt permise expresii numerice și are loc atribuirea valorii 3.14 în locul specificat în expresie.

Aplicație. Scrieți un program BASIC pentru calculul expresiei (1). Se vor prezenta mai multe versiuni.

$$\frac{5/8+2/7}{3/4-1/6} \quad (1)$$

a) 10 N=5/8+2/7
20 P=3/4-1/6
30 PRINT N/P
40 END

b) 10 F=(5/8+2/7)/(3/4-1/6)
20 PRINT F
30 END

c) 10 PRINT (5/8+2/7)/(3/4-1/6)
20 END

Tipărirea valorilor unei variabile numerice

PRINT este, fără îndoială, cea mai utilizată instrucțiune BASIC. Ea permite calculatorului să vă vorbească după cum ... o programați! Formatul general al instrucțiunii este:

Format general

PRINT [<listă>]

unde <listă> reprezintă o succesiune de elemente admise de limbaj, separate între ele prin caracterele „,” (virgulă) sau „;” (punct și virgulă).

Particularizînd formatul instrucțiunii de editare **PRINT** la linia 30

30 PRINT A

(listă) se reduce la variabila A a cărei valoare calculată în linia 20 se afișează în momentul execuției programului. În acest caz, utilizarea separatorilor (" , " sau " ; ") nu a fost necesară.

TEST

a) Completați programul următor atribuind variabilei Y1 valoarea 10 și afișați valoarea variabilei Y1.

```
10 .....
20 .....
99 END

-----

20 PRINT Y1
10 Y1=10
99 END
```

b) Precizați funcțiunile următorului program BASIC-aMIC:

```
5 INIT                                40 PRINT B
10 A=5                                50 C=A+B
20 PRINT A                             60 PRINT C
30 B=7                                70 PRINT
-----                               99 END
-----
```

(5) șterge ecranul; (10) atribuie variabilei A valoarea 5; (20) tipărește valoarea variabilei A; (30) atribuie variabilei B valoarea 7; (40) tipărește valoarea variabilei B; (50) însumează valoarea variabilei A cu valoarea variabilei B și atribuie rezultatul variabilei C; (60) tipărește valoarea variabilei C; (70) tipărește o linie albă; (99) terminare program.

c) Precizați rezultatul următorului program BASIC. Verificați-vă răspunsurile cu un calculator aMIC.

```
c1) 10 LET X=1
     20 LET X=3
     30 PRINT X
     40 END
     RUN
```

```
c2) 10 A=8
     20 B=A
     30 PRINT B
     40 END
     RUN
```

```
-----
      3
c3) 10 A=25.3
     20 PRINT A
     30 A=8
     40 PRINT A
     99 END
     RUN
```

```
-----
      8
c4) 10 A=12
     20 B= 5
     30 PRINT C
     40 END
     RUN
```

```
-----
25.3
 8
```

```
-----
0
```

Separatorul " , " (virgula)

Pentru a merge mai departe cu explicațiile privind utilizarea instrucțiunii **PRINT**, vă propunem acum analiza instrucțiunii introduse ulterior în linia 40:

```
40 PRINT "A=", A
```

Referindu-ne la sintaxa instrucțiunii **PRINT**, remarcați cum, în acest caz, (listă) este reprezentată de două elemente admise de BASIC și anume: șir de caractere sau **constantă șir** ("A=") și variabilă (A). Cele două elemente sînt separate prin caracterul " , " (virgulă).

Vă reamintiți cum la execuția programului, calculatorul a tipărit mai întâi A și într-o poziție depărtată (vom vedea îndată cât de depărtată) a afișat 113.097.

Bănuți cine a despărțit cele două elemente de pe linia de editare? Ați intuit desigur că este vorba de virgulă! Atenție, deci, la virgulă!

Cînd separatorul folosit între elementele unei liste este **virgula**, editarea rezultatelor se face în două zone de 15 caractere fiecare: 1–15 și 16–30. Dacă valoarea de tipărit depășește lungimea unei zone, elementul următor este tipărit la începutul primei zone libere.

Observație. Pentru a afla cîte poziții are display-ul cu care lucrați, examinați următorul program BASIC și numărați apoi X-urile din lungul unei linii terminal.

```

5 PRINT "          1          2          3          4"
10 PRINT "1234567890123456789012345678901234567890"
20 PRINT "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX"
99 END

```

În fig. 2.2. puteți urmări modul în care calculatorul aMIC a executat instrucțiunea **PRINT** din linia 40.

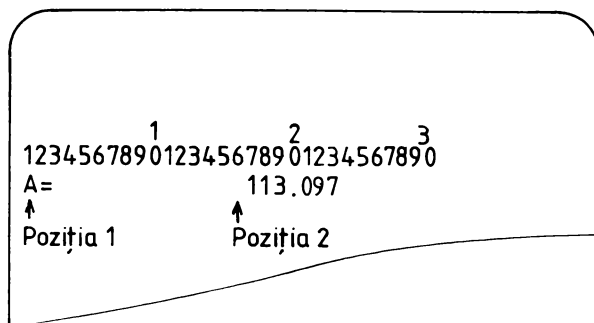


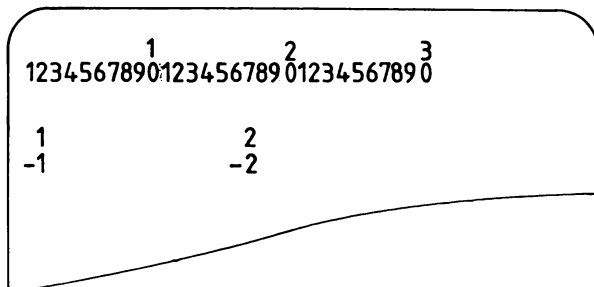
Fig. 2.2. Ilustrarea modului de utilizare a separatorului „,” într-o instrucțiune **PRINT** (valori pozitive).

Remarcați că valoarea lui A nu s-a afișat din coloana 16 așa cum am afirmat mai înainte, ci din coloana 17. Explicația este următoarea: cînd calculatorul afișează un număr pozitiv sau zero este lăsat mai întâi un spațiu liber, după care este afișat numărul. Priviți ce se întîmplă atunci cînd numerele negative sînt afișate alături de cele pozitive.

```

5 INIT          20 PRINT -1, -2
10 PRINT 1, 2   30 END
                RUN

```



Numerele negative sînt afișate cu semnul minus, fiind urmate de valoarea respectivă, în timp ce numerele pozitive sînt precedate de un spațiu liber.

TEST

Care este rezultatul execuției următoarelor programe BASIC-aMIC?

a) SCR

```
10 PRINT 1, 2, 3, 4
```

```
20 END
```

```
RUN
```

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

b) SCR

```
10 PRINT 1, 2, 3, 4, 5, -6, 7
```

```
20 END
```

```
-----
1           2
3           4
5           -6
7
```

Separatorul " ; " (punct și virgulă)

În cazul în care doriți ca rezultatele pe care le afișează calculatorul dvs. să fie mai apropiate folosiți în instrucțiunea **PRINT** în loc de virgulă, punct și virgulă. Utilizarea ca separator a lui " ; " conduce la tipărirea fără salt la următoarea zonă.

Revenind la programul nostru, înlocuiți în instrucțiunea 40 virgula cu punct și virgulă.

```
50 PRINT "A="; A [RETURN]
```

Tastați:

```
[RUN] [RETURN]
```

și remarcați cum imaginea este cu totul alta; 113.097 (valoarea lui A) nu mai este afișat începînd din coloana 16 (17, de fapt, datorită semnelui) ci din coloana 4!

Într-o instrucțiune **PRINT** în care folosim ca separator punct și virgulă numerele pozitive se editează în secvența: spațiu liber, număr, spațiu liber iar numerele negative după cum urmează: semnul minus, număr, blank.

Observație. Dacă doriți spații libere, le puteți include în șirurile de caractere respective.

Astfel, la execuția instrucțiunii:

```
60 PRINT " A = "; A
```

după semnul egal, calculatorul va lăsa automat două spații libere (blancuri).

Cum răspundeți la următoarea întrebare? Ce se întîmplă în cazul cînd (listă) din instrucțiunea **PRINT** se termină cu " , " sau " ; "? Iată și răspunsul. Existența unui separator " , " sau " ; " după ultimul caracter al listei suprimă operațiile **RETURN DE CAR** și **SALT LA LINIE NOUĂ** și astfel instrucțiunea **PRINT** următoare va edita datele pe aceeași linie, în continuare.

Dacă ați înțeles răspunsul la întrebarea noastră, încercați să scrieți instrucțiunea din linia 60 și în alt mod. Ați reușit? Dacă nu, reflectați asupra următoarei versiuni:

```
60 PRINT " A = ",
65 PRINT A
```

sau (cu punct și virgulă):

```
60 PRINT " A = ";
65 PRINT A
```

Restrângerea unui program

Indemînarea în programarea calculatoarelor nu constă atît în cunoașterea unui număr foarte mare de instrucțiuni (comenzi) diferite, cît în găsirea unor căi de a scrie programe eficiente. Programele eficiente sînt acele programe care își ating scopul propus prin folosirea unui număr minim de linii de program.

Priviți la lista programului. Acest program folosește 6 linii de instrucțiuni. Deși numărul acestora nu este impresionant de mare, vă punem următoarea întrebare: este posibilă restrîngerea acestui program? Încercați această secvență de instrucțiuni BASIC:

```
20 [RETURN]
30 PRINT "A = "; 4 * PI * R ↑ 2 [RETURN]
```

Considerînd că acest program de șase linii se află în memoria calculatorului dvs. modificați programul prin tipărirea numărului liniei 20 și apăsarea tastei [RETURN]. Dacă vă amintiți, din conversația precedentă, acest tip de operație șterge din memoria calculatorului linia respectivă. După aceea, veți rescrie linia 30, înlocuind variabila numerică A cu operația $4 * PI * R \uparrow 2$. Variabila numerică A este acum complet eliminată din prima parte a programului (linia 20).

Tastați:

RUN [RETURN]

Se pare că aMIC-ul a afișat aceeași valoare ca și în versiunea precedentă.

Observație. Instrucțiunea **PRINT** admite ca element al listei și expresie numerică.

Dar, această versiune de program folosește doar cinci linii de program și conține o variabilă mai puțin. Aceasta nu reprezintă o reducere remarcabilă a complexității programului, dar exemplul ilustrează că este într-adevăr posibil să rezolvi o problemă în cel puțin două moduri diferite.

Majoritatea problemelor pot fi rezolvate în nenumărate moduri. Dar întotdeauna există doar cîteva moduri mai eficiente pentru rezolvarea acestora; și aici intervin pricepera și experiența dvs.

TEST

Precizați care sînt rezultatele execuției următoarelor programe BASIC-aMIC?

a) 10 A=2
20 B=3
30 PRINT A+B
40 END
RUN

b) 10 A=2
20 B=3
30 C=A+B
40 PRINT C
50 END
RUN

c) 10 C=A+B
20 A=2
30 B=3
40 PRINT C
50 END
RUN

.....
5

.....
5

.....
0

SUMAR

Instrucțiunea PRINT	Exemplu
Instrucțiune	
PRINT (șir de caractere)	10 PRINT "A+B" 5 A=7
PRINT (variabilă numerică)	10 PRINT A
PRINT (expresie numerică)	10 PRINT 2 * 3
PRINT (listă)	10 PRINT A; "B="; 4*3.2.

În procesul exemplificării acestor instrucțiuni **PRINT**, a fost necesar să introducem operațiile de atribuire de tip **LET**.

TEST

1. Care dintre următoarele instrucțiuni **PRINT** trebuie precedate de o operație de atribuire?

- | | |
|--|------------------------|
| a) PRINT "DIMINEȚI CU FERESTRE
DESCHISE" | e) PRINT Y3 |
| b) PRINT "5+3" | f) PRINT A+Z4 |
| c) PRINT "3·14 TZURCA" | g) PRINT "A+Z4" |
| d) PRINT 5 * 2-3 | h) PRINT "A\$" |

2. Care este răspunsul calculatorului pentru fiecare din următoarele programe?

- | | |
|---|--|
| a) 10 INIT
20 PRINT 5 * 8
30 END | b) 10 INIT
20 PRINT 5 * M
30 PRINT |
| c) 10 INIT
20 X=2
30 Y=3
40 PRINT "R="; X * Y
50 END | d) 10 INIT
20 A=2
30 B=5
40 PRINT A; " * "; B
50 PRINT "R="; A * B
60 END |

**Particularități ale programării
în limbajul BASIC-PRAE**

vol. 2, pag. 199

Priviți, vă rugăm, programul. Examinați-l linie cu linie

10 R=3	60 PRINT 4; " * "; PI; " * "; R;
20 A=4 * PI * R ↑ 2	" ↑ "; 2; "="; A
30 PRINT A	70 PRINT 4; " * "; PI; " * "; R;
40 PRINT "A=", A	" ↑ "; 2; "="; 4 * PI * R ↑ 2
50 PRINT "A="; A	80 END

după modelul:

- 10 Atribuie variabilei R valoarea 3.
- 20 Atribuie variabilei A valoarea expresiei numerice $4\pi R^2$.
- 30 Tipărește valoarea calculată în linia precedentă.
- 40 Virgula decalează imprimarea valorii variabilei A la începutul zonei următoare. Linia terminal se consideră împărțită în zone a 14

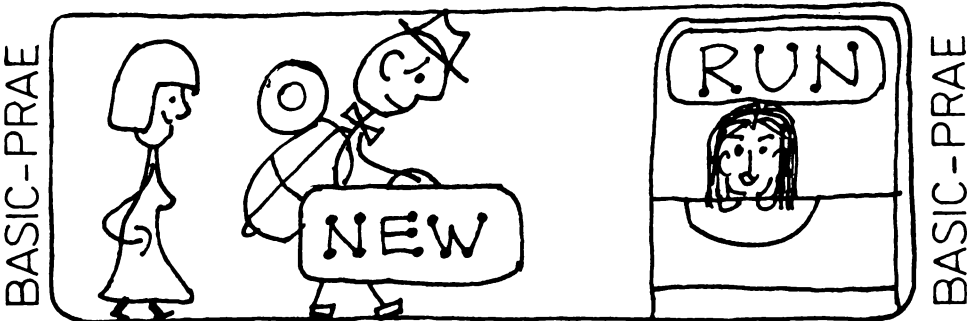
caractere (5 cîmpuri pentru un terminal cu 80 caractere, respectiv 8 cîmpuri pentru linia cu 132 caractere).

50 Punctul și virgula determină imprimarea valorii lui A imediat după semnul egal.

60, 70 Editează valoarea ariei rezervorului (A) calculată în două moduri diferite.

Remarci

- Numele variabilelor simple sînt formate dintr-o literă urmată de oricîte caractere alfanumerice (numai primele două caractere sînt luate în considerare).
- Numele variabilelor încep obligatoriu cu o literă și nu trebuie să coincidă cu cuvintele rezervate.
- Variabilele pot fi numerice și variabile șir.
- În momentul lansării în execuție a unui program, toate variabilele acestuia sînt inițializate cu zero.
- Într-un program BASIC-PRAE este posibilă definirea prin același nume a unei variabile simple și a unei variabile indexate.
- Sînt permise două tipuri de constante: numerice și șiruri.
- Toate constantele BASIC-PRAE sînt considerate în format flotant.
- Constantele se pot reprezenta într-unul din formatele: întreg, real, exponențial.
- Operațiile aritmetice admise sînt: adunarea (+), scăderea (-), înmulțirea (*), împărțirea (/), ridicarea la putere (\uparrow).
- Dacă într-o expresie aritmetică se folosesc variabile cărora nu li s-a atribuit anterior o valoare, expresia rămîne nedefinită.
- Prioritatea operațiilor este cea cunoscută.
- BASIC-PRAE acceptă mai multe instrucțiuni separate între ele prin caracterul " : " pe o linie program linie multiinstrucțiune (exemplu: 30 R=3 : A=4 * PI * R2 : PRINT A).
- Instrucțiunea de atribuire LET (cuvîntul LET este opțional) poate fi plasată oriunde în cadrul unei linii multiinstrucțiune. (Exemplu: 10 LET A=3 : B=5 : VIN=18).



În continuare, tastați **NEW**, [CR] în vederea introducerii programului analizat în memoria calculatorului personal PRAE. După ce v-ați asigurat că toate liniile au fost introduse corect, executați programul cu comanda **RUN**, [CR]. Analizați rezultatele.

Aplicație. Fiind date coordonatele punctelor M (3, 4) și N (-3.4, 5.75), să se scrie un program care calculează și afișează valoarea distanței MN.

```

5 CLS
10 X1=3 : Y1=4
20 X2=-3.4 : Y2=5.75
30 MN=((X2-X1)↑2+(Y2-Y1)↑2)↑0.5
40 PRINT "DISTANTA MN=", MN
50 END

```


TESTE

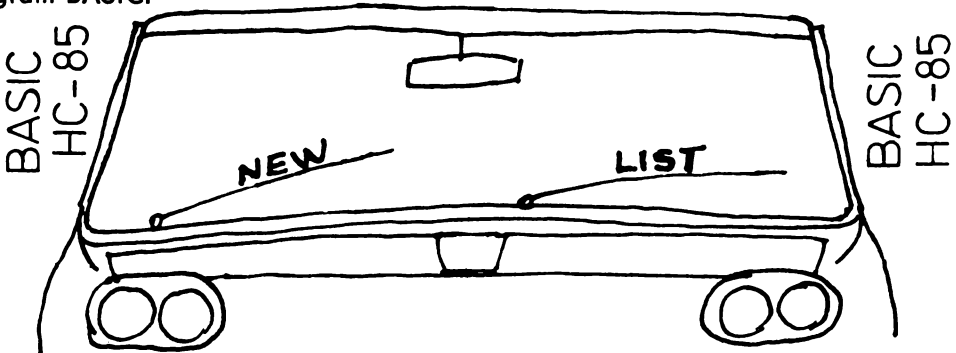
- Se consideră dreptunghiul cu virfurile în punctele $M(1,5)$; $N\left(\frac{2}{5}, \frac{5}{19}\right)$; $P(2,3)$; $Q\left(\frac{13}{5}, \frac{21}{5}\right)$.
Să se scrie un program BASIC care calculează și afișează lungimile laturilor și ale diagonalelor.
- Să se scrie un program BASIC care calculează și afișează valorile expresiilor:
 - a) $E = \frac{2,83(3,37)}{7,82} + 3,29$
 - b) $X = \sqrt{183,5}$
 - c) $A = \sqrt{p(p-2)(p-3)(p-7)}$
unde, $p=6$.
- Să se calculeze latura cubului (I) înscris într-un con circular drept cu raza bazei $R=3$ m și înălțimea $h=15$ m. Se va utiliza relația: $l = \frac{2Rh}{2R+h\sqrt{2}}$.
- Să se calculeze muchia unui tetraedru regulat înscris în sfera de rază $R=3$ m. Se va utiliza relația: $X = \frac{2R\sqrt{6}}{3}$.
- Să se calculeze latura (I) unui cub înscris în sfera de rază $R=3$ m. Se va utiliza relația: $l = \frac{2R\sqrt{3}}{3}$.

□ **Particularități ale programării în limbajul
BASIC HC-85, TIM S, SPECTRUM**

vol. 2, pag. 199

Variabile

Înainte de a studia utilizarea instrucțiunilor **LET** și **PRINT** în limbajul BASIC implementat pe calculatoarele HC-85, TIM S, SPECTRUM, ștergeți cu comanda **NEW** orice program sau variabilă anterioară și porniți totul de la zero. Așadar, tastați **NEW** (tasta A), **[ENTER]**, **[LIST]**, **[ENTER]** și vă veți convinge că în memoria calculatorului dvs. nu mai există nici un program BASIC.



Introduceți acum programul și apoi executați-l. Nu uitați că după introducerea fiecărei linii trebuie să tastați **[ENTER]**.

```

10 LET R=3
20 LET A=4 * PI * R ↑ 2
30 PRINT A
40 PRINT "A=", A
50 PRINT "A="; A

60 PRINT "A" ' A
70 PRINT "4 * 3.14 *"; R; " ↑ ";
  2; "="; A
80 PRINT 4; " * "; PI; " * "; R;
  " ↑ "; 2; "="; 4 * PI * R ↑ 2
9999 STOP

```

Instrucțiunile care prezintă interes pentru noi sînt cele din liniile 10, 20, 40, 50, 60. Restul instrucțiunilor (liniile 30, 70 și 80) sînt cunoștințe mai vechi.

Instrucțiunea **LET** din linia 10 atribuie variabilei R valoarea 3. Cuvîntul **LET** este obligatoriu (tasta L), deci nu poate fi omis din structura nici unei instrucțiuni de atribuire.

Cît privește variabilele BASIC HC-85, TIM S, SPECTRUM **regulile** sînt următoarele:

- numele variabilelor numerice simple pot fi de lungime arbitrară pornind de la o literă (obligatoriu, primul caracter) și continuînd cu litere, cifre, spațiu;
- numele variabilelor se referă la înțelesul literelor și nu la modul lor de scriere (litere mari, litere mici sau combinat).

Instrucțiunea LET

În linia 20, instrucțiunea **LET** atribuie valoarea expresiei ce calculează aria rezervorului sferic variabilei A. Cu această ocazie spunem că am definit variabila A (în linia 10 am definit variabila R).

Rețin atenția operatorii aritmetici pe care i-am utilizat în expresia numerică: "*" (pentru înmulțire) și "↑" (pentru ridicare la putere).

La întîlnirea constantei intrinseci **PI** se va atribui valoarea 3.14... în locul specificat în expresie.

În BASIC HC-85, TIM-S, SPECTRUM sînt folosiți următorii operatori aritmetici: "↑" (ridicare la putere), "*" (înmulțire), "/" (împărțire), "+" (adunare), "-" (scădere), "-" (schimbarea semnului).

Observație. Pentru ridicarea la putere este necesar ca puterea să fie număr pozitiv.

Calculul expresiilor permite utilizarea parantezelor care vor direcționa și ordinea de execuție a operațiilor (se execută toate operațiile din interiorul parantezelor; se execută toate operațiile unare de schimbare a semnului și ridicările la putere; se execută toate operațiile de înmulțire și împărțire; se execută toate operațiile de adunare și scădere).

Separatorul apostrof

Instrucțiunea **PRINT** din liniile 30, 40 50 și 70 permite tipărirea rezultatului – aria rezervorului sferic în patru moduri diferite. În cele ce urmează vă propunem să examinăm fiecare linie în parte.

Instrucțiunea

```
30 PRINT A
```

tipărește valoarea variabilei A calculată anterior 113.09734.

Instrucțiunea **PRINT** din linia 40 folosește separatorul ", " (virgulă) care, în BASIC HC-85, TIM S, SPECTRUM determină ca tipărirea valorilor să se facă câte două pe linie, una la începutul liniei iar cealaltă la mijloc.

Separatorul " ; " (punct și virgulă) utilizat în cadrul instrucțiunii **PRINT** din linia 50 nu are nici un efect special, valorile fiind tipărite una după alta, pe aceeași linie.

Și acum, un element de noutate pentru BASIC HC-85, TIM S, SPECTRUM – utilizarea apostrofului (v. linia 60) ca separator al elementelor unei liste din cadrul instrucțiunii **PRINT**.

Caracterul apostrof face ca valorile să fie tipărite una sub alta.

TEST

Examinați următorul program, linie cu linie și comentați fiecare instrucțiune:

10 PRINT	70 PRINT 2, 3
20 PRINT	80 PRINT , 2
30 PRINT	90 PRINT , 2; 3
40 PRINT "DO"; 40	100 PRINT 2 ' A
50 PRINT , "DO"; 50	15 PRINT "12345678901234567890
60 PRINT 23	123"

Linie multiinstrucțiune. Programul poate fi scris și sub forma liniilor multiinstrucțiune. Concatenarea instrucțiunilor se face cu ajutorul simbolului " : ".

TEST

Răspundeți prin DA sau NU la următoarele întrebări:

- Numele variabilelor numerice simple sînt formate dintr-un număr limitat de caractere.
- Numele variabilelor încep obligatoriu cu o literă.
- Litera mare și litera mică omoloagă sînt interpretate la fel.
- DANI și dani desemnează aceeași variabilă.
- Variabila aG 1 este scrisă corect.
- Atribuirea de noi valori variabilei se face prin instrucțiunea **LET**.
- Cuvîntul **LET** în cadrul unei instrucțiuni de atribuire în BASIC HC-85, TIM S, SPECTRUM este opțional.
- Operatorul aritmetic pentru ridicarea la putere este * *.
- Expresia $E=Y \uparrow - 3$ este corectă din punct de vedere BASIC.
- La scrierea expresiei
 $4 * * PI * * R \uparrow 2 = 4 * PI * R \uparrow 2$
 interpretorul BASIC HC-85, TIM S, SPECTRUM nu semnalează eroare.
- Instrucțiunea de atribuire
 10 **LET** = 5
 este acceptată de interpretor.

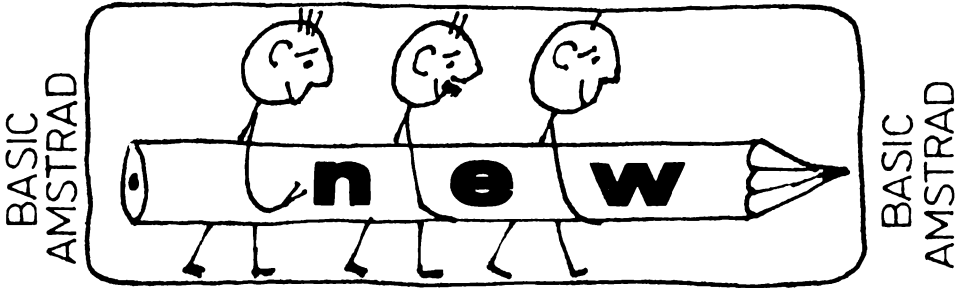
Particularități ale programării în limbajul BASIC-AMSTRAD

vol. 2, pag. 200

Variabile

Programul ilustrează modul de utilizare a instrucțiunilor de atribuire (liniile 10 și 20) și de editare în limbajul BASIC-AMSTRAD. Executați programul nu înainte de a șterge însă vechiul program din memorie cu comanda **NEW** și reveniți la manual pentru explicații suplimentare.

Linia 10 determină atribuirea unei valori – 3 în cazul nostru – variabilei reale r. O variabilă este identificată printr-un șir de caractere.



tere care începe cu o literă, urmată de un număr oarecare de litere sau cifre. Variabilele admise de limbajul BASIC-AMSTRAD sînt de următoarele tipuri: numerice reale, numerice întregi, șiruri de caractere.

Remarci

- Numele de variabilă nu trebuie să coincidă cu cuvintele rezervate BASIC-AMSTRAD.
- Fiecare nume de variabilă dintr-un program este asociat unei zone din memorie ce este utilizată pentru stocarea valorii curente a variabilei.
- Tipul unei variabile se poate stabili și prin caracterele speciale:
 - $\%$ – variabile numerice întregi
 - ! – variabile numerice reale
 - \$ – variabile șir de caractere
 ce se atașează numelui variabilei.
- Orice variabilă numerică la care ultimul caracter din nume nu este simbolul " $\%$ " este considerată variabilă pentru date în virgulă mobilă, adică variabila reală (variabilele R, A din program).
- Numele unei variabile numerice reale și numele unei variabile numerice întregi desemnează întotdeauna variabile distincte, chiar dacă numele este același, diferind, evident, numai prin simbolul " $\%$ " ce trebuie să fie prezent în cazul variabilei întregi (v. conversația 3).
- O variabilă poate fi declarată și implicit cu ajutorul instrucțiunilor: **DEFINT**, **DEFREAL**, **DEFSTR** (v. vol. 2, pag. 12).

TEST

Răspundeți prin DA sau NU la următoarele întrebări:

- Variabila R este de tip întreg.
- Variabila R $\%$ este de tip real.
- Variabilele A1 $\%$ și A1 au același nume dar sînt de tipuri diferite.
- Într-un program pot exista variabile simple numerice și variabile simple nenumerice (șir de caractere) cu același nume: L, L $\%$, L\$.
- Următoarele nume de variabile numerice sînt corecte: AGA . 4 $\%$, AGA - 4 $\%$, A - G - A - 4 $\%$.
- Următoarele nume de variabile numerice sînt incorecte: 38X $\%$, 3 - 8X $\%$, A38 A, LET, TEL, ELT.

Operații numerice

În BASIC-AMSTRAD datele numerice sînt prelucrate cu ajutorul operatorilor și funcțiilor (v. vol. 2, pag. 12) numerice. În linia 20 a programului

$$20 \ a = 4 * PI * r \uparrow 2$$

s-au utilizat: operatorul de atribuire "=" ce asignează valoarea expresiei numerice ($4\pi R^2$) variabilei a, operatorii aritmetici "*" și "\uparrow" pentru înmulțire, respectiv ridicare la putere și constanta intrinsecă PI.

Remarci

- O instrucțiune de atribuire memorează valoarea unei expresii plasate în partea dreaptă a semnului de egalitate ("=") în zonele de memorie corespunzând variabilei specificate în partea stângă a semnului de egalitate ("=").
- Structura generală a unei instrucțiuni de atribuire în BASIC-AMSTRAD cuprinde cuvântul opțional LET, urmat în mod obligatoriu de numele unei variabile, de semnul de egalitate ("=") și de o expresie numerică.
- În BASIC-AMSTRAD sint folosiți următorii operatori aritmetici: " \uparrow " (ridicarea la putere), " $*$ " (înmulțire), "/" (împărțire), "+" (adunare), "-" (scădere), "-" (schimbarea semnului).
- Constanta intrinsecă PI poate fi utilizată oriunde sint permise expresii numerice și are loc atribuirea valorii 3.14519265 în locul specificat în expresie.
- Ordinea de execuție a operațiilor într-o expresie numerică este: 1) se execută toate operațiile din interiorul parantezelor; 2) se execută toate operațiile unare de schimbare a semnului (-) și ridicările la putere (\uparrow) - de la stînga spre dreapta; 3) se execută toate operațiile de înmulțire și împărțire (de la stînga spre dreapta); 4) se execută toate operațiile de adunare și scădere (de la stînga spre dreapta).
- Orice operator aritmetic trebuie urmat de o expresie numerică.
- În cazul operațiilor de ridicare la putere a unor numere negative, nu este obligatorie folosirea parantezelor.
- Atunci cînd într-o expresie numerică nu există paranteze dar operatorii sint de aceeași prioritate în ierarhia execuției, evaluarea se face întotdeauna de la stînga la dreapta.
- Ordinea normală în care este evaluată o expresie poate fi schimbată prin folosirea parantezelor.

TEST

Răspundeți prin DA sau NU la următoarele întrebări:

- Expresia $X = B \uparrow - 2$ este scrisă corect.
- Expresia $X = - B \uparrow 2$ este scrisă corect.
- Expresia $X = - B \uparrow A$ este scrisă corect.
- Expresia $X = (- B) \uparrow 2$ este scrisă corect.
- Expresia $X = 3 + - 4$ este scrisă corect.
- Expresia $X = A (- B)$ este scrisă corect.
- Expresia $X = - B + A * - C$ este scrisă corect.
- În expresia $X = X * Y / Z$ se înmulțesc mai întii X cu Y iar produsul se împarte după aceea la Z.
- În expresia $E = X * (Y / Z)$ se înmulțește mai întii X cu Y iar produsul se împarte după aceea la Z.
- În expresia $E = (X + Y) * V$ se însumează mai întii Y cu X și apoi rezultatul se înmulțește cu V.
- În cadrul expresiei $E = X + (Y - ((A / B) \uparrow 2))$ se împarte mai întii A la B, apoi se face ridicarea la putere, urmată de diferență; în final, se adună X la valoarea dinainte.
- În cadrul expresiei:

$$E = 4 * PI * R \uparrow 2$$

se efectuează mai întii ridicarea la putere ($R \uparrow 2$), după aceea se înmulțește rezultatul cu PI, iar în final se înmulțește noul rezultat cu 4.

- În cadrul expresiei:

$$E = 4 * PI * R \uparrow 3 / 3$$

se efectuează mai întii ridicarea la putere ($R \uparrow 3$), după aceea se împarte rezultatul la 3, se înmulțește în continuare rezultatul obținut cu 4 iar, în final noul rezultat se amplifică cu PI.

Tabulare

După cum ați putut constata, o expresie numerică poate să conțină o singură variabilă sau constantă sau un grup de variabile sau constante și funcții, toate numerice, reunite prin operatori numerici. După ce expresiile sint evaluate, valorile lor se atribuie, prin instrucțiunea de atribuire

LET, unor variabile. După evaluare, ele sînt tipărite (la imprimantă) sau afișate (la terminal) cu ajutorul instrucțiunii **PRINT** (vezi liniile 25, 30, 35, 40, 45).

```

25 PRINT A
30 PRINT "A=", A
35 PRINT "A=: A
40 PRINT 4; "X"; PI; "X"; R; "↑"; "2"; "="; A
45 PRINT 4; "X"; PI; "X"; R; "↑"; 2; "="; 4 * PI * R ↑ 2

```

Formatul general al instrucțiunii **PRINT** este:

FORMAT GENERAL	
< număr de linie >	PRINT [# < număr canal > ,] [< listă >]

în care:

(număr canal) este numărul canalului necesar desemnării unei ferestre ecran, imprimantei, casetei sau dischetei pe care textul urmează să fie imprimat. Valoarea implicită pentru (număr canal) se ia zero (monitorul);

(listă) reprezintă o succesiune de elemente admise de limbaj (constante, variabile, expresii numerice etc.) separate între ele prin caracterele " , " sau " ; " .

```

113.097336
A =          113.097336
A = 113.097336
4 x 3.14159265 x 3^2 = 113.097336
4 x 3.14159265 x 3^2 = 113.097336

```

Fig. 2.3. Rezultatele execuției programului.

Ce se întimplă în urma execuției instrucțiunii **PRINT** din linia 25? După cum remarcați (v. fig. 2.3), pe monitor s-a afișat simplu 113.097336 ce reprezintă (cum numai noi știm) valoarea ariei rezervorului sferic.

Parcurgînd în continuare rezultatele obținute, remarcați pe rîndul imediat următor (v. fig. 2.3) aceeași valoare dar precedată și de numele variabilei. Acest rezultat se datorează instrucțiunii **PRINT** din linia 30 care utilizează separatorul " , " (v. formatul general).

În BASIC-AMSTRAD, virgula cere calculatorului să afișeze valoarea următoare peste 13 coloane. Totuși, dacă numărul de caractere ce trebuie afișat depășește cifra 12, valoarea următoare este decalată cu alte 13 coloane cu scopul de a păstra totdeauna un spațiu între valorile de afișat.

Observație. Mărimea zonei de 13 caractere poate fi modificată prin instrucțiunea **ZONE** (vezi Conversația 3).

Pe rîndul imediat următor (vezi fig. 2.3) remarcați, evident, același rezultat dar altfel editat. De această dată, în cadrul instrucțiunii **PRINT** (vezi

linia 35) am utilizat simbolul " ; " care comandă calculatorului să afișeze următoarea valoare imediat după precedenta (numai să nu fie prea mare pentru a se menține pe linie).

Următoarele valori (vezi fig. 2.3) reprezintă tot valoarea ariei rezervorului sferic, calculată însă în două moduri (vezi liniile 40 și 45) diferite.

Linie multiinstrucțiune (multiplă)

În BASIC-AMSTRAD orice linie de program poate conține un număr arbitrar de instrucțiuni (linie multiinstrucțiune sau multiplă) separate între ele cu ajutorul caracterului " : " (două puncte).

Încercați, în continuare, să modificați programul utilizând liniile multiinstrucțiune. Ștergeți programul precedent utilizând comanda **NEW** și introduceți următoarea secvență de instrucțiuni.

```
5 MODE 2
10 r=3 : a=4 * PI * r ↑ 2
20 PRINT a : PRINT 4 ; "x" ; PI ; "x" ; r ; " ↑ " ; 2 ;
   " = " ; a : PRINT 4 ; "x" ; PI ; "x" ; r ; " ↑ " ; 2 ;
   " = " ; 4 * PI * r ↑ 2
```

Executați programul și analizați rezultatele obținute. Ele trebuie să fie identice cu cele ale programului precedent.

□ Particularități ale programării în limbajul BASIC-COMMODORE

vol. 2, pag. 200

Să analizăm împreună programul linie cu linie.

10 R=3	50 ? "A=" ; A
20 A=4 * π * R ↑ 2	60 ? 4 "x" π "x" R " ↑ " 2 " = " A
30 ? A	70 ? 4 "x" π "x" R " ↑ " 2 " = "
40 ? "A=" , A	4 * π * R ↑ 2

10 Atribuie variabilei R valoarea 3.

20 Atribuie variabilei A valoarea expresiei numerice $4\pi R^2$.

30 Tipărește valoarea calculată în linia 20.

40 Virgula decalează imprimarea valorii variabilei A la începutul zonei următoare. Linia terminal se consideră împărțită în 4 zone a câte 10 caractere fiecare.

50 Punctul și virgula determină imprimarea valorii lui A imediat după semnul egal.

60, 70 Editează valoarea ariei rezervorului calculată în două moduri diferite.

Remarci

- Numele variabilelor simple sint formate dintr-o literă urmată de oricâte caractere alfanumerice (numai primele două caractere sint luate în considerare).
- Numele unei variabile nu trebuie să coincidă cu cuvintele rezervate.
- Variabilele pot fi: numerice reale, numerice întregi, șir.

- Tipul întreg al unei variabile se indică prin caracterul "0" adăugat numelui acesteia.
- Într-un program pot exista variabile simple numerice (tip întreg, real) și variabile simple nenumerice cu același nume.
- BASIC-COMMODORE acceptă constante numerice (întregi, reale) și constante șir.
- Constantele numerice pot fi reprezentate în format: întreg, zecimal, exponențial.
- Operațiile aritmetice admise sînt: adunarea (+), scăderea (-), înmulțirea (*), împărțirea (/), ridicarea la putere (^).
- Prioritatea operațiilor este cea cunoscută.
- Dacă într-o expresie aritmetică se folosesc variabile cărora nu li s-a atribuit anterior o valoare, expresia rămîne nedefinită.
- BASIC-COMMODORE acceptă liniile multiinstrucțiune, instrucțiunile fiind separate de caracterul ":".
- Într-o instrucțiune de atribuire cuvîntul **LET** este opțional.

COMMODORE

NEW RUN

COMMODORE

Și-acum, eliminați programul curent din memoria calculatorului cu comanda **NEW**, [RETURN], tastați programul și dacă nu aveți erori de sintaxă, executați programul cu comanda **RUN**, [RETURN]. Ați obținut pe monitorul dvs. 113.097336? Ați înțeles foarte bine semnificația celor doi separatori " , " și " ; " din cadrul unei instrucțiuni **PRINT**? Dacă da, vă propunem să realizați în continuare, singuri, fără ajutorul nostru, o simulare privind execuția acestui program. Vă sugerăm să folosiți o foaie de matematică pe care s-o împărțiți în 40 de coloane (0÷39). Atenție la cele patru zone ale liniei terminal! Cel care vă poate verifica cel mai bine este numai calculatorul dvs. personal COMMODORE. Succes!

Aplicație. Fiind date punctele A(3,4) și B(5,6) să se scrie un program care determină ecuația dreptei definită de cele două puncte.

```

5 CHR$ (147)
10 X1 = 3 : Y1 = 4
20 X2 = 5 : Y2 = 6
30 PANTA = (Y2 - Y1) / (X2 - X1)
40 TLIBER = Y1 - M * X1
50 PRINT " ( " ; X1 ; " , " ; Y1 ; " ) , ( " ; X2 ;
   " , " ; Y2 ; " ) "
60 PRINT "ECUAȚIA DREPTEI ESTE: Y = " ;
70 PRINT PANTA ; " * X + ( " ; TLIBER ;
   " ) "
80 END

```

TESTE

- Să se calculeze viteza și timpul de cădere ale unui corp care cade liber (fără viteză inițială) de la înălțimea de 30 m. Se vor utiliza relațiile de calcul: $v = \sqrt{2gh}$ și $t = \sqrt{\frac{2h}{g}}$.
- Să se calculeze timpul de urcare și înălțimea maximă la care se ridică un corp care este aruncat vertical cu viteza $v_0 = 20$ m/s. Se vor utiliza relațiile de calcul: $t = \frac{v_0}{g}$ și $h = \frac{v_0^2}{2g}$, unde $g = 9,81$ m/s².

- Două corpuri de mase $m_1=1$ kg și $m_2=2$ kg se mișcă unul spre celălalt cu vitezele $v_1=2$ m/s și $v_2=-3$ m/s. Să se calculeze energia cinetică pierdută prin ciocnirea lor (păstică). Se va utiliza relația: $Q = \frac{1}{2} \frac{m_1 \cdot m_2}{m_1 + m_2} (v_1 - v_2)^2$ J.
- Să se calculeze presiunea datorată apei de mare la o adâncime $h=80$ cm. Se va utiliza relația: $p = \rho gh$, unde $\rho = 1,026$ g/cm³.
- Să se calculeze viteza de scurgere a apei printr-un tub cu diametrul interior $d=4$ cm, știind că debitul de volum $Q_v=10$ litri/minut. Se va utiliza relația: $v = 40v/\pi d^2$ (m/s).

□ Particularități ale programării în limbajul BASIC-80

vol. 2, pag. 200

Analiza programului

Priviți programul. Vă invităm să-l lecturați cu atenție încercînd, pe cît posibil, o comparație cu programele precedente.

```
10 PI=3.14159
20 R=3
30 A=4 * PI * R ↑ 2
40 PRINT A
```

```
50 PRINT "A=", A
60 PRINT "A="; A
70 PRINT 4 " * " PI " * " R " ↑ " 2
   " = " A
80 PRINT 4 " * " PI " * " R " ↑ " 2
   " = " 4 * PI * R ↑ 2
```

Ați găsit vreo deosebire? Vom examina acest program linie cu linie.

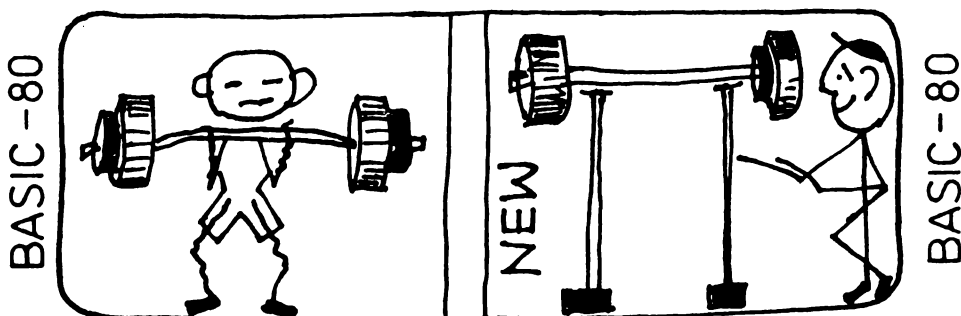
- 10 Atribuie valoarea 3.14159 variabilei PI ! În BASIC-80 lipsește constanta π și, în consecință, o vom transmite noi microcalculatorului. Numele variabilei conține două litere.
- 20 Inițializează pe R cu valoarea 3.
- 30 Atribuie valoarea expresiei numerice $4\pi R^2$ variabilei A. Desigur, se putea înlocui direct valoarea lui PI, dar noi am preferat prima variantă.
- 40 Tipărește valoarea ariei rezervorului sferic de rază 3.
- 50 Virgula decalează imprimarea valorii variabilei A (aria rezervorului sferic) la începutul zonei următoare. BASIC-80 împarte linia de afișare în zone de cîte 14 poziții fiecare.
- 60 Punctul și virgula dintre "A=" și A determină imprimarea valorii variabilei A imediat după semnul egal.
- 70 Tipărește în clar formula de calcul a ariei rezervorului sferic în care PI, R și A au valorile determinate de instrucțiunile din liniile 10, 20 respectiv 30. De notat absența separatorului " ; " din cadrul instrucțiunii.
- 80 Tipărește în clar formula de calcul a ariei rezervorului sferic unde PI și R au valorile determinate de instrucțiunile din liniile 10 și 20 iar A se calculează chiar în cadrul instrucțiunii. Nu este necesară utilizarea separatorului " ; ".

Remarci

- În BASIC-80 nu sînt restricții privind lungimea numelui unei variabile dar numai primele 40 de caractere sînt luate în considerare.
- Numele unei variabile nu trebuie să coincidă cu un nume rezervat (comenzi, instrucțiuni, funcții, operatori logici). El poate conține litere, cifre sau caracterul ".".

- Variabilele pot fi variabile întregi, simplă precizie sau dublă precizie și variabile șir.
- Tipul unei variabile poate fi stabilit și printr-un caracter special ce urmează numele variabilei: \$ – variabila de tip șir; % – variabilă întreagă; ! – variabilă simplă precizie; # – variabilă dublă precizie.
- Tipul unei variabile BASIC-80 poate fi declarat și explicit prin instrucțiunile DEFINT, DEFSG, DEFDBL.
- Tipul implicit pentru o variabilă este simplă precizie.
- Sînt permise două tipuri de constante: numerice și șiruri.
- BASIC-80 recunoaște cinci tipuri de constante numerice: întregi, virgulă fixă, virgulă mobilă (forma cu exponent), virgulă mobilă dublă precizie, hexazecimale și octale.
- Constantele pot fi reprezentate în simplă precizie (memorate cu 7 cifre și reprezentate cu max. 6 cifre) și dublă precizie (cu 16 cifre).
- Operațiile aritmetice admise sînt: adunarea (+), scăderea (-), înmulțirea (*), împărțirea (/), ridicarea la putere (^), împărțirea întreagă (\) și restul împărțirii (MOD).
- Dacă se atribuie unei variabile numerice o constantă al cărei tip diferă de cel al variabilei, atunci valoarea numerică se convertește automat la tipul variabilei.
- La evaluarea expresiilor, toți operanzii sînt convertiți automat la tipul operandului de precizie maximă care este și tipul rezultatului expresiei.
- La conversia din virgulă mobilă în întreg, se face rotunjirea la întregul cel mai apropiat și nu se trunchiază partea fracționară.
- La conversia din dublă precizie în simplă precizie se rețin numai primele 7 cifre zecimale, cu rotunjire la ultima cifră.
- Operatorii aritmetici (\) și MOD se pot folosi și cu operanzi neîntregi ce sînt automat convertiți la întregi înainte de operație.
- Prioritatea operațiilor este cea cunoscută.
- În cazul unei depășiri sau împărțiri cu zero se afișează un mesaj, iar execuția programului continuă.

Dacă ați înțeles programul puteți, desigur, să-l tastați și apoi să-l executați. Înainte de a-l tasta, dați însă comanda **NEW** pentru a elimina din memoria internă a calculatorului programul existent. De asemenea, dacă doriți, puteți introduce pe o linie program mai multe instrucțiuni separate prin simbolul " : ".



Aplicație. Introduceți și executați următoarele programe BASIC-80.

a) 10 PRINT "EXEMPLU DE ADU
NARE"
20 AANR = 8
30 ABNR = 5
40 ACNR = AANR + ABNR
50 PRINT : PRINT
60 PRINT AANR " + " ABNR
" = " ACNR
70 END

b) 10 PRINT "EXEMPLU DE SCĂDERE"
20 AANR = 75
30 ABNR = 100
40 ACNR = AANR - ABNR
50 PRINT : PRINT : PRINT
60 PRINT AANR " - " ABNR
" = " ACNR
70 END

```
c) 10 PRINT "EXEMPLU DE INMULȚIRE"
20 AANR = 12
30 ABNR = 13
40 ACNR = AANR * ABNR
50 PRINT : PRINT
60 PRINT AANR ; " * " ABNR
   " = " ACNR :
70 END
```

```
d) 10 PRINT "EXEMPLU DE ÎMPĂRȚIRE"
20 AANR = 120 / 20
30 ABNR = 195
40 ACNR = AANR / ABNR
50 PRINT : PRINT
60 PRINT AANR " / " ABNR " = "
   ACNR
70 END
```

```
e) 10 PRINT "EXEMPLU DE RIDICARE LA
   PUTERE"
20 AANR = 13
30 ABNR = 2 ↑ AANR
40 PRINT "2 LA PUTEREA" AANR " = "
   ABNR
50 END
```

```
f) 10 A = 95630
20 B = 8523.
30 C = A * B
40 PRINT A, " * ", B, " = ", C
50 PRINT A " * " B " = " C
60 PRINT A; *; B; =; C
70 END
```

□ Particularități ale programării în limbajul BASIC-PLUS

vol. 2, pag. 200

Priviți programul. Nu vă grăbiți în parcurgerea lui.

```
10 R=3
20 A=4 * PI * R ** 2
30 PRINT A
40 PRINT "A=", A
50 PRINT "A="; A
60 PRINT 4; " * "; PI; " * "; R; " ↑ "; 2; "="; A
70 PRINT 4; " * "; PI; " * "; R; " ↑ "; 2; "="; 4 * PI * R ↑ 2
```

Analizați fiecare instrucțiune în parte și comparați-o cu cea din programele precedente. Ați descoperit vreo deosebire? Dacă ați găsit-o înseamnă că ați făcut o analiză riguroasă, dacă nu, reluați lectura programului cu mai multă atenție, după cum urmează:

- 10 Atribuie variabilei R valoarea 3.
- 20 Atribuie valoarea expresiei numerice $4\pi R^2$ variabilei A. Operatorul aritmetic pentru ridicarea la putere este "**".
- 30 Tipărește valoarea calculată în linia precedentă.
- 40 Virgula decalează imprimarea valorii variabilei A la începutul zonei următoare. Linia terminal se consideră împărțită în zone care au o lungime de 14 caractere.
- 50 Separatorul punct și virgulă determină imprimarea valorii lui A la două spații după semnul egal (valoarea extrasă este numerică pozitivă).
- 60, 70 Editează valoarea ariei rezervorului (A), utilizându-se două moduri diferite de calcul.

Remarci

- Numele variabilelor simple sint formate dintr-o literă sau o literă urmată de o singură cifră.
- Variabilele pot fi numerice și variabile șir.

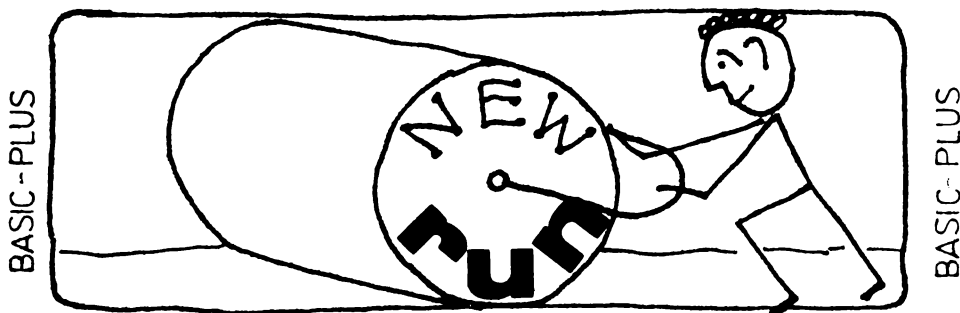
- Variabilele simple numerice pot fi întregi (referă valori numerice de tip întreg ce se reprezintă pe un cuvint) și reale (referă valori numerice de tip real ce se reprezintă pe două cuvinte).
- Tipul întreg al unei variabile se indică prin caracterul "0'0" adăugat numelui acesteia.
- Într-un program BASIC-PLUS pot exista două variabile cu același nume dar de tipuri diferite.
- Într-un program pot exista variabile simple numerice și variabile simple șir cu același nume.
- Sunt permise două tipuri de constante: numerice și șiruri.
- BASIC-PLUS recunoaște constantele numerice întregi și constantele reale.
- Constantele numerice pot fi reprezentate în format: întreg, zecimal, exponențial.
- Operațiile aritmetice admise sint: adunarea (+), scăderea (-), înmulțirea (*), împărțirea (/), ridicarea la putere (**).
- Dacă într-o expresie aritmetică se folosesc variabile cărora nu li s-au atribuit anterior o valoare, expresia rămâne nedefinită.
- Prioritatea operațiilor este cea cunoscută.
- Structura generală a unei instrucțiuni de atribuire cuprinde cuvintul opțional **LET** urmat, în mod obligatoriu, de numele uneia sau mai multor variabile (atribuire multiplă), de semnul de egalitate și de o expresie numerică.
- Dacă se atribuie unei variabile numerice o constantă de un tip diferit de al variabilei, atunci valoarea numerică se convertește automat la tipul variabilei (vezi linia 10).
- La evaluarea expresiilor, toți operandii sint convertiți automat la tipul operandului de precizie maximă care este și tipul rezultatului expresiei. Astfel, dacă un operand este de tip întreg iar celălalt este de tip real se face mai întâi conversia tipului întreg în tip real după care se execută operația cu valori de tip real.
- BASIC-PLUS acceptă linii multiinstrucțiune.

TEST

Precizați valoarea lui A în urma execuției următorului program:

```
10 PRINT
20 A = 4 * PI * R ** 2
30 PRINT A
```

R. Zero, deoarece variabilei R nu i s-a atribuit nici o valoare.



Dacă doriți să introduceți programul în calculatoarele INDEPENDENT, CORAL dați mai întâi comanda **NEW** după care apăsați pe **[RETURN]**. (v. fig. 2.4). Tastați **RUN**, **[RETURN]** și veți obține, în urma execuției programului, în diferite formate de editare 113.097 ce reprezintă aria rezervorului de rază 3 m.

```

10 R = 3
20 A = 4 * PI * R ** 2
30 PRINT A
1000 END
:RUN
113.097
END OF PROGRAM AT LINE 1000

```

a)

b)

```

10 R = 3
20 A = 4 * PI * R ** 2
50 PRINT A = "; A
1000 END

:RUN
A = 113.097
END OF PROGRAM AT LINE 1000

```

```

10 R = 3
20 A = 4 * PI * R ** 2
40 PRINT "A =", A
1000 END
:RUN
A = 113.097
END OF PROGRAM AT LINE 1000

```

c)

d)

```

10 R = 3
20 A = 4 * PI * R ** 2
60 PRINT 4; "X"; PI; "X"; R; " ^ 2; " = "; A
70 PRINT 4; " * "; PI; " * "; R; " ^ 2; " = "; 4 * PI * R ^ 2
1000 END
:RUN
4 * 3.14159 * 3 ^ 2 = 113.097
4 * 3.14159 * 3 ^ 2 = 113.097
END OF PROGRAM AT LINE 1000

```

Fig. 2.4. Rezultatele execuției (a, b, c, d) programului.

Aplicație. Într-un triunghi dreptunghic isoscel se cunoaște lungimea catetei, $A_1=A_2=8$ cm. Să se scrie un program BASIC pentru calculul ariei triunghiului (S) și înălțimii (H) duse din vârful unghiului drept.

```

10 PRINT
20 A1, A2 = 8
30 S = (A1 * A2) / 2
40 PRINT "ARIA =", S
50 B = (2 * A1 ** 2) ** 0,5
60 H = A1 * A1 / B
70 PRINT "AD =", H
80 END

```

TESTE

- Să se calculeze debitul de volum al apei ce curge printr-o conductă orizontală cu secțiunea transversală variabilă. Se cunosc: secțiunile transversale $S_1=10^{-3}$ m², $S_2=2 \cdot 10^{-3}$ m², și diferența de nivel dintre coloanele de lichid din sondele de presiune pe conductă $\Delta h=0,2$ m. Se va utiliza relația de calcul:

$$Q_v = S_1 S_2 \sqrt{2g \Delta h (S_2^2 - S_1^2)} \quad (\text{m}^3/\text{s}).$$

- Un corp cu masa $m=2$ kg este aruncat orizontal cu viteza inițială $v_0=20$ m/s. Să se calculeze energia cinetică a corpului după un interval de timp $\Delta t=4$ s. Se va utiliza relația de calcul $E_0 = \frac{mv^2}{2} = \frac{m}{2} \cdot (v_0^2 + g^2 \Delta t^2)$, unde $g=9,81$ m/s².
- Să se calculeze rezultanta a două forțe de mărime egală $F=\sqrt{2}$ N avind direcții perpendiculare.
- Un corp cu masa $m=8$ kg se deplasează fără frecare pe un plan orizontal cu accelerația $a=0,5$ m/s². Să se calculeze: a) mărimea forței orizontale care deplasează corpul; b) viteza corpului la momentul $t=30$ s; c) distanța parcursă de corp în timpul $t=30$ s; d) energia cinetică a corpului în momentul $t=20$ s. Se vor utiliza formulele: $F=ma$ (N); $v=at$ (m/s); $x=at^2/2$ (m); $E_0 = \frac{ma^2 t^2}{2}$ (J).
- Să se calculeze volumul corpului obținut prin rotația unei suprafețe hexagonale regulate în jurul unei laturi, știind că latura hexagonului (a) este egală cu 8 cm. Se va utiliza relația: $V = \frac{9\pi a^3}{2}$.

□ Particularități ale programării în limbajul ABASIC

vol. 2, pag. 201

Acum, după ce aveți experiența limbajului BASIC implementat pe calculatoare personale, micro și mini calculatoare, ce puteți spune despre acest program? Identificați pentru început cele două variabile A și R. Ele sînt variabile numerice reale. În ABASIC variabilele numerice pot fi: numerice reale și numerice întregi. O variabilă este identificată printr-un șir de caractere ce începe cu o literă urmată de un număr oarecare de litere sau cifre. În vederea precizării explicite a tipului, ea se termină, opțional, cu unul din caracterele "!" ; "#" pentru variabile numerice reale și cu "0/" pentru variabile numerice întregi.

Variabilele care nu au precizat un tip implicit sînt împărțite în clase de tipuri, funcție de primul caracter al numelui (prin instrucțiuni ca: **DEFINT**, **DEFDBL**). Dacă precizarea tipurilor implicite nu este făcută de utilizator, atunci toate variabilele sînt considerate numerice reale.

ABASIC permite ca **operații aritmetice**: adunarea (+), scăderea (-), înmulțirea (*), împărțirea (/), ridicarea la putere (** sau ^), împărțirea întregă (\, // sau **DIV**) și restul împărțirii întregi (**MOD**).

Instrucțiunea **LET** (opțională) permite și o atribuire multiplă. Dacă este cazul se realizează trunchierea valorii reale la întreg.

Cît privește instrucțiunea **PRINT** menționăm că în limbajul **ABASIC** linia terminal se consideră împărțită pe zone de lungime 14 caractere. Pentru a defini alte valori pentru lungimea liniei de imprimat (implicit 132) și lungimea unei zone (implicit 14), utilizatorul dispune de instrucțiunea **OPTION WIDTH** al cărei format este:

Format general
OPTION WIDTH < expresie 1 >, < expresie 2 >

unde valorile convertite la întregi pentru (expresie 1) și (expresie 2) reprezintă lungimea liniei, respectiv a zonei de imprimat.

Semnificația separatorilor ", " și "; " este aceeași ca la **BASIC-80** și **BASIC-PLUS**.

ABASIC **NEW NEW** **ABASIC**

În sfîrșit, o ultimă remarcă privind ștergerea programelor. Un program încărcat poate fi șters în totalitate prin execuția instrucțiunii **NEW**.

Și-acum nu vă mai rămîne decît să introduceți programul, să-l executați și să vă convingeți singuri de cele afirmate anterior.

TEMA 2

Răspundeți prin **DA** sau **NU** la următoarele întrebări:

- A reprezintă un nume de variabilă numerică acceptată de toate versiunile **BASIC** studiate.
- **LET** poate fi un nume de variabilă pentru **BASIC-80**.
- **AB** poate fi un nume de variabilă pentru **BASIC-COMMODORE**.
- Cuvîntul rezervat **LET** este obligatoriu pentru o instrucțiune de atribuire în **BASIC HC-85**, **TIM S**, **SPECTRUM**.
- Toate calculatoarele personale pot realiza operații matematice.
- Pentru calculatoarele **aMIC**, **INDEPENDENT**, **CORAL** numele unei variabile numerice este format dintr-o literă urmată de cel mult o cifră.

- Într-o variabilă BASIC-80, COMMODORE numai primele două caractere sînt semnificative.
- Caracterele "0/0", "I", "#" pot fi atașate unei variabile BASIC-80 și ABASIC.
- Caracterul "0/0" atașat unei variabile BASIC-80, ABASIC, BASIC-AMSTRAD, BASIC-COMMODORE, BASIC-PLUS precizează tipul întreg al unei variabile.
- Caracterul "#" poate fi atașat unei variabile BASIC-80, ABASIC dar nu poate fi atașat unei variabile BASIC-aMIC, GWBASIC.
- Rezultatul execuției următorului program este zero.

```
10 Y = 5
20 PRINT X + Y - 5
```

- Rezultatul execuției următorului program este 8.

```
10 I = 4
20 I = + 4
30 PRINT I
```

- Rezultatul execuției următorului program este 8.

```
10 W = 7
20 C = W + 1
30 PRINT 7
```

Înlocuiți cuvintele care lipsesc din următoarele propoziții:

a) Într-o variabilă BASIC numai primele două caractere sînt semnificative, pe cînd într-o variabilă BASIC numai primele 40 caractere sînt semnificative.

b) Pentru ștergerea programului curent din memoria calculatorului se utilizează comanda **SCR**.

c) Comanda **NEW** nu se utilizează pe calculatoarele, în timp ce comanda **SCR** este folosită doar pentru calculatorul

d) Separatorul apostrof este specific limbajului BASIC și face ca valorile să fie tipărite

e) Instrucțiunea **PRINT** poate fi înlocuită cu în limbajul BASIC implementat pe calculatoarele

f) Liniile multiinstrucțiune nu sînt admise în limbajul BASIC implementat pe calculatorul

Să se scrie cite un program BASIC pentru fiecare din problemele de mai jos:

a) Fiind date coordonatele punctelor M(3, 4) și N(-3.4, 5.75), să se calculeze și afișeze valoarea distanței $MP=PN$, unde P(X, Y) este mijlocul segmentului MN.

b) Să se calculeze și afișeze coordonatele punctului de intersecție al dreptelor de ecuații:

$$Y=3x+7$$

și

$$Y=-4x+14$$

Observație. Dreptele se intersectează deoarece pantele lor (3 și 4) sînt diferite între ele.

c) Relația pentru calculul inductanței unei bobine este $L = \frac{\mu_0 \mu_r N^2 S}{l}$

în care μ_0 reprezintă permeabilitatea vidului, μ_r – permeabilitatea relativă, N – numărul de spire, S – aria secțiunii, l – lungimea bobinei.

Să se calculeze inductanța unei bobine cunoscîndu-se: $\mu_0=4\pi\cdot 10^{-7}$ N/A², $\mu_r=300$, $N=1\,000$, $S=10^4$ cm², $l=62,8$ cm.

Observație. Valorile lui S și l se vor exprima în SI (sistemul internațional), respectiv m² și m.

d) Randamentul motorului Otto se exprimă prin relația $\eta=1-\frac{1}{\epsilon^{\gamma-1}}$

unde η reprezintă randamentul, $\epsilon=\frac{V_1}{V_2}$ reprezintă raportul de compresie, iar γ reprezintă exponentul adiabatic.

Să se calculeze randamentul unui motor Otto, dacă se cunoaște $\epsilon=5$ și $\gamma=1,4$ (substanța de lucru este aerul).

e) Să se calculeze randamentul motorului DIESEL cunoscînd $\epsilon=10$, $\rho=2$, $\gamma=1,4$. Se va utiliza relația:

$$\eta=1-\frac{\rho^{\gamma}-1}{\gamma\epsilon^{\gamma-1}(\rho-1)}$$

f) Să se calculeze constanta universală a gazelor (R) cunoscîndu-se: $p_0=101\,325$ N/m², $V_{\mu_0}=22,42$ m³/kmol, $T_0=273,15$ K. Se va utiliza relația:

$$R=\frac{p_0 V_{\mu_0}}{T_0}$$

g) Să se afle volumul cilindrului a cărui secțiune axială este un pătrat de arie $a=16$ m². Se va utiliza relația $V=\pi R^2 h = \frac{a\sqrt{a}}{4} \pi$.

h) Să se calculeze volumul conului înscris într-un tetraedru regulat de muchie $a=10$ m. Se va utiliza relația:

$$V=\frac{\pi a^3 \sqrt{6}}{108} \left(R=\frac{a\sqrt{3}}{6}, h=\frac{a\sqrt{3}}{6} \right)$$

i) Să se scrie ecuația perpendicularei care trece prin mijlocul dreptei determinată de punctele $A(1, 2)$ și $B(4, 7)$.

Observație. Diferențele x_2-x_1 și y_2-y_1 sînt diferite de zero (3 respectiv 5).

Să se scrie un program BASIC care calculează și afișează cantitățile totale trimestriale de jucării fabricate de întreprinderea „URSU-

LEȚUL", cunoscând cantitățile de jucării (în bucăți) fabricate în fiecare lună: ianuarie (3 000), februarie (8 000), martie (7 500), aprilie (6 000), mai (5 000), iunie (2 300), iulie (1 000), august (3 200), septembrie (400), octombrie (600), noiembrie (400), decembrie (9 000).

□ Să se scrie un program BASIC care calculează și afișează consumul mediu anual al unui articol cunoscând consumurile anuale (în bucăți) pe ultimii patru ani: 1984 (321), 1985 (432), 1986 (766), 1987 (876).

SOLUȚIA TEMEI 2

□ Răspunsurile sint:

DA, NU, DA, DA, DA, DA, DA, DA, DA, DA, DA, DA, DA, NU.

□ Cuvintele lipsă sint:

a) PRAE, COMMODORE; 80; b) aMIC; c) aMIC, AMSTRAD; d) HC-85, TIM S, SPECTRUM; una sub alta; e) COMMODORE, AMSTRAD, FELIX C; f) aMIC.

□ Programele BASIC sint:

a) Problema se reduce la calculul coordonatelor x și y ale punctului P. Deoarece punctul P se găsește la mijlocul distanței MN, putem scrie că:

$$x = \frac{x_1 + x_2}{2},$$

$$y = \frac{y_1 + y_2}{2}.$$

În program se vor atribui variabilelor x_1 , y_1 coordonatele punctului M, iar variabilelor x_2 , y_2 coordonatele punctului N. În final se va calcula distanța MP cu formula cunoscută (v. și aplicația de la pag. 181), după care se va tipări valoarea calculată.

Prezentăm în continuare lista programului rulat pe calculatorul personal FELIX PC.

```

NEW
20 y1 = 4
10 x1 = 3
30 x2 = -3.4
35 x = (x1 + x2)/2
40 y2 = 5.75
50 MP = (x2 - x) ↑ 2 + (y2 - y) ↑ 2) ↑ 0.5
60 PRINT "DISTANȚA MP = ", MP
45 y = (y1 + y2)/2

b) 10 LET M1 = 3
20 LET N1 = 7
30 LET M2 = -4
40 LET N2 = 14
50 PRINT "Y = ( "; M1; " ) *
X + ( "; N1; " ) "
60 PRINT "Y = ( "; M2; " ) *
X + ( "; N2; " ) "
70 LET X = (N2 - N1) / (M1 - M2)
80 LET Y = M1 * X + N1
90 PRINT
100 PRINT "PUNCTUL DE INTERSECȚIE
( "; X; ", "; Y; " ) "
110 END

```

Semnificația instrucțiunilor este următoarea:

- 10 Atribuire variabilei M1 panta primei drepte.
- 20 Atribuire variabilei N1 valoarea termenului liber al ecuației primei drepte.
- 30 Atribuire variabilei M2 panta celei de-a doua drepte.
- 40 Atribuire variabilei N2 valoarea termenului liber al ecuației celei de-a doua drepte.
- 50 Tipărește ecuația primei drepte $Y=(3) * X+(7)$.
- 60 Tipărește ecuația celei de-a doua drepte $Y=(-4) * X+(14)$.
- 70 Calculează abscisa punctului de intersecție.
- 80 Calculează ordonata punctului de intersecție.
- 90 Tipărește un rând liber.
- 100 Editează rezultatul final.

Remarcă. Înainte de tastarea acestui program, dați comenzile **SCR** (aMIC), și **NEW** (PRAE, HC-85, TIM S, SPECTRUM, AMSTRAD, COMMODORE, M-118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX-C) pentru ștergerea programului curent din memoria calculatorului. Utilizarea cuvintului **LET** în instrucțiunile 10, 20, 30, 40 70, 80 este obligatorie numai pentru calculatoarele HC-85, TIM S, SPECTRUM. Executați programul cu comanda **RUN**.

ABASIC

SCR

ABASIC

RUN

- | | |
|--|--|
| <p>c) 10 PRINT "INDUCTANTA BOBI-
NEI = "; (4 * PI / 10 ↑ 7 *
300 * 1000 ↑ 2) / (62 . 8 / 100); "H"
20 END</p> <p>d) 10 PRINT "RANDAMENTUL = ";
1 - 1 / 5 ↑ (1 . 4 - 1)
20 END</p> <p>e) 10 LET D = 2
20 LET E = 10
30 LET G = 1 . 4
40 LET R = 1 - (D ↑ G - 1) /
(G * E ↑ (G - 1) * (D - 1))
50 PRINT "RANDAMENTUL = ";
R</p> <p>f) 10 PRINT "R = "; 101325 *
22 . 42 / 273 . 15</p> | <p>g) 10 PRINT "VOLUMUL = ";
16 * 16 ↑ 0 . 5 * PI / 4
20 END</p> <p>h) 10 LET A = 10
20 LET V = PI * A ↑ 3 * 6 ↑ 0 . 5 / 108
30 PRINT "VOLUMUL = "; V
40 END</p> <p>i) 10 LET X1 = 1
20 LET Y1 = 2
30 LET X2 = 4
40 LET Y2 = 7
50 PRINT "("; X1; ", "; Y1; ") SI
("; X2; ", "; Y2; ") "
60 PRINT "ECUAȚIA ESTE: ";
70 LET X0 = (X1 + X2) / 2</p> |
|--|--|

```

80 LET Y0 = (Y1 + Y2) / 2
90 LET M = (Y2 - Y1) / (X2 - X1)
100 LET M0 = - 1 / M
110 LET N = Y0 - M0 * X0

```

```

120 PRINT " Y = "; M0; " * X +
      (" ; N; " ) "
130 END

```

Programul pentru calculul cantităților totale de jucării fabricate este:

```

10 LET I1 = 3000
20 LET F = 8000
30 LET M1 = 7500
40 LET A1 = 6000
50 LET M2 = 5000
60 LET I2 = 2300
70 LET I3 = 1000
80 LET A2 = 3200
90 LET S = 400
100 LET I4 = 600
110 LET N = 400
120 LET D = 9000
130 LET T1 = I1 + F + M1
140 LET T2 = A1 + M2 + I2
150 LET T3 = I3 + A2 + S

```

```

160 LET T4 = I4 + N + D
170 PRINT
180 PRINT "TOTAL JUCARII TRIMESTRUL
      1 = "; T1
190 PRINT
200 PRINT "TOTAL JUCARII TRIMESTRUL
      2 = "; T2
210 PRINT
220 PRINT "TOTAL JUCARII TRIMESTRUL
      3 = "; T3
230 PRINT
240 PRINT "TOTAL JUCARII TRIMESTRUL
      4 = "; T4
250 END

```

Programul pentru calculul consumului mediu anual este:

```

10 LET A1 = 321
20 LET A2 = 432
30 LET A3 = 766
40 LET A4 = 876
50 LET C = (A1 + A2 + A3 + A4) / 4
60 PRINT "CONSUMUL MEDIU ANUAL = "; C
70 END

```

CONVERSAȚIA 3

Proiectarea și realizarea unui program pentru calculul ariei și volumului unui rezervor sferic a cărui rază poate lua diferite valori. Metodică de analiză, proiectare și realizare a programelor BASIC. Introducerea dinamică a datelor. Instrucțiunea INPUT. Documentarea programelor. Instrucțiunea REM. Salvarea unui program BASIC pe medii magnetice. Depanarea unui program. APLICAȚII și TESTE pentru cititor



EXEMPLELE 3 – PC, m, M, F

□ Metodică de analiză, proiectare și realizare a programelor

Formularea problemei

Vom relua în cadrul acestei conversații problema rezervorului sferic, complicînd-o sub următorul aspect: nu mai stabilim de la început valoarea razei ($R=3$), ci o vom transmite calculatorului la solicitarea acestuia. De asemenea, ne propunem să calculăm și volumul rezervorului pe care să-l afișăm, alături de valorile razei și ariei (calculate).

Ce problemă simplă, veți exclama! Într-adevăr, problema nu este dificilă, dar trebuie să recunoaștem că ceea ce pare simplu pentru unii este foarte complicat pentru alții.

Desigur, ea poate fi rezolvată foarte bine și de către un elev de școală elementară sau chiar de un adult (părinții elevului!). Obiectivul nostru, însă, este rezolvarea informatică a problemei cu ajutorul calculatorului personal, microcalculatorului etc.

Înainte de a scrie un program trebuie să cunoaștem și să înțelegem foarte bine problema. În cazul nostru, problema este una dintre cele mai simple, calculul ariei și volumului unui rezervor sferic atunci cînd se cunoaște raza.

Procesul de alcătuire a programului, pe care îl vom urma pe tot parcursul acestei lucrări*, constă din următoarele faze: analiză, proiectare, codificare (implementare) în limbajul BASIC (pe calculatoare personale, microcalculatoare, mini și maxi calculatoare), testare și depanare.

Faza de analiză

În cadrul acestei faze se poartă o discuție cu persoana care a solicitat realizarea programului, urmînd să aflăm cum dorește să-i fie prezentată lista cu rezultate, care sînt datele de intrare, precum și intențiile programului solicitat. În rezolvarea acestei probleme se urmărește ca pe ecranul monitorului să se afișeze simplu, într-o formă deloc pretențioasă, cele trei valori pentru: rază, arie și volum (vezi modulul de analiză structurată, fig. 3.1 (a, b)).

În rezolvarea informatică a unei probleme, întotdeauna se pornește cu datele de ieșire, urmînd să ne întrebăm apoi dacă avem suficiente date de intrare pentru a putea fi obținute ieșirile dorite. În cazul nostru răspunsul este afirmativ.

Dacă răspunsul la această întrebare este negativ, vom defini încă un tip de date și anume **datele de stare**, care nu sînt nici de intrare și nici de ieșire. Asociînd fiecărui tip de date cîte un nume de **variabilă** obținem **variabilele de intrare, de ieșire și de stare** care, structurate într-o tabelă, alcătuiesc ceea ce numim **tabela de variabile**.

* Vezi și „Învățăm FORTRAN... conversînd cu calculatorul”, „Învățăm COBOL... conversînd cu calculatorul”.

R = A = V =

a)

RAZA (NR. REAL)		RAZA (NR. INTREG)
xx.xx		xx
R = xx.xx	A = xxx.xx	V = xxx.xx

b)

TABELA DE VARIABILE

Nume program: EXEMPLELE 3-PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieșire
R: raza rezervorului		A: aria rezervorului V: volumul rezervorului

c)

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 3-PC, m, M, F

Descrierea programului

Programul citește de la terminal valoarea razei (pe care o și tipărește), calculează, afișează aria și volumul rezervorului sferic.

Intrări

Raza rezervorului.

Ieșiri

Lista cu raza, aria și volumul rezervorului sferic.

Lista de funcțiuni ale programului

1. Citire rază
2. Calculul ariei rezervorului (sferic)
3. Calculul volumului rezervorului (sferic)
4. Afișarea rezultatului
5. Stop

d)

Fig. 3.1, a-e. Modulul de analiză structurată; a, b) formatul datelor de ieșire; c) tabela de variabile; d) specificațiile de programare;

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE	
Nume program: EXEMPLELE 3-PC, m, M, F	
Modul	Funcțiuni
CIT-R	1
PREL-R	2,3
REZULTATE	4,5

Fig. 3.1. e) alocarea funcțiilor de prelucrare.

Tabela de variabile pentru EXEMPLELE 3 – PC, m, M, F este reprezentată în modulul de analiză structurată, fig. 3.1 (c).

Specificații de programare

Înainte de a începe proiectarea unui program, trebuie să avem o descriere generală a acestuia, o identificare a intrărilor și a ieșirilor, precum și o listă a funcțiilor (pașilor) programului. Împreună, aceste elemente de documentare reprezintă specificațiile de programare. Ele sunt scrise în general de către un grup de analiști sau programatori, respectând o anumită metodologie. Specificațiile de programare pentru EXEMPLELE 3 – PC, m, M, F sunt prezentate în modulul de analiză structurată, fig. 3.1 (d).

Remarcă. Este important ca schemele și specificațiile de programare să fie validate și de persoana care a solicitat realizarea programului. Această aprobare ajută la asigurarea interpretării corecte a programului dorit. Dacă sînt erori, ele pot fi înlăturate înainte de a se fi depus un volum de muncă pentru proiectarea programului.

Faza de proiectare

Asigurîndu-ne că nu s-au strecurat greșeli în faza de analiză a problemei, putem aborda în continuare faza de proiectare a programului, utilizînd ca instrumente de proiectare: diagrama de structură, pseudocodul și schema logică.

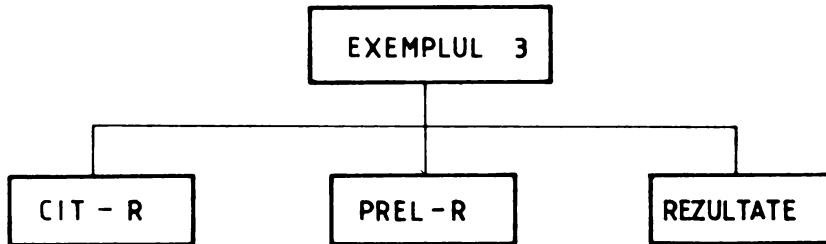
Diagrama de structură

Diagrama de structură (similară clasicei organigrame a unei întreprinderi) reflectă, într-un mod simplu de reprezentare grafică, structura logică de prelucrare a unui program, pe nivele de detaliere, conform următoarelor **reguli**.

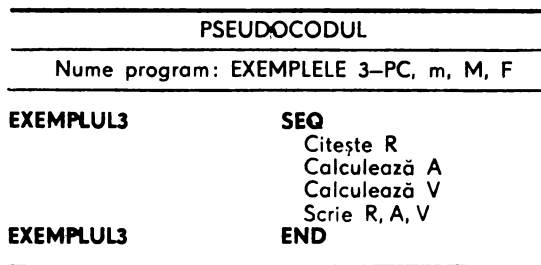
- Nivelele de detaliere se parcurg de sus în jos (TOP DOWN);
- Fiecare nivel reprezintă o detaliere a nivelului precedent;
- Blocurile situate pe un același nivel se parcurg de la stînga la dreapta;
- Terminarea parcurgerii unui nivel presupune parcurgerea următorului bloc din nivelul anterior.

Într-o diagramă de structură blocurile se reprezintă (grafic) printr-un dreptunghi în interiorul căruia se trece numele modulelor de prelucrare (vezi modulul de proiectare structurată, fig. 3.2 (a)).

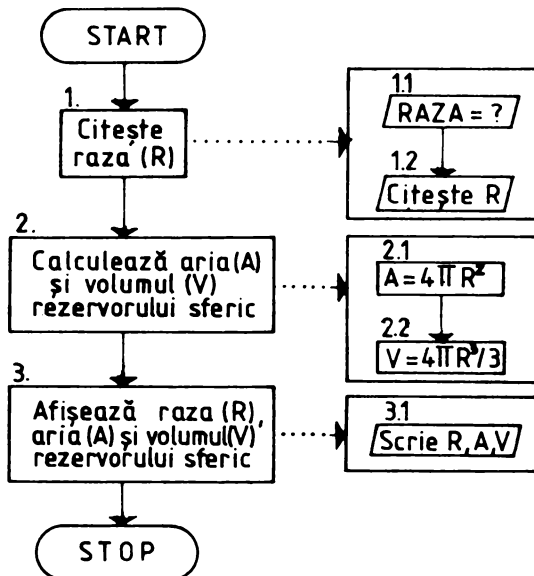
Funcțiunile modulelor de prelucrare (CIT-R, PREL-R, REZULTATE) corespund listei de funcțiuni ale programului (faza de analiză) din cadrul specificațiilor de programare (vezi modulul de analiză structurată, fig. 3.1(e)).



a)



b)



c)

Fig. 3.2. Modulul de proiectare structurată: a) diagrama de structură; b) pseudocodul; c) schema logică.

Dintre virtuțile diagramei de structură amintim următoarele: 1) este independentă de limbaj sau tip calculator; 2) poate reda o structură a datelor sau a logicii de prelucrare; 3) scoate în evidență structura funcțională a programului; 4) scoate în evidență necesarul de operații de intrare/ieșire.

Observație. Diagrama de structură prezintă și un dezavantaj: este depărtată de structura limbajelor de programare în general, de BASIC în particular.

Pseudocodul

Pseudocodul este folosit pentru a exprima detaliat, într-un text narativ, logica unui program. El a devenit o tehnică cunoscută deoarece se pretează la documentația programării structurate. Există variate „dialecte” de pseudocod. Unele tind să fie foarte pretențioase (dificile), altele nu. Indiferent de convențiile pseudocodului, el nu trebuie să fie ambiguu, ci destul de precis pentru a putea fi înțeles corect de programatori.

Limbajul pseudocod pe care vi-l propunem este alcătuit din câteva instrucțiuni standard ce definesc structurile algoritmice fundamentale (secvența, selecția, iterația) la care se adaugă instrucțiuni, în bună măsură la latitudinea celui care scrie programul.

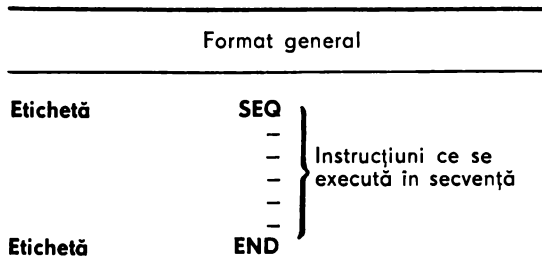
Remarcă. Deși pseudocodul poate fi similar cu actualul cod (limbaj) BASIC, mulți programatori cu experiență consideră că scrierea în pseudocod reprezintă un consum inutil de timp!

Pseudocodul corespunzător EXEMPLELOR 3 – PC, m, M, F este prezentat în modulul de proiectare structurată, fig. 3.2 (b).

Remarcați în descrierea pseudocodului instrucțiunile standard: **SEQ** și **END** (precedate de eticheta EXEMPLUL 3) care codifică o primă structură algoritmică cu care dvs. faceți cunoștință în acest moment: **structura secvențială**. În cazul cel mai simplu (cazul nostru), **structura secvențială** definește un grup de instrucțiuni (citește, calculează, scrie) care se execută una după alta.

În general, programele conțin **secvențe** alcătuite din grupuri de instrucțiuni care se execută numai în anumite condiții, cit și grupuri de instrucțiuni care se execută de atâtea ori cit timp sau pînă cînd este îndeplinită o condiție.

Formatul general al blocului de secvență este:



Remarci

- Instrucțiunile **SEQ** și **END** au obligatoriu aceeași etichetă;
- Instrucțiunea **SEQ** reprezintă punctul unic de intrare în bloc;
- Instrucțiunea **END** reprezintă punctul unic de ieșire din bloc.

Particularizînd formatul general al blocului de secvență la EXEMPLELE 3 – PC, m, M, F, observați cum EXEMPLUL 3 joacă rolul etichetei (aceeași pentru instrucțiunile **SEQ** și **END**, iar: citește R, calculează A, calculează V, scrie R, A, V reprezintă acțiuni (instrucțiuni) ce se execută în secvență.

Acțiuni primitive și neprimitive. Algoritm. De notat că toate cele patru acțiuni listate mai sus pot fi înțelese și executate de către calculator (sub formă de instrucțiuni BASIC) fără a mai fi nevoie de nici o altă informație suplimentară. Spunem că avem de-a face în această situație cu o succesiune de acțiuni primitive. Lista tuturor acțiunilor primitive constituie algoritmul problemei de rezolvat. Desigur, nu întotdeauna o problemă poate fi descompusă de la început în acțiuni primitive. De cele mai multe ori începem prin a identifica acțiunile neprimitive (opusul celor primitive) pe care, prin rafinări, descompuneri repetate, le transformăm în acțiuni primitive. Rafinarea unui algoritm este într-o oarecare măsură o chestiune personală. Pe măsură ce cîștigați experiență în dezvoltarea algoritmilor și în codificarea lor în limbajul BASIC, puteți constata că faceți tot mai puțină rafinare a algoritmilor.

Schema logică

Reprezintă o metodă grafică de documentare a programului. În cadrul schemelor logice se utilizează simboluri specifice (paralelogramul – pentru operații de citire/scriere; dreptunghiul – pentru calcule; dreptunghiul racordat la ambele capete – pentru START/STOP; romb – pentru decizii), legate între ele prin săgeți reprezentînd fluxul desfășurării execuției.

Ea constituie, alături de diagrama de structură, una din tehnicile de reprezentare a algoritmilor.

Schema logică a programului este ilustrată în modulul de proiectare structurată, fig. 3.2 (c).

Faza de implementare (codificare)

În timpul codificării au loc trei pași: scrierea programului (sursă) BASIC (pasul 1); introducerea programului (pasul 2) și execuția programului (pasul 3). Datorită lipsei de compatibilitate dintre interpretele BASIC, calculatoare și/sau sistemul de operare folosite, va trebui să ne însușim în parte, pentru fiecare calculator, setul de reguli specifice.

În ceea ce ne privește, vom căuta ca pe întreg parcursul lucrării să semnalăm particularitățile de programare care intervin.

Faza de testare și depanare

Așa cum o lucrare cu conținut științific sau literar, corectă din punct de vedere gramatical, poate avea un plan greșit, în mod similar un program BASIC fără greșeli de sintaxă poate opera greșit. Interpretorul BASIC nu dispune de nici o metodă pentru a cunoaște ceea ce dorește programa-

matorul să obțină prin program. În faza de testare, programatorul se asigură dacă programul realizat este în concordanță cu specificațiile de programare.

□ Codificarea în limbajul BASIC-aMIC

vol. 2, pag. 201

Scrierea programului (pasul 1). Reguli generale

Limbajul BASIC-aMIC dispune de câteva reguli generale de scriere (sintaxă) ce trebuie cunoscute și respectate cu strictețe.

REGULI

- Pentru scrierea unui program BASIC se folosesc numai litere mari.
- Fiecare linie program începe cu un număr de linie.
- Numărul de linie este un întreg cuprins între 1 și 32767.
- Liniile program conțin o singură instrucțiune (**linie simplă**).
- Numele variabilelor simple sînt formate dintr-o literă urmată de cel mult o cifră.
- Numele variabilelor indexate sînt formate dintr-o literă.
- Numărul de indici ai unei variabile indexate este de 1–2 (vector sau matrice).
- Sistemul elimină spațiile din orice șir de caractere, cu excepția celor din textele șirurilor de caractere (cuprinse între ghilimele).
- În aMIC instrucțiunea **END** trebuie să fie ultima instrucțiune din program (cu numărul de linie cel mai mare).
- Operatorii aritmetici admiși sînt: + (adunare); – (scădere); * (înmulțire); / (împărțire); ↑ (ridicare la putere).
- Operatorii de relație admiși sînt: = ; < ; > ; <= ; >= ; <> (diferit).
- Limbajul BASIC-aMIC nu admite operatori logici (**NOT, AND, OR**).
- Funcțiile matematice standard admise sînt: **PI, SIN, COS, TAN, ATN, LOG, EXP, SQR, ABS, INT, SGN, RND**.
- Funcțiile admise pentru lucrul cu șirurile de caractere sînt: **VAL, STR\$, LEN, CHR\$, INKEY\$**.
- Instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF... THEN, ON... GOTO**.
- Instrucțiunea de ciclare permisă este **FOR... NEXT**.
- Instrucțiunile de intrare/ieșire permise sînt: **INPUT, READ, DATA, RESTORE** (fără număr de linie), **PRINT, PRINT AT, MAT READ, MAT INPUT, MAT PRINT**.
- Alte funcții (pentru prelucrarea matricilor) admise sînt: **TRN, ZER, CON, IDN**.

În cele ce urmează vom aplica aceste reguli la scrierea programului.

După cum vă reamintiți (vezi fazele de analiză și proiectare), programul citește de la tastatură valoarea razei rezervorului, calculează aria și volumul rezervorului iar în final tipărește rezultatele.

Instrucțiunea REM

Vă rugăm să nu treceți încă la tastatură! Întii vom scrie programul și numai după aceea îl vom introduce, executa și dacă este nevoie, îl vom și depana. Pentru scrierea acestui program, vă propunem să acceptați și colaborarea cu noi! Vom începe prin a „da programului un titlu”.

10 REM EXEMPLUL 3 – PC (AMIC)

A nu se crede că aceasta este o obligație din partea dvs. Nul Titlul pe care l-ați dat vă va ajuta mai târziu la regăsirea acestui program, deoarece pe parcursul lucrării se vor aduna mai multe programe. EXEMPLUL 3 – PC(AMIC) reprezintă pentru program un comentariu sau o declarație. În limbajul BASIC orice comentariu se introduce prin instrucțiunea **REM**.

Formatul instrucțiunii este:

Format general
<număr linie> REM <mesaj>

unde <mesaj> reprezintă comentariul program.

Remarci

- Instrucțiunea **REM** poate apare oriunde în cadrul unui program.
- Se recomandă includerea între ghilimele a comentariului (vezi regula).

De notat că la execuția programului toate liniile care încep cu **REM** sînt ignorate. **REM**-urile prezintă importanță numai pentru cel care citește programul. Ele au un rol deosebit pentru documentarea programelor BASIC.

Vă propunem să listați în continuare, utilizînd instrucțiunea **REM**, funcțiunile programului:

20 REM "PROGRAMUL CITEȘTE RAZA (R)" 30 REM "A UNUI REZERVOR SFERIC," 40 REM "CALCULEAZĂ ARIA (A),"	50 REM "VOLUMUL (V) ȘI TIPĂREȘTE:" 60 REM "RAZA, ARIA, VOLUMUL"
---	--

Instrucțiunea **INPUT**

Acum dorim să citim (**INPUT**) într-o variabilă numerică (R) valoarea razei rezervorului sferic. Cum primesc variabilele valori? Există mai multe modalități, și anume: 1) prin atribuire propriu-zisă (instrucțiunea **LET**); 2) prin introducere dinamică (instrucțiunea **INPUT**) și 3) prin introducere statică (instrucțiunile **READ** și **DATA**).

Remarci

- Între două atribuiri valoarea unei variabile rămîne constantă.
- În momentul lansării în execuție a unui program toate variabilele din program primesc valoarea zero.

Dacă în programele dvs. plasați instrucțiunea **INPUT**, calculatorul se va opri, va afișa un simbol (două puncte, semnul întrebării etc.) și va aștepta răspunsul dvs. Așadar, instrucțiunea **INPUT** reprezintă un excelent mod de introducere a datelor în calculator. Ea plasează valorile ce urmează a se introduce într-o variabilă. În instrucțiunea **INPUT** se precizează variabilele de intrare definite în tabela de variabile.

Formatul general al instrucțiunii este:

Format general
<număr linie> INPUT <variabilă> { ,<variabilă> }

unde, <variabilă> reprezintă orice variabilă admisă, indiferent de tip.

Valorile variabilei se introduc în mod conversațional în timpul execuției programului.

Observație. Se recomandă utilizarea instrucțiunii **INPUT** ori de câte ori datele necesare execuției unui program nu sînt foarte numeroase.

Într-o instrucțiune **INPUT** putem întîlni deopotrivă variabile numerice, alfanumerice etc. Întrebarea care se pune este "cum procedăm într-o astfel de situație?".

Pentru a elimina orice confuzie sau vreun eventual blocaj al programului (atunci cînd răspunsul la întrebare nu este corect calculatorul afișează un mesaj de eroare și de nenumărate ori se blochează) se plasează înaintea instrucțiunii **INPUT** o instrucțiune **PRINT** cu texte explicative care să specifice numele variabilelor ce trebuie evaluate de la tastatură.

Considerați că avem suficiente elemente pentru programarea în BASIC a primului pas al algoritmului: "Citește R?" Dacă da, scrieți următoarele două instrucțiuni:

```
70 PRINT "RAZA=";
80 INPUT R
```

Observație. Am pus " ; " (vezi linia 70) deoarece dorim ca la execuția programului valoarea razei (R) să se introducă în continuarea mesajului: "RAZA=". În caz contrar, ar fi apărut pe rîndul următor.

După cum constatați, de-abia acum'am ajuns la momentul: "Conversația 2" (R=3 sau orice altă valoare doreați!). Pentru cei care au înțeles programul din conversația precedentă, nimic nu mai este complicat de aici înainte. Urmăriți documentația de proiectare și completați programul dvs. cu următoarele instrucțiuni:

```
90 A=4 * PI * R ↑ 2
100 V=4 * PI * R ↑ 3/3
110 PRINT "R="; R ; " A="; A ; " V="; V
120 END
```

Introducerea programului (pasul 2).

Reguli generale

Calculatoarele sînt virtual nefolositoare dacă nu ar exista modalități de introducere în memoria lor a datelor și programelor noastre.

Calculatorul personal aMIC'ia cunoștință de programul pe care i l-am stabilit atît prin folosirea unor medii magnetice (casete), cît și prin intermediul unor medii "electronice" (console cu tastatură).

În vederea introducerii și corectării interactive a programelor scrise în limbajul BASIC-aMIC se vor avea în vedere următoarele **reguli**:

- linia program BASIC poate fi tastată numai după apariția cursorului " ■ ";
- pentru ca linia să fie prelucrată se va acționa tasta **[RETURN]**;
- listarea instrucțiunilor unui program (toate sau numai o parte) se realizează cu comanda **LIST**;
- ștergerea ultimului caracter introdus se realizează prin apăsarea tastei **[DEL]**;

- ștergerea unei linii în curs de introducere se realizează acționind simultan **[CTRL]** și **[V]**;
- ștergerea unei linii program se realizează prin scrierea numărului liniei, urmată de acționarea tastei **[RETURN]**;
- introducerea unei instrucțiuni cu un număr de linie deja existent în program are ca efect înlocuirea vechii linii program cu cea nouă;
- ștergerea din memorie a unui program pentru a face posibilă introducerea unui nou program se realizează cu comanda **SCR**;
- ștergerea ecranului în vederea scrierii sau desenării pe pagină nouă se realizează cu **INIT**.

Și-acum, vă invităm la muncă! Tastați instrucțiunile programului respectând întocmai regulile sintetizate mai sus. Succes!

Salvarea unui program BASIC pe casetă. Un program BASIC aflat în memorie poate fi salvat pe casetă magnetică utilizând comanda (BASIC) **SAVE**. Procedura de salvare se realizează în cinci pași, după cum urmează:

1) Se tastează **SAVE**, **[RETURN]**, patru caractere hexazecimale (numele programului) în această ordine.

2) Se poziționează caseta în poziția dorită.

3) Se apasă la casetofon clapele 1 (roșie) și a 4-a (de la stânga la dreapta).

4) Se apasă tasta **[RETURN]**.

5) Apariția mesajului **READY** dă de știre că salvarea programului a reușit.

Aplicație. Salvați programul pe o casetă audio sub numele PC3A.

Citirea de pe casetă a unui program BASIC. Procedura de citire a unui program BASIC înregistrat pe o casetă (cu comanda **BASIC-SAVE**) este următoarea:

1) Se tastează comanda **LOAD** urmată de **[RETURN]** și de numele programului (patru caractere hexazecimale).

2) Se poziționează caseta cât mai aproape de locul în care a fost înregistrat programul.

3) Se apasă clapele 4 și 6 ale casetofonului.

4) Se eliberează clapa 6 și se apasă pe **[RETURN]**.

5) Apariția mesajului **READY** indică o citire de program reușită.

6) În caz de eroare la citire, apare mesajul **TAPE ERROR** și operațiile trebuie reluate!

Aplicație. Citiți de pe casetă programul PC3A salvat anterior. Dacă ați reușit, continuați. Dacă nu ați reușit, continuați!

Execuția programului (pasul 3).

Reguli generale

Procesul de execuție a unui program trebuie înțeles astfel: 1) se transmite programul sursă, ca intrare, interpretorului BASIC; 2) interpretorul analizează fiecare instrucțiune, verifică existența erorilor, după care execută instrucțiunea BASIC solicitată. Pentru a executa instrucțiunea respectivă, interpretorul traduce instrucțiunea din codul ASCII într-un număr de instrucțiuni mașină executabile, pe care le execută în continuare.

Erorile pe care interpretorul le poate detecta se referă în general la erorile sintactice ale programului. Astfel de erori apar atunci cînd limbajul folosit în programul sursă nu este în conformitate cu regulile limbajului BASIC-aMIC. Fiecare eroare trebuie corectată astfel încît la o nouă execuție interpretorul BASIC să nu mai semnaleze nici o eroare sintactică.

În vederea lansării în execuție a unui program scris în limbajul BASIC-aMIC trebuie cunoscute și respectate următoarele **reguli**:

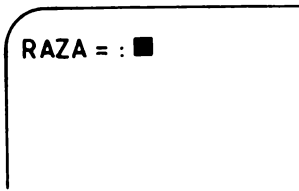
- lansarea în execuție a unui program începînd cu prima instrucțiune se realizează cu comanda **RUN**;
- lansarea în execuție a unui program începînd cu orice instrucțiune se realizează cu comanda **RUN** [(număr linie)] sau cu instrucțiunea **GOTO** (număr linie);
- depistarea unei erori în timpul execuției unui program duce la apariția mesajului:

ERROR (cod) IN LINE (număr linie)

și la abandonarea execuției programului;
Lista mesajelor de erori se găsește în vol. 2, pag. 2;

- întreruperea execuției unui program se realizează acționînd tasta **[INT]**. Reluarea execuției se face prin apăsarea tastei **[C]**.

Destul cu teoria! Să trecem la execuția programului pe care sperăm că l-ați încărcat de pe casetă. Tastați comanda **RUN!** A început dialogul. Pe ecran s-a afișat **RAZA=**, ca urmare a execuției instrucțiunii din linia 70.



RAZA = : ■

Nimic nou pînă acum. Între timp, după semnul egal au apărut două puncte. Le-ați remarcat? Ele provin de la instrucțiunea **INPUT**. Ori de cîte ori se execută instrucțiunea **INPUT**, calculatorul afișează pe ecran " : " întrebînd: "ce valoare doriți să atribuiți variabilei R?"

Cum veți răspunde dvs. la această . . . provocare?

Tipăriți o valoare numerică. Ați tastat cumva 3, valoarea razei din conversația precedentă? Puteți introduce, desigur, orice valoare doriți!

După ce ați introdus valoarea numerică a lui R (3, de exemplu), calculatorul așteaptă pînă apăsați pe **[RETURN]**.

Calculatorul aMIC, ca de altfel toate calculatoarele, este foarte, foarte răbdător, el vă va aștepta oricît, staționînd în linia 80 pînă cînd veți tipări o valoare numerică și veți apăsa pe tasta **[RETURN]**.

Deci, apăsați pe **[RETURN]**. Calculatorul reia execuția programului de la linia 90 începînd cu calculul ariei și volumului rezervorului sferic și continuînd apoi cu tipărirea valorilor calculate, specificate în linia 110.


```

RUN
R=3  A=113.097  V=113.097

```

În acest moment putem spune că programul este în întregime executat.

Acum puteți introduce o altă comandă **RUN**, răspunzând la instrucțiunea **INPUT** cu orice valoare doriți. În acest fel vă puteți „juca” oricât doriți iar execuția programului nu va fi reluată pînă ce nu veți apăsa pe tasta **[RETURN]**.

Observație. Comparînd acest program cu cel din conversația precedentă puteți constata avantajul net al acestuia. Prin utilizarea instrucțiunii dinamice (**INPUT**) de introducere a datelor (în acest program) nu mai este nevoie să specificați valorile numerice în cadrul programului (cu instrucțiunea **LET**). Valorile numerice sînt introduse de la claviatură, în timpul procesului de execuție a programului.

Bineînțeles, puteți experimenta cu acest program atîta timp cît el rezidă în memoria calculatorului. Introduceți doar comanda **RUN** ori de cîte ori doriți să reluați programul, răspunzînd apoi celor două puncte generate de instrucțiunea **INPUT**, cu valori numerice.

Aplicație. Scrieți un program BASIC-aMIC care să afișeze:

```

RUN
RAZA CERCULUI = 8
LUNGIMEA CERCULUI =
10 PRINT "RAZA CERCULUI = ";
20 INPUT R
30 L = 2 * PI * R
40 PRINT "LUNGIMEA CERCULUI = "; L
50 END

```

```

50 PRINT "X-Y="; X-Y
60 END
a3) 10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X+Y="; X+Y
60 END
a4) 10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X/Y="; X/Y
60 END

```

TESTE

a) Restrîngeți următoarele patru programe BASIC într-unul singur:

```

a1) 10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X+Y="; X+Y
60 END

a2) 10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y

```

```

R. 10 PRINT "X=";
20 INPUT X
30 PRINT "Y=";
40 INPUT Y
50 PRINT "X+Y="; X+Y
60 PRINT "X-Y="; X-Y
70 PRINT "X * Y="; X * Y
80 PRINT "X/Y="; X/Y
90 END

```

b) Scrieți un program BASIC care să calculeze și să afișeze valoarea expresiei $A*B+C$ pentru anumite valori de

intrare ale lui A, B și C. Prezentați două variante.

- b1) 10 INPUT A, B, C
 20 D=A * B+C
 30 PRINT "A * B+C="; D
 40 END
- b2) 10 INPUT A, B, C
 20 PRINT "A*B+C="; A*B+C
 30 END

c) Fiecare din următoarele instrucțiuni BASIC conține cite o eroare. Identificați eroarea și scrieți corect fiecare instrucțiune în parte.

```
10 INPUT X; Y
20 "X+Y"; X+Y
30 LET X+2=Y
40 INPUT X, Y, Z,
50 INPUT, X, Y
60 PRINT "X+2="; X+2
70 IMPUT X, A
```

d) Un corp cu masa $m=1,00$ kg este aruncat orizontal cu viteza inițială $v_0=20$ m/s. Să se calculeze energia cinetică a corpului după un interval de timp $\Delta t=4$ s.

Indicație. Se va utiliza formula de calcul $E_c = \frac{1}{2} m (v_0^2 + g^2 \Delta t^2)$. Valorile variabilelor de intrare se vor introduce dinamic.

RUN

M, V0, T0 : 1, 20, 4

ENERGIA CINETICĂ = 969.8888 J

Ready



e) Precizați ce greșeală (de logică) s-a strecurat la introducerea următorului program BASIC care calculează aria și lungimea unui cerc de rază R.

```
10 REM PROGRAMUL CALCULEAZA
   ARIA (A) ȘI
20 REM LUNGIMEA (L) UNUI CERC
30 PRINT "RAZA=";
40 INPUT R
50 LET L=2 * PI * R
60 A=PI * R * 2
70 PRINT "R="; R, "L="; L,
   "A="; A
80 END
```

R. 60 A = PI * R ↑ 2

Varianta de utilizare a instrucțiunii INPUT

Cu o instrucțiune **INPUT** se pot atribui una, două sau mai multe valori variabilelor din cadrul programului.

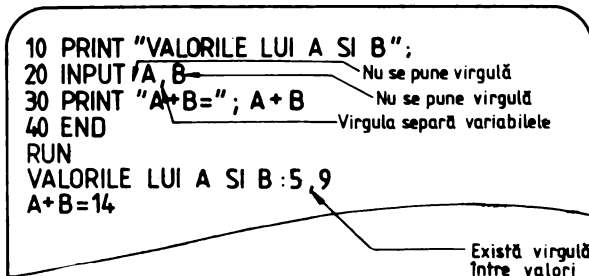


Fig. 3.3.

În fig. 3.3 se prezintă un exemplu de utilizare a instrucțiunii **INPUT** pentru citirea dinamică a valorilor variabilelor A și B.

De notat că 5 este valoarea alocată primei variabile (A), iar 9 este valoarea alocată celei de-a doua variabile (B).

□ Codificarea în limbajul BASIC-PRAE

vol. 2, pag. 201

Scrierea programului. Reguli generale

Regulile generale de scriere (sintaxă) în limbajul BASIC-PRAE sînt următoarele:

- pentru scrierea unui program BASIC-PRAE se folosesc numai litere mari;
- fiecare linie-program începe cu un număr de linie;
- numărul de linie este un număr întreg format din 1 pînă la 5 cifre, putînd avea valoarea cuprinsă între 1 și 65529;
- o linie program poate conține una (linie simplă) sau mai multe (linie multi-instrucțiune) instrucțiuni separate între ele prin două puncte (" : ");
- numele variabilelor numerice simple sînt formate dintr-o literă urmată de oricîte caractere alfanumerice (**numai primele două caractere sînt luate în considerare**);
- numele variabilelor indexate – idem;
- numărul de indici ai unei variabile indexate este de la 1 la 255;
- operatorii aritmetici admiși sînt: + (adunare); - (scădere); * (înmulțire); / (împărțire); † (ridicare la putere);
- operatorii de relație admiși sînt: =; <; >; <=; >=; <>;
- operatorii logici admiși sînt: **NOT, AND, OR**;
- funcțiile matematice admise sînt: **PI, SIN, COS, TAN, ATN, LOG, EXP, SQR, ABS, INT, SGN**;
- funcțiile grafice admise sînt: **PLOT, PLOT, DRAW, DRAWC, CIRCLE, CIRCLEC**;
- funcțiile pentru lucrul cu șirurile de caractere admise sînt: **LEFT\$, RIGHT\$, MID, VAL, STR, LEN, CHR\$, ASC**;
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF ... THEN ... ELSE, ON ... GOTO**;
- instrucțiunea de ciclare permisă este **FOR ... NEXT**.

Să aplicăm aceste reguli la scrierea programului. Imaginea acestuia (vezi vol. 2, pag. 201) este identică cu cea a programului precedent. În cele ce urmează vom preciza cîteva particularități privind instrucțiunile **REM** și **INPUT**.

Remarci

- Pe o linie multiinstrucțiune, instrucțiunea **REM** trebuie să fie ultima instrucțiune a liniei (Exemplu: 10 A=2: **REM** INIȚIALIZARE). În caz contrar, textul instrucțiunii următoare se consideră ca făcînd parte din textul comentariului.
- În locul lui **REM** (trei litere) se poate utiliza apostroful, care are aceeași semnificație cu **REM**.
- Utilizarea apostrofului pentru comentarii oferă unele facilități în plus: poate ocupa prima poziție într-o linie multiinstrucțiune și poate termina o linie simplă sau multiplă (multiinstrucțiune) fără a fi precedat de caracterul de separare instrucțiuni " : " (Exemplu: 10 A=4*PI*R † 2 ' ARIA SFEREI).
- În BASIC-PRAE există posibilitatea cuplării instrucțiunilor **PRINT** și **INPUT**. Începînd cu conversația următoare vom folosi și noi această facilitate importantă a interpretorului.

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii interactive a programelor BASIC-PRAE sint:

- linia program poate fi tastată numai după apariția cursorului "←";
- pentru ca linia să fie prelucrată se va acționa tasta **[CR]**;
- listarea instrucțiunilor unui program (toate sau numai o parte) se realizează cu comanda **LIST**;
- ștergerea ultimului caracter introdus se realizează prin acționarea tastei **[DEL]**;
- ștergerea unei linii program se realizează prin scrierea numărului liniei, urmată de acționarea tastei **[CR]**;
- o metodă foarte eficientă de a corecta greșelile din liniile unui program este folosirea funcției **EDIT**;
- ștergerea din memorie a unui program pentru a face posibilă introducerea unui nou program se realizează cu comanda **NEW**;
- ștergerea ecranului se realizează cu instrucțiunea **CLS**;
- generarea automată a numerelor de linie se face cu comanda **AUTO** al cărei format general este:

Format general

AUTO{<nr. linie>},{<pas>}

Observație. Dacă <nr. linie> și <pas> lipsesc, se generează numere de linie începând cu 10 și pasul 10. Revenirea în modul de numerotare manuală se face apăsând de două ori pe **[CR]**;

- Renumeratoarea liniilor se face cu comanda **RENUMBER** al cărei format general este:

Format general

RENUMBER{<nr. linie>, <pas>, <nr. linie minim>}

unde: <nr. linie> reprezintă numărul de linie de la care se începe renumerotarea; <pas> este mărimea pasului dintre numerele de linie atribuite la două linii consecutive, iar <nr. linie minim> reprezintă numărul de linie minim ce se renumerează.

Salvarea unui program pe casetă magnetică

Pentru păstrarea programelor BASIC-PRAE avem nevoie de un casetofon obișnuit (DECK EM-1001, Dana etc.) și de o casetă tot din cele obișnuite (se pot folosi și magnetofonele). Procedura de salvare a unui program este următoarea:

- 1) Se tastează **ASAVE** <nume>, unde <nume> reprezintă numele programului ales de utilizator, format din maximum 16 caractere.
- 2) Se poziționează caseta în punctul dorit. Înainte de a acționa **[CR]** se pot tasta câteva blancuri pentru a verifica nivelul semnalului de înregistrare la casetofon (trebuie să fie în jur de zero decibeli). Dacă nivelul este reglat corect se poate porni casetofonul și apoi se poate tasta **[CR]**.
- 3) Se apasă clapele 1 (roșie) și 4 (de la stînga la dreapta).
- 4) Se tastează **[CR]**.
- 5) În timpul înregistrării se aude un sunet specific. Când calculatorul PRAE a terminat transmiterea programului, pe ecran apare mesajul "TRY

TO READ" ca o invitație de a încerca să citim ceea ce am înregistrat pe casetă. Se apasă clapele [5], [2], [5], [4], [CR] în această ordine.

6) Corectitudinea salvării este indicată de apariția mesajului **READY**. În cazul apariției unei erori la înregistrare, se afișează mesajul "TAPE ERROR". În această ultimă situație, ce reprezintă un caz nefericit, trebuie să încercăm salvarea programului cu un alt nivel de înregistrare al casetofonului, sau, eventual, pe o altă porțiune a casetei magnetice.

Aplicație. Salvați programul pe o casetă audio, sub numele: PC3PRAE.

Citirea de pe casetă a unui program BASIC

Citirea unui program BASIC înregistrat pe o casetă se face cu comanda **ALOAD** al cărei format general este:

Format general
ALOAD "<nume>"

După acționarea tastei [CR] calculatorul PRAE este pregătit să găzduiască programul. Pot să apară două situații: a) pe casetă se găsește programul chemat cu comanda **ALOAD**; b) pe casetă se găsește un program diferit de cel specificat în comanda **ALOAD**. În cel de-al doilea caz, PRAE afișează numele acestuia. Când calculatorul găsește programul solicitat (cazul a), el procedează imediat la încărcarea programului (blocurilor). Așteptați câteva secunde pînă la apariția mesajului **READY** (timpul necesar conversiei). Dacă în timpul încărcării programului s-a detectat undeva o eroare, veți citi mesajul "TAPE ERROR" și după câteva secunde se trece în **READY**. În cazul cînd nu v-a reușit prima tentativă de încărcare a programului, "numărați pînă la zece" (nimic mai mult!) și reluați operațiunea de la comanda **ALOAD**. Cauzele nereușitei se pot datora fie stării necorespunzătoare a capului de înregistrare, fie vitezei casetei magnetice sau chiar unor deteriorări ale acesteia.

Și-acum, pe scurt, procedura de citire a unui program BASIC înregistrat pe o casetă:

- 1) Se tastează comanda **ALOAD**.
- 2) Se poziționează caseta cit mai aproape posibil de locul în care a fost înregistrat programul.
- 3) Se tastează clapele [4], [6] ale casetofonului în această ordine.

- 4) Pe ecran se listează numele tuturor programelor existente pînă la programul căutat.

- 5) La o citire corectă a programului se va afișa mesajul **READY**. În caz de eroare, vă va întîmpina mesajul TAPE ERROR, iar operațiile trebuie reluate.

Metodă specială de salvare a unui program BASIC-PRAE

Pentru salvarea programelor BASIC-PRAE deosebit de lungi, se recomandă [36] utilizarea comenzii **SAVE** al cărei format general este:

Format general
SAVE <nume program>&4100, PEEK(&415E)+PEEK(&415F)*256+1

Cu această comandă se salvează o zonă de memorie care începe cu primele variabile de sistem BASIC și se termină cu adresa de sfârșit a programului BASIC. Metoda se aplică pentru calculatoarele PRAE cu aceeași capacitate de memorie (variabilele de sistem BASIC diferă de la un PRAE cu 16 K de cele de la un PRAE cu 48 K).

Cu comanda **SAVE** puteți salva un program BASIC împreună cu variabilele sale simple. Formatul general este:

Format general

SAVE <nume program>&4100, **PEEK**(&4160)+**PEEK**(&4161)*256+1

Comenzile pentru salvarea și încărcarea imaginilor de pe ecran au următoarele formate generale:

Format general

SAVE <nume&6000>, &8000

Format general

SAVE <nume&E000>, &FFFF

Format general

LOAD <nume&6000>

Format general

LOAD <nume&E000>

Remarci

- Comanda **ASAVE** convertește programul curent aflat în memorie în cod ASCII și îl salvează pe suportul extern (casetă magnetică).
- Comanda **ALOAD** permite încărcarea în memoria calculatorului a programelor scrise în cod ASCII. Fiecare linie începe cu un număr de linie și se termină cu [CR] (retur de car). Sfirșitul se detectează fie prin codul CTRL/Z din text, fie prin marcajul EOF.
- Comanda **SAVE** produce salvarea blocului de memorie curent din memoria internă pe suport extern. Formatul general este:

Format general

SAVE <nume fișier> <adresa de la care se salvează> <adresa pînă la care se salvează>

- Comanda **LOAD** servește la încărcarea unui program la <adresă început> de pe un suport extern indicat.

Format general

LOAD <nume fișier> <adresă început>

- Un program salvat cu **SAVE** nu poate fi încărcat cu comanda **ALOAD**.
- Un program salvat cu **ASAVE** poate fi încărcat cu **ALOAD**.
- Comanda **AMERGE** servește pentru încărcarea și interclasarea unui program de pe casetă cu programul aflat în memorie. În timpul căutării programului (nume fișier) se listează numele tuturor programelor întilnite pe suport.

Format general

AMERGE (nume fișier)

Aplicație. Citiți de pe casetă programul PC3PRAE, respectind regulile listate anterior.

Execuția programului. Reguli generale

Pentru lansarea în execuție a unui program BASIC-PRAE, trebuie cunoscute și respectate următoarele **reguli** generale:

- Comanda **RUN** realizează lansarea în execuție a programului curent existent în memoria internă a calculatorului, de la instrucțiunea cu cel mai mic număr de linie;
- comanda **RUN** realizează lansarea în execuție a programului curent de la instrucțiunea care are numărul de linie egal cu (nr. linie);

Format general

RUN (nr. linie)

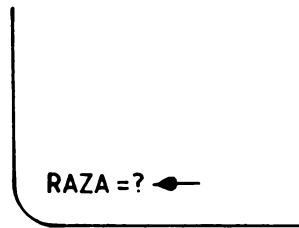
- comanda **RUN** poate fi lansată și dintr-un program dacă este precedată de un număr de linie.

Format general

(nr. linie) **RUN**

- întreruperea execuției unui program se realizează acționind tasta **[CTRL]**. Re-luarea execuției se face prin apăsarea tastei **[C]**;
- lista mesajelor de eroare BASIC-PRAE se găsește în vol. 2, pag. 3.
- comanda **CONTINUE** permite reluarea execuției programului din punctul de întrerupere realizat prin instrucțiunea **STOP**;
- comanda **CONTINUE** nu poate înlocui comanda **RUN** și nu are sens decât în punctele de întrerupere ale unui program lansat prin **RUN**.

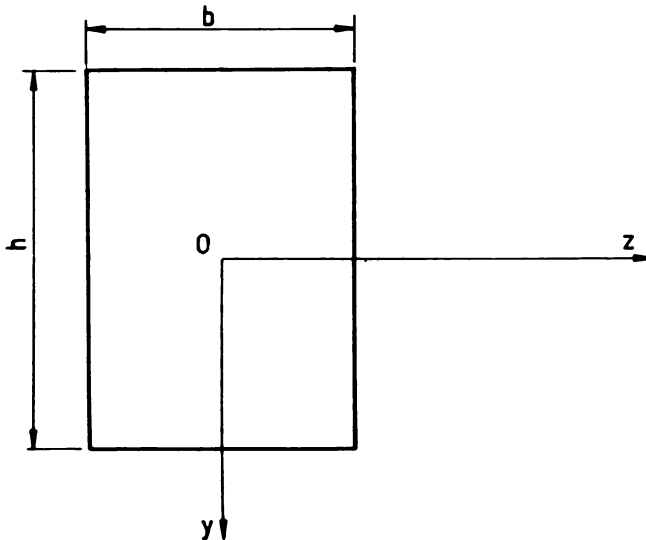
Ați încărcat programul de pe casetă în memoria calculatorului? Dacă nu, vă rugăm s-o faceți acum. Tastați apoi **RUN** și urmăriți imaginea de pe ecran. După ce calculatorul a afișat "RAZA=", remarcați cum, imediat, în continuarea mesajului, a apărut un semn de întrebare. Din acest moment calculatorul așteaptă . . . valoarea razei rezervorului, pe care vă rugăm s-o introduceți tastind 3, **[CR]** în această ordine. Dacă vă permiteți fantezii și introduceți ceva diferit de un număr (nu neapărat 3!), veți primi mesajul "INVALID INPUT" și din nou își va face apariția prietenul nostru curios – semnul întrebării.



Remarcați pe rîndul următor valorile ariei și volumului rezervorului sferic. Între timp a apărut și **READY** care parcă vrea să spună "pe cînd o nouă conversație?"

Aplicație. Să se calculeze momentele de inerție axiale (I_z , I_y) pentru suprafața dreptunghiulară din figura de mai jos, în raport cu axele Oz și Oy .

Indicație. Se vor utiliza formulele de calcul: $I_z = \frac{bh^3}{12}$; $I_y = \frac{b^3h}{12}$.



Specificațiile de programare și documentația de proiectare sînt ilustrate în

modulele de analiză și proiectare structurată.

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieșire
B: lățimea		I_Z : momentul de inerție axial în raport cu axa OZ
H: lungimea		I_Y : momentul de inerție axial în raport cu axa OY .

a)

Fig. 3.4. Modulele de analiză și proiectare structurată: a) tabela de variabile;

SPECIFICAȚII DE PROGRAMARE

Descrierea programului

Programul citește de la terminal lungimea și lățimea secțiunii, calculează momentele de inerție axiale și afișează rezultatul.

Intrări

Lungime, lățime

Ieșiri

Momentele de inerție axiale IZ, IY

Lista de funcțiuni ale programului

1. Citește date intrare
2. Calculează momentul de inerție în raport cu axa OZ
3. Calculează momentul de inerție în raport cu axa OY
4. Afișează rezultatul
5. Stop.

b)

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Modul	Funcțiuni
CIT-DAT	1
PREL-DAT	2,3
LIST	4
SFIRȘIT	5

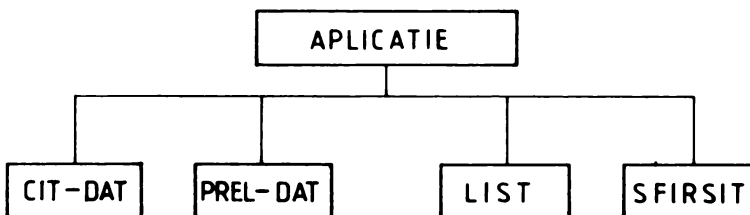
c)

DATELE DE INTRARE

B	H
20*	60*

* in cm.

d)



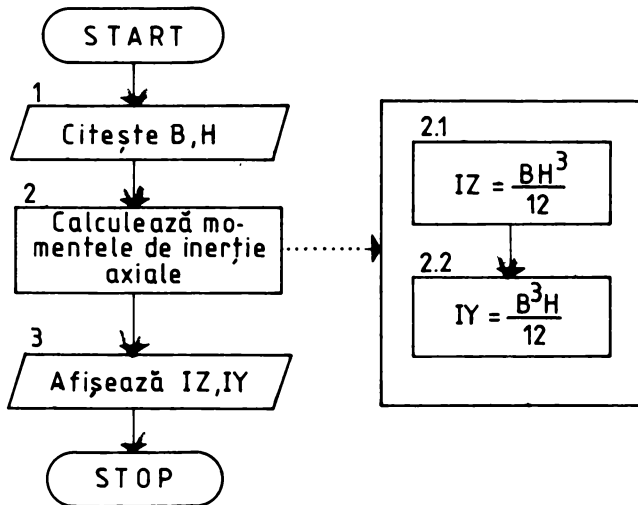
e)

Fig. 3.4. a-g. Modulele de analiză și proiectare structurată: b) specificații de programare; c) alocarea funcțiunilor de prelucrare; d) datele de intrare; e) diagrama de structură;

PSEUDOCODUL

APLICAȚIE	SEQ INPUT B, H
PREL-DAT	SEQ IZ=(B*H↑3)/12 IY=(B↑3*H)/12
PREL-DAT	END PRINT "IZ="; IZ, "IY="; IY
APLICAȚIE	END

f)



g)

f) pseudocodul; g) schema logică.

```

10 REM PROGRAMUL CALCULEAZĂ
    MOMENTELE
20 REM DE INERȚIE AXIALE PEN-
30 REM TRU SUPRAFAȚA DREPT-
    UNGHIULARĂ
40 PRINT "B, H";
50 INPUT B, H
60 IZ=(B * H ↑ 3)/12
70 IY=(B ↑ 3 * H)/12
80 PRINT "IZ="; IZ, "IY="; IY
90 END
RUN
B, H? 20, 60
IZ= 360000 IY= 40000
  
```

TEST

Scrieți un program BASIC-PRAE ca-
re să afișeze:
RUN

```

GENERATOAREA CILINDRULUI = ? 8
RAZA CILINDRULUI = ? 2
ARIA LATERALĂ =
VOLUMUL =
  
```

```

10 PRINT "GENERATOAREA
    CILINDRULUI = ";
20 INPUT G
30 PRINT "RAZA CILINDRULUI = ";
40 INPUT R
50 AL=2 * PI * R * G
60 AT=AL+2 * PI * R ↑ 2
70 V=PI * R ↑ 2 * G
80 PRINT "ARIA LATERALĂ="; AL
90 PRINT "ARIA TOTALĂ="; AT
100 PRINT "VOLUMUL="; V
110 END
  
```

□ Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 202

Scrierea programului. Reguli generale

Limbajul BASIC HC-85, TIM S, SPECTRUM dispune de următoarele reguli generale de scriere:

- fiecare linie program începe cu un număr de linie;
- numărul de linie este un număr întreg format din 1 până la 4 cifre, putând avea valoarea cuprinsă între 1 și 9999;
- o linie program poate conține una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- numele variabilelor numerice simple sînt formate dintr-un număr nelimitat de caractere (litere, cifre, blanc) care încep cu o literă. Litera mare și litera mică omoloagă sînt interpretate la fel;
- variabilele de tip șir de caractere sînt formate dintr-o literă urmată de "\$";
- variabilele utilizate în instrucțiunea **FOR** sînt formate dintr-o singură literă;
- numele variabilelor indexate sînt formate dintr-o literă;
- operatorii aritmetici admiși în limbajul BASIC HC-85, TIM S, SPECTRUM sînt: +; -; *; /; ↑;
- operatorii de relație admiși sînt: < ; > ; <= ; >= ; < > ;
- operatorii logici admiși sînt: **NOT, AND, OR**;
- funcțiile aritmetice admise sînt: **SIN, COS, TAN, ATN, LN, EXP, SQR, ABS, INT, SGN, ASN, ACS**;
- funcțiile grafice admise sînt: **PLOT, DRAW, CIRCLE, POINT**;
- instrucțiunile admise pentru utilizarea culorilor sînt: **PAPER, INK, FLASH, INVERSE, OVER, BORDER, ATTR**;
- funcțiile care operează cu șiruri de caractere sînt: **LEN, STR\$, VAL, RIGHT\$, MID\$, CHR\$, INKEY\$**;
- alte instrucțiuni și funcții disponibile sînt: **USR, BIN, RANDOMIZE, RND, DEF**;
- funcțiile de intrare admise sînt: **INKEY\$, PEEK**;
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF ... THEN**;
- instrucțiunea de ciclare permisă este **FOR ... NEXT**;
- instrucțiunile de intrare/ieșire permise sînt: **INPUT, INPUTLINE, PRINT, LPRINT, READ, DATA, RESTORE, IN, OUT, POKE**;
- instrucțiunea admisă pentru generarea sunetelor este **BEEP**.

Vom aplica aceste reguli la scrierea programului.

```

10 REM PROGRAMUL CITEȘTE RAZA           40 LET A=4 * PI * R ↑ 2
(R) A UNUI REZERVOR SFERIC,           50 LET V=4 * PI * R ↑ 3/3
CALCULEAZĂ ARIA (A), VOLUMUL (V)     60 PRINT "R="; R; "A="; A; "V=";
ȘI TIPĂREȘTE: RAZA, ARIA, VOLUMUL V
20 PRINT "RĂZA=";                     70 STOP
30 INPUT R

```

Remarcați modul de scriere a comentariului din linia 10 – pe patru linii (fizice) ecran. Pentru tipărirea mesajului "RAZA=" am folosit instrucțiunea **PRINT** cu separatorul punct și virgulă (vezi linia 20). Instrucțiunea **INPUT** (din linia 30) citește valoarea razei (în variabila R) în timpul execuției programului.

Observație. Instrucțiunile de atribuire din liniile 40 și 50 folosite pentru calculul ariei și volumului rezervorului cer folosirea obligatorie a cuvintului **LET**.

Pentru afișarea rezultatului final s-a scris instrucțiunea **PRINT** din linia 60 care utilizează pentru tabulare separatorul " ; ". În sfârșit, instrucțiunea **STOP** (vezi linia 70) oprește execuția programului.

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii și corectării interactive a programelor BASIC HC-85, TIM S, SPECTRUM sînt următoarele:

- linia program poate fi tastată numai după apariția literei de cursor;
- litera de cursor (care arată unde va fi inserat următorul caracter tastat) poate fi: **K, L, C, E, G**. Cele cinci litere cursor se identifică cu modurile de lucru ale calculatorului;
- literele de cursor **K, L, C, E** sau **G** pot fi obținute prin acționarea tastelor [**CAPS SHIFT**], [**SYMBOL SHIFT**];
- modul **K** se formează automat la începutul liniei după caracterul " : " sau după **THEN**;
- în modul **L** (litere mici) se afișează simbolul principal de pe tastă iar literele se editează cu minuscule. [**CAPS SHIFT**] cu o tastă literă editează litera mare în modul **L**;
- în modul **C** (litere mari), literele se afișează cu majuscule. [**CAPS LOCK**] realizează trecerea **L** la **C** și **C** la **L**;
- cursorul **E** apare prin tastarea simultană a [**CAPS SHIFT**] și [**SYMBOL SHIFT**];
- modul **E** (extindere) este utilizat pentru realizarea semnificației scrise deasupra tastei cu albastru (TIM S) prin apăsarea tastei sau pentru realizarea semnificației scrise dedesubtul tastei cu roșu (TIM S) prin tastarea simultană a tastei alese și [**SYMBOL SHIFT**];
- în modul **G** (tastați 9 și [**CAPS SHIFT**]) – semigrafic și grafic tastarea cifrelor va edita caracterele semigrafice înscrise pe taste, iar tastarea literelor de la **A** la **T** va edita caractere grafice definite de utilizator.
- liniile se editează pe liniile inferioare ale ecranului alfanumeric;
- funcțiile de editare pentru un calculator cuplat cu un televizor alb/negru sînt realizate de tastele numerice apăsate simultan cu [**CAPS SHIFT**];
- ștergerea ultimului caracter introdus se realizează prin acționarea tastei [**DELETE**];
- comenzile **TRUE VIDEO** și **INV VIDEO** editează caracterul ce urmează a fi tastat în pozitiv și negativ;
- comenzile **>**, **<** deplasează cursorul în sensul dorit fără a șterge textul;
- pentru ca linia să fie prelucrată, se va acționa tasta [**ENTER**];
- linia de program (precedată de un număr de linie) va fi plasată în textul editat în partea superioară a ecranului;
- dacă linia este incorectă, în linia de editare va apare semnul întrebării;
- ultima linie editată are plasat între numărul de linie și restul liniei simbolul "**<**" numit și cursor de linie editată;
- pentru modificarea unei linii de program se deplasează cursorul linie la linia respectivă (cu ajutorul "**^**", "**V**"), după care se tastează **EDIT**. Linia va fi adusă în spațiul de editare unde se modifică tastînd **<**, **>** și [**DELETE**]. În final, se tastează [**ENTER**];
- comanda **NEW** inițializează sistemul BASIC ștergînd programul sau variabilele curente;
- comenzile **LIST** și **LIST** (nr. linie) afișează pe ecran textul programului BASIC începînd cu cel mai mic număr de linie, respectiv (nr. linie). La umplerea ecranului (instrucțiunea listează 22 linii începînd din partea de sus a ecranului și (nr. linie) linie curentă), în partea de jos a acestuia se afișează "scroll?" (dacă se tastează "**N**" se oprește afișarea);
- într-o instrucțiune de atribuire, cuvîntul **LET** este obligatoriu;
- o variabilă simplă și o variabilă tablou pot avea același nume;
- listarea la imprimantă se realizează cu instrucțiunile **LLIST** și **LPRINT** care au aceleași funcții ca **LIST** și **PRINT**. Nu va mai apare mesajul "scroll".

Salvarea unui program BASIC pe casetă

Pentru păstrarea programelor BASIC HC-85, TIM S, SPECTRUM avem nevoie de un casetofon audio uzual și de casete audio obișnuite (casetele audio de 60 minute asigură un spațiu de înregistrare de circa 900 K). Procedura de salvare [45] a unui program este următoarea:

1) se tastează **SAVE** care are formatul:

Format general
SAVE "(nume)"

unde, (nume) reprezintă numele programului ales de utilizator, format din maximum 10 caractere;

2) se poziționează banda în punctul unde se dorește înregistrarea, se pornește casetofonul și se apasă oricare tastă (calculatorul răspunde cu mesajul: "START TAPE THEN PRESS ANY KEY"). În timpul înregistrării se aude un sunet specific în difuzorul casetofonului. Corectitudinea salvării este indicată de apariția mesajului O.K.

REMARCI

- Pe o casetă audio obișnuită se pot salva patru tipuri de informații: 1) programul BASIC împreună cu variabilele sale; 2) tablouri numerice; 3) tablouri de caractere; 4) șiruri de biți (program în limbaj de asamblare sau imagini binare). Formatele generale ale comenzii **SAVE** (cu semnificația respectivă) sint:

Format general	Semnificație
SAVE "(nume)"	Salvează programul BASIC și variabilele lui în fișierul (nume).
SAVE "(nume)" LINE (n)	Idem, cu deosebirea că, la încărcare, programul intră automat în execuție de la linia (n).
SAVE "(nume)" CODE "(start) ","(lungime)"	Memorează șirul de biți de la adresa (start) pe (lungime) octeți în fișierul (nume).
SAVE "(nume)" DATA (litera())	Memorează tablouri numerice (litera) în fișierul (nume).
SAVE "(nume)" DATA (litera\$())	Memorează "șir" (litera \$) în fișierul (nume)
SAVE "(nume)" SCREEN\$	Salvează imaginea binară de pe ecran în fișierul (nume).

- Pentru verificarea informațiilor de pe casetă cu informațiile aflate deja în memorie, se utilizează comanda **VERIFY**.
- În timpul căutării programului specificat, calculatorul afișează numele tuturor programelor întâlnite.
- **VERIFY** are același format și aceeași semnificație ca **LOAD** numai că realizează doar o comparație cu programul din memorie. Dacă în urma comparării informațiile sint identice se afișează mesajul "O.K.", iar în caz de neconcordanță se listează mesajul "R TAPE LOADING ERROR".

Aplicație. Salvați programul pe o casetă audio sub numele: EX3PCHCTS. Verificați informațiile de pe casetă cu informațiile aflate deja în memorie.

Citirea de pe casetă a unui program BASIC. Citirea unui program BASIC HC-85, TIM S, SPECTRUM înregistrat pe o casetă se face cu comanda **LOAD** al cărei format general este:

Format general
LOAD "<nume>"

De reținut că această comandă șterge vechiul program aflat în calculator împreună cu variabilele sale înainte de a încărca unul nou.

Formatele generale ale comenzii **LOAD** (cu semnificația respectivă) [45] sint:

Format general	Semnificație
LOAD ""	Încarcă de pe casetă primul fișier întilnit.
LOAD "<nume>" DATA <litera()>	Șterge orice tablou cu numele <litera()> din programul rezident în memoria calculatorului și încarcă în loc tot ce se găsește în fișierul <nume> de pe casetă.
LOAD "<nume>" DATA <litera\$()>	Șterge orice tablou cu numele <litera\$()> din programul rezident în memoria calculatorului și încarcă în loc tot ce se găsește în fișierul <nume> de pe casetă.
LOAD "<nume>"CODE LOAD "<nume>"CODE"<start>" LOAD "<nume>"CODE"<start>"<lungime>"	Încarcă un șir de biți din fișierul <nume> în memoria calculatorului începînd de la adresa <start> (pe un număr de octeți <lungime>) scriind peste ceea ce se găsește în memorie.
LOAD "<nume>" SCREEN\$	Încarcă fișierul <nume> în zona de memorie alocată imaginii ecran. <i>Observație.</i> Comanda este sinonimă cu LOAD "<nume>" CODE 16384, 6912.

Remarci

- La memorarea imaginii video nu poate fi folosită comanda **VERIFY**.
- Comanda **MERGE** încarcă un program înregistrat pe casetă în memoria calcu-

Format general
MERGE "<nume>"

lulatorului (ca și **LOAD**), dar, spre deosebire de comanda **LOAD**, ea șterge din liniile și variabilele vechiului program aflat în memorie doar pe cele care se suprapun ca număr sau ca nume cu cele din programul aflat în fișierul "<nume>". De reținut că fișierul "<nume>" conține numai programe scrise în limbajul BASIC. (n trebuie să fie cuprins între 0 și 65535). Pentru **PAUSE 0** calculatorul oprește definitiv execuția programului, iar 65535 corespunde la aproximativ 1320 secunde. Pentru continuarea la oprirea cu **PAUSE 0** se acționează orice tastă.

Aplicație. Citiți de pe casetă programul EX3PCHCTS salvat anterior.

Execuția programului. Reguli generale

Dacă programul nu se afiă în memoria calculatorului, vă invităm și de această dată să-l încărcăm de pe casetă (vezi aplicația de la pag. 91 pe care trebuia s-o realizați singuri!) și apoi să-l executați. Să urmărim împreună ce se petrece pe ecranul monitorului:

```

RAZA=?3
R=3 A=113.09734 V=113.09734

9 STOP statement,70:1

```

După ce linia 10 a fost ignorată de calculator, ca urmare a execuției instrucțiunii din linia 20 s-a tipărit mesajul: "RAZA=" și imediat după aceea, în continuarea mesajului, și-a făcut apariția o mai veche cunoștință de-a noastră – semnul întrebării. Instrucțiunea **INPUT** din linia 30 așteaptă să introduceți o valoare pentru variabila R. Tastați 3 și acționați **[ENTER]**. Calculatorul afișează rezultatul și se oprește din execuție cu eroare 9, datorită instrucțiunii **STOP** (linia 70). Programul se poate continua (în cazul nostru nu este nevoie deoarece **STOP** este ultima instrucțiune din program) tastând de două ori **CONTINUE**.

Pentru lansarea în execuție a unui program BASIC HC-85, TIM S, SPECTRUM trebuie cunoscute și respectate următoarele **reguli** generale:

- comanda **RUN** realizează lansarea în execuție a programului curent existent în memoria calculatorului, de la instrucțiunea cu cel mai mic număr de linie;
- comanda **RUN** (nr. linie) lansează în execuție programul curent existent în memoria calculatorului de la instrucțiunea care are numărul de linie egal cu (nr. linie);
- comanda **GOTO** al cărei format general este:

Format general

GOTO (nr. linie)

provoacă execuția programului de la linia (nr. linie) fără a șterge nimic din memorie;

- comanda **RUN** poate fi lansată și dintr-un program dacă este precedată de un număr de linie;
- întreruperea execuției unui program se realizează acționind tasta **[BREAK] (CAPS SHIFT și SPACE)**. Calculatorul răspunde cu mesajul: "L BREAK INTO PROGRAM". Apăsarea tastei **[CONTINUE]** reia execuția programului cu următoarea instrucțiune BASIC;
- lista mesajelor de eroare BASIC se găsește în vol. 2, pag. 5.
- instrucțiunea **STOP** are același efect ca și tasta **[BREAK]**. Execuția se poate relua tastind **CONT (CONTINUE)**;
- după introducerea datelor prin instrucțiunea **INPUT** se tastează **[ENTER]**;
- **PAUSE** al cărei format general este:

Format general
PAUSE n

oprește execuția programului și așteaptă un număr de n/50 secunde.

TESTE

- a) Scrieți un program BASIC HC-85, TIM S, SPECTRUM care să afișeze:

```
RUN
RAZA, ÎNĂLȚIMEA CONULUI = ? 3, 8
VOLUMUL CONULUI =
```

- b) Pentru calculul expresiei:

$$E = \frac{n^2(n+1)^2}{4} - \frac{n(n+1)(2n+1)}{6} - \frac{n(n+1)}{2}$$

s-a scris un program BASIC în trei variante (b₁, b₂, b₃). Precizați care dintre versiuni este corectă.

- b1) 10 PRINT "N=";
20 INPUT N
30 LET U=N*(N+1)/2
40 LET E=U*U-U*(2*N+1)/3-U
50 PRINT "E="; E
60 END
- b2) 10 PRINT "N=";
20 INPUT N
30 LET E=(N*(N+1)/2)*
(N*(N+1)/2 - (2*N+1)/3-1)
40 PRINT "E="; E
50 END
- b3) 10 PRINT "N=";
20 INPUT N

```
30 LET U=N*(N+1)/2
40 LET E=U↑2-U*(2*N+1)/3-U
50 PRINT "E="; E
60 END
```

- R. Toate trei.

- c) Precizați rezultatul execuției următorului program BASIC HC-85, TIM S, SPECTRUM, dacă lui A și B li se atribuie dinamic valorile 820, respectiv 105.

```
5 REM ** INPUT **
10 INPUT A
20 PRINT , A
30 INPUT B
40 PRINT , B
50 LET C=A+B
60 PRINT , "----"
70 PRINT , C
80 STOP
```

- R. RUN
? 820 820
? 105 105
 925
- 9 STOP statement , 80 : 1

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 202

Scrierea programului. Reguli generale

Pentru scrierea unui program în limbajul BASIC-AMSTRAD (6128) trebuie respectate următoarele **reguli** generale:

- fiecare linie program începe cu un număr de linie;
- o linie program poate să conțină una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- o variabilă este identificată printr-un șir de caractere care începe cu o literă, urmată de un număr oarecare de litere sau cifre;
- variabilele admise sînt de următoarele tipuri: numerice reale, numerice întregi, șiruri de caractere;
- tipul unei variabile se poate stabili și prin caracterele speciale: "!" (pentru variabilele numerice reale); "%0" (pentru variabilele numerice întregi); "\$" (pentru șiruri de caractere) ce se atașează numelui variabilei;
- o variabilă poate fi declarată și implicit cu ajutorul instrucțiunilor admise: **DEFINT, DEFREAL, DEFSTR**;
- dacă precizarea tipurilor implicite nu este făcută de utilizator, toate variabilele se consideră numerice reale;
- numele unei variabile indexate este orice nume de variabilă acceptat de către BASIC-AMSTRAD (6128);
- valoarea minimă a indicelui unui tablou este zero;
- operatorii aritmetici admiși sînt: "+" (pentru adunare); "-" (pentru scădere); "*" (pentru înmulțire); "/" (pentru împărțire); "↑" (pentru ridicare la putere);
- operatorii de relație admiși sînt: "="; "<"; ">"; "<="; ">="; "<>";
- operatorii logici admiși sînt: **AND, OR, XOR, NOT**;
- funcțiile matematice admise sînt: **SIN, COS, TAN, ATN, PI, SQR, EXP, LOG, LOG10, INT, ABS, SGN, FIX, CREAL, CINT, ROUND, RND, RANDOMIZE**;
- funcțiile admise pentru șirurile de caractere sînt: **LEN, POS, VPOS, INSTR, MID\$, LEFT\$, RIGHT\$, STRING\$, SPACE\$, LOWER\$, UPPER\$, COPYCHR\$**;
- funcțiile de conversie admise sînt: **CHR\$, VAL, STR\$, ASC, HEX\$**;
- alte funcții (auxiliare) disponibile sînt: **TAB, ERR, ERROR, DERR, SPC, ZONE, EOF, TIME, MAX, MIN, MOD, MODE**;
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF... THEN... ELSE, IF... GOTO, ON ERROR GOTO, ON GOTO**;
- instrucțiunile de ciclare permise sînt: **FOR... NEXT, WHILE... WEND**;
- instrucțiunile de intrare/ieșire permise sînt: **READ, DATA, RESTORE, INPUT, LINE INPUT, PRINT, PRINT SPC, PRINT TAB, PRINT USING, POKE, PEEK**;
- instrucțiunile admise pentru operațiile de intrare/ieșire pe disc sînt: **OPENIN, OPENOUT, CLOSEIN, CLOSEOUT, INPUT, LINE INPUT, PRINT, PRINT USING, WRITE, SAVE, LOAD, RUN**;
- instrucțiunile grafice admise sînt: **LOCATE, WINDOW, MOVE, MOVER, ORIGIN, XPOS, YPOS, GRAPHICS PEN, GRAPHICS PAPER, INK, MASK, TAG, TAGOFF, DRAW, DRAWR, CLG, CURSOR, PLOT**;
- instrucțiunile admise pentru generarea sunetelor sînt: **SOUND, ENT, ENV, ON SQ GOSUB**.

Vom aplica aceste reguli la scrierea programului.

Variabile cu același nume dar de tipuri diferite

După cum constatați, prima linie (10) am „sacrificat-o” pentru documentarea programului.

Observație. În BASIC-AMSTRAD, **REM** poate fi înlocuit printr-un apostrof.

Pentru tipărirea pe ecran a mesajului "RAZA=" am scris instrucțiunea 20. Instrucțiunea **INPUT** din linia 30 citește (dinamic) valoarea razei (R). Instrucțiunea din linia 40... De ce v-ați oprit? $40 \ r^0_0=r$. Nu sînteți obișnuiți cu variabila (r^0_0) din stînga semnului egal al instrucțiunii de atribuire?

Amintiți-vă că în BASIC-AMSTRAD tipul unei variabile se poate stabili și prin caracterele speciale: "%0"; "!"; "\$" ce se atașează numelui

variabilei. Variabila $r^0/0$ este o variabilă numerică de tip întreg. Rețineți așadar, că $r^0/0$ și r sînt variabile distincte, chiar dacă numele lor este același.

Instrucțiunea ZONE

Pentru afișarea unui rînd de spații am scris instrucțiunile 50, 100, 120. Instrucțiunile 55 și 60 servesc la afișarea unui mic cap de tabel în pozițiile 1 și 21 ale ecranului.

55 ZONE 20

60 PRINT "RAZA (Nr. real)", "RAZA (Nr. întreg)"

Instrucțiunea **PRINT** din linia 60 afișează cele două șiruri de caractere în pozițiile 1 și 14 (tabulare standard pentru virgulă)! Ce ne facem? Tot BASIC-ul AMSTRAD "ne rezolvă", întrucît dispune de o instrucțiune specială – **ZONE** – care permite modificarea tabulării standard definite de separatorul virgulă într-o instrucțiune **PRINT**, cu un număr ce ia valori cuprinse între 1 și 255.

Formatul general al instrucțiunii este:

Format general

<nr. linie>**ZONE**<nr. întreg>

unde <nr. întreg> ia valorile precizate mai sus.

Efectul instrucțiunii **ZONE** (linia 55) se răsfrînge și asupra instrucțiunii **PRINT** din linia 70 care afișează valorile propriu-zise ale lui r și $r^0/0$ în coloanele 1 și 21.

Pentru calculul ariei și volumului rezervorului sferic am scris instrucțiunile de atribuire 80 și 90 în care variabila r este reală.

Rezultatul final este tipărit cu instrucțiunea 110 după un rînd de spații editat în linia 100.

Instrucțiunile **STOP** și **END** termină execuția programului redînd controlul sistemului.

PRINT SPC, PRINT TAB

Este momentul să vă prezentăm încă două opțiuni ale instrucțiunii **PRINT** și anume: **SPC** și **TAB**, pe care le vom utiliza foarte des în cadrul lucrării. **SPC** pregătește un număr întreg de spații libere indicat înainte de a imprima sau afișa articolul indicat, cu condiția ca acesta din urmă să se mențină pe linia de imprimat.

Formatul general este:

Format general

<nr. linie>**PRINT**[# <nr. canal>],[<listă>] [;][**SPC**(<nr. întreg>)][<listă>]

TAB realizează saltul la coloana al cărei număr este egal cu valoarea indicată.

Formatul general este:

Format general

(nr. linie) **PRINT** [# (nr. canal)][(listă)][;][**TAB**((nr. intreg))][(listă)]

Aplicație. Să se modifice instrucțiunea **PRINT** din liniile 60 și 70 ale programului exemplu, utilizând opțiunile **SPC** și **TAB**. Noile instrucțiuni vor prelua funcțiunea instrucțiunii **ZONE** din linia 55 a programului.

60 **print** "RAZA (Nr. real)"; **spc** (7);
"RAZA (Nr. intreg)"

70 **print** r; **spc** (15); r%/

și

60 **print** "RAZA (Nr. real)"; **tab** (21);
"RAZA (Nr. intreg)"

70 **print** r; **tab** (21); r%/

TEST

Simulați cu creionul și hîrtia rezultatul execuției următoarelor programe:

a) 5 **cls**

10 a=8.25 : b=7.8 : a%/ = a

20 **PRINT** a, b

30 **PRINT** a, a%/, b

40 **ZONE** 8

50 **PRINT** a, b, a%/,

60 **PRINT** "A", a, "A%/", a%/, "B", b

70 **PRINT** **spc** (0) "A"

80 **PRINT** **spc** (0); a%/,

90 **PRINT** **tab** (30); a; **spc** (5); "a"

b) 10 **for** i=0 to 6

20 **print** **spc** (i); "a"

30 **next** i

c) 5 a=1

10 **for** i=0 to 6

20 **print** **spc** (i); i

30 **next** i

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii și corectării interactive a programelor BASIC-AMSTRAD sînt următoarele:

- linia program poate fi tastată numai după apariția cursorului "█";
- pentru introducerea unui cuvînt cheie BASIC în calculator, puteți utiliza indiferent MAJUSCULE sau minuscule;
- se preferă ca tastarea instrucțiunilor BASIC și a programelor să se realizeze cu minuscule, deoarece la listarea programului (cu litere mari) cuvintele cheie prost ortografiate nu sînt convertite în majuscule;
- cînd se ajunge în marginea dreaptă a ecranului (s-au tastat mai mult de 40 caractere) caracterul următor trece automat în linia următoare;
- comanda **NEW** inițializează sistemul BASIC ștergînd programul sau variabilele curente;
- punerea la zero a sistemului se realizează acționînd simultan tastele [**CONTROL**], [**SHIFT**], [**ESC**];
- comanda **LIST** cu formatul

Format general

LIST[(ansamblu de linii)][.(număr canal)]

listează programul pe canalul dorit (#0 este ecranul, #8 este imprimanta) cu majuscule;

- Observație.** Puteți omite primul sau ultimul număr de linie al parametrului (ansamblu de linii) în vederea listării programului de la început sau până la sfârșit;
- listarea programului poate fi întreruptă prin acționarea o singură dată a tastei [ESC]. Pentru continuare, apăsați bara de spații. Dacă apăsați de două ori pe [ESC], listarea programului încetează și se revine în modul direct;
 - pentru ca linia să fie prelucrată se va acționa tasta [RETURN]. În condiții normale de funcționare a calculatorului, tasta [ENTER] are același efect ca și tasta [RETURN], funcțiunile tastei [ENTER] putând fi însă redefinite;
 - pentru ștergerea unui caracter din stînga cursorului tastați [DEL];
 - pentru tastarea majusculelor și a simbolurilor superioare înscrise pe taste mențineți apăsată tasta [SHIFT];
 - pentru tastarea majusculelor și a simbolurilor superioare înscrise pe taste puteți acționa, de asemenea, tastele [CONTROL] [CAPS LOCK] în această ordine. În acest caz, nu mai este nevoie să mențineți tot timpul apăsată tasta [SHIFT];
 - pentru ștergerea unui caracter care se găsește sub cursor tastați [CLR];
 - comanda **AUTO** al cărei format general este:

Format general

AUTO{(nr. linie)},{(increment)}

generează automat numerele de linie;

- parametrul facultativ (nr. linie) fixează primul număr de linie care se va genera. În absența acestuia se ia valoarea 10. Parametrul (increment), de asemenea facultativ, fixează pasul de secvențare a liniilor. Dacă nu se precizează, el ia valoarea 10. Pentru oprirea numerotării automate a liniilor acționați tasta [RETURN];
- dacă una din liniile programului a fost tastată incorect (vezi mesajul Syntax error sau alt mesaj de eroare) este posibilă editarea (modificarea) liniei fără a fi nevoiți să o retastăm;
- există trei metode pentru editarea unui program: 1) **retastarea integrală a liniei**; 2) **editarea cu ajutorul cursorului**; 3) **COPY cursor sau copie cu ajutorul cursorului**.

Metoda de editare cu ajutorul cursorului (2) constă în poziționarea cursorului pe caracterul dorit, ștergerea acestuia și eventual înlocuirea lui cu un alt caracter. Procedura este următoarea: 1) tastați **edit** (nr. linie); 2) poziționați cursorul pe caracterul eronat; 3) tastați [CLR], introduceți noul caracter; 4) tastați [RETURN]; 5) tastați **list** [RETURN] pentru a verifica dacă linia (nr. linie) este corectă.

Aplicație. Corecțai cu metoda prezentată mai sus linia 10 a următorului program:

```
5 a = 3
10 print a
20 end
```

Pentru a corecta linia 10 (l în r), tastați

```
edit 10 [RETURN]
```

Linia 10 apare sub linia 20, iar cursorul se poziționează pe litera p (primul caracter din print). Pentru a elimina pe l din print, acționați tasta [→] până ce cursorul se poziționează pe l. Tastați [CLR]. În acest moment a fost șters caracterul l. Tastați în locul lui l pe r și acționați tasta [RETURN]. Dați comanda **list** [RETURN] și veți obține:

```
5 a = 3
10 PRINT a
20 END
Ready
```



Observație. Dacă executați micuțul program de mai sus, veți obține pe ecranul monitorului (vezi fig. 3.5) – cum de altfel era și de așteptat – un mesaj de eroare și

imediat sub el linia 10 eronată. În acest caz aplicați metoda de editare cu ajutorul cursorului de la pasul 3.

```
run
Syntax error in 10
10 p|int a
```

Fig. 3.5.

Metoda COPY cursor (3) constă în copierea cu cel de-al doilea cursor **[SHIFT]**, și una din tastele: **[↑]**, **[↓]**, **[→]**, **[←]** a liniei de program pe care dorim să o modificăm. Procedura este următoarea: 1) mențineți apăsată tasta **[SHIFT]** și acționați tasta cursorului **[↑]** până în momentul poziționării cursorului la începutul liniei de modificat (cursorul principal rămâne imobilizat); 2) mențineți apăsată tasta **[COPY]** până în momentul în care cursorul s-a plasat în poziția dorită (linia pe care o corecțăm este rescrisă în același timp după ultima instrucțiune din program, iar cursorul principal se imobilizează în același loc ca și cursorul de copiere); 3) tastați caracterul pe care doriți să-l introduceți; el apare numai în linia copiată (cursorul principal s-a deplasat în timp ce cursorul de copiere rămâne imobilizat); 4) mențineți apăsată tasta **[COPY]** pentru copierea integrală a liniei asupra căreia s-au operat modificări. Tastați **[RETURN]** (cursorul de copiere dispare, iar cursorul principal se plasează sub linia copiată).

Aplicație. Corecțăm cu metoda **COPY CURSOR** linia 10 a programului din aplicația precedentă (pag. 98).

Pentru a corecta linia 10 (l în r) mențineți apăsată tasta **[SHIFT]** și acționați tasta cursorului **[↑]** până cînd se suprapune cu cifra 1 din linia 10. În fig. 3.6 (a) remarcați și poziția cursorului principal – aceeași!

```
5 a=3
10 p|1|nt a
20 end
■
```

a)

```
Ready
5 a=3
10 p| |nt a
20 end
10 p■
```

b)

```
Ready  
5 a=3  
10 print a  
20 end  
10 pr■
```

c)

```
Ready  
5 a=3  
10 print a  
20 end  
10 pr■
```

d)

```
Ready  
5 a=3  
10 print a■  
20 end  
10 print a■
```

e)

```
Ready  
5 a=3  
10 print a  
20 end  
10 print a  
■
```

f)

```
list  
5 a=3  
10 PRINT a  
20 END  
Ready  
■
```

g)

Fig. 3.6 a-g.

În continuare mențineți apăsată tasta **[SHIFT]** și tastați **COPY** pînă cînd cursorul s-a suprapus peste litera **l**. În același timp, linia **10** este rescrisă la baza programului iar cursorul principal se imobilizează în același loc ca și cursorul de copiere (vezi fig. 3.6, b).

Tastați **r** care apare numai pe linia **10** de la baza programului (vezi fig. 3.6, c). Remarcați cum cursorul principal s-a deplasat, în timp ce cursorul de copiere a rămas pe loc. Acum apăsați pe **[SHIFT]** și deplasați cursorul de copiere pînă se suprapune peste litera **i** (vezi fig. 3.6, d).

În sfîrșit, mențineți apăsată tasta **[SHIFT]** și apăsați pe **COPY** în vederea copierii integrale a liniei **10** (vezi fig. 3.6, e).

Tastați **[RETURN]** astfel încît noua linie **10** (care o elimină pe precedenta linie **10**) să fie introdusă în memoria calculatorului.

Observați cum cursorul de copiere dispăre, iar cursorul principal se plasează sub linia **10** (vezi fig. 3.6 (f)).

Puteți cere listarea programului tastînd:

list [RETURN]

Imaginea noului program este prezentată în fig. 3.6 (g).

Salvarea unui program BASIC-AMSTRAD pe dischetă*

Pentru început, trebuie să facem precizarea că AMSTRAD 6128 utilizează dischete de 3 inch (vezi fig. 3.7).

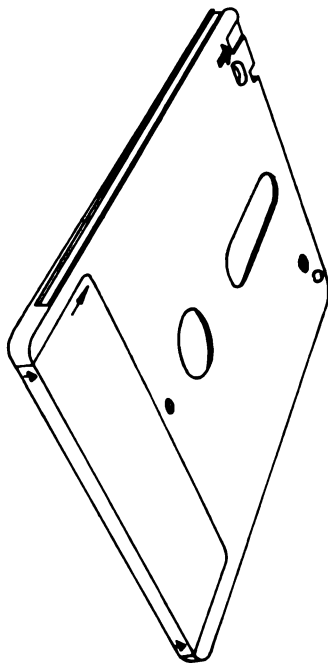


Fig. 3.7.

Fiecare față a dischetei poate fi utilizată separat: colțul din stînga al fiecărei fețe ce aparține unei dischete poartă o săgeată ce indică o

* Se pot utiliza, de asemenea, și casetele magnetice.

gaură acoperită de un obturator. Aceasta este o gaură de protecție la scriere, care va împiedica să se scrie peste informațiile înregistrate. Când gaura este închisă, calculatorul poate scrie pe dischetă. Când este deschisă, nu poate fi scris nimic. Pentru închiderea și deschiderea găurii de protecție procedați ca în fig. 3.8.

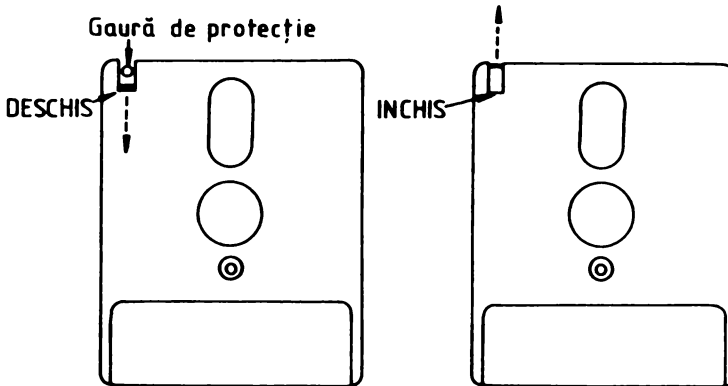


Fig. 3.8.

Observație. Procedura este valabilă pentru dischetele CF-2 (Compact Floppy) AMSOFT.

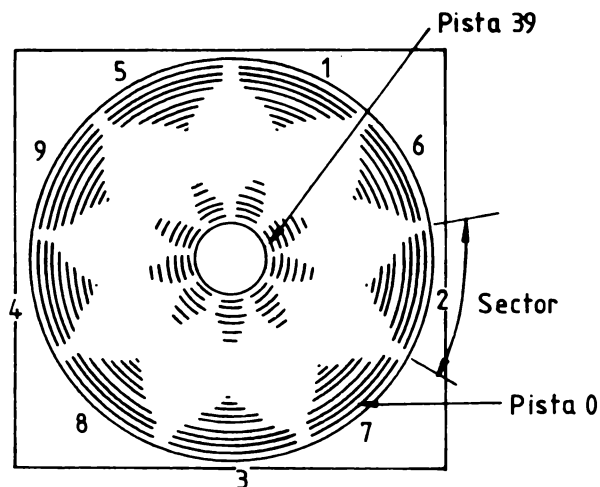
Deși dvs. v-ați deprins cu practica salvării programelor BASIC pe casetă, va trebui ca, în cazul salvării programelor pe dischetă, să vă însușiți câteva noțiuni suplimentare. Sînt două diferențe esențiale. Nu puteți utiliza niciodată pentru înregistrare o dischetă neinițializată (virgină) cum procedați în cazul casetelor. Orice dischetă trebuie să fie mai întii formatată. Va trebui ca numele pe care le dați fișierelor de pe dischetă să fie corecte. Dacă numele fișierelor de pe casetă urmau reguli nu întotdeauna precise, cele de pe dischetă trebuie, dimpotrivă, să se conformeze strict regulilor CP/M.

Formatarea dischetelor. Înainte de a scrie pe o dischetă neinițializată, trebuie să începem prin a o formata. Formatara poate fi înțeleasă ca și construirea unui set de etajere prevăzute cu compartimente pentru stocarea informației sau, altfel spus, o structură capabilă de a primi și totodată de a restitui informații.

Operația de formatare împarte fiecare față a dischetei în 360 de zone distincte.

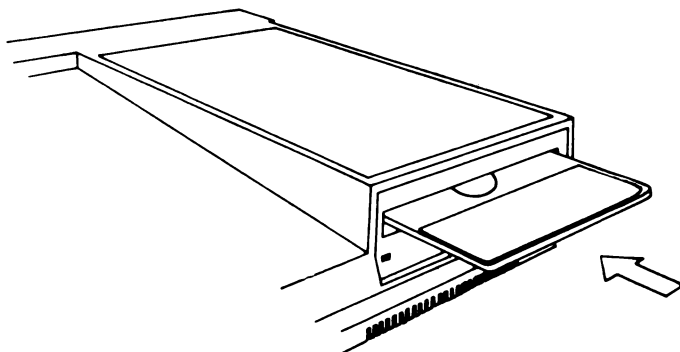
Suprafața unei dischete este împărțită în 40 de piste concentrice (0-39), fiecare pistă fiind împărțită în 9 sectoare.

Un sector, indiferent de pistă, poate conține 512 octeți sau 1'0 Ko spațiu disponibil pe fiecare față a dischetei.



Cum folosim dischetele CP/M? Comenzi CP/M. Înainte de a utiliza o dischetă neinițializată în vederea scrierii și citirii programelor BASIC, formatați-o cu ajutorul feței 1 a unei dischete pe care se află înregistrat sistemul de operare CP/M.

Treziți geniul adormit și introduceți fața 1 a dischetei CP/M în unitatea floppi:



Dacă sistemul dvs. are două unități floppi, introduceți discheta CP/M în unitatea principală (unitatea A).

Tastați:

| cpm [RETURN]

Observație. Bara verticală | se obține prin acționarea tastelor [SHIFT] și @.

După câteva secunde, pe ecran va apare mesajul:

CP/M plus AMSTRAD Consumer Electronics plc
care ne atenționează că sistemul este sub controlul CP/M plus.

Symbolul A> de pe ecran este echivalentul lui Ready sub BASIC. El ne avertizează că sistemul este pregătit să primească comenzile dvs.

Dacă tastați, de exemplu comanda BASIC:

cls [RETURN]

. . . calculatorul vă răspunde repetind cuvîntul neînțeles, urmat de un semn de întrebare:

CLS?

. . . demonstrîndu-ne că el nu înțelege comanda transmisă.

Să facem cunoștință acum cu cîteva comenzi CP/M. Tastați:

dir [RETURN]

Veți vedea cum apar repertoriul (catalogul) comenzilor și utilitarele sistemului de operare CP/M. Unul dintre ele se numește **DISCKIT 3**. Tastați:

DISCKIT 3

Timp de cîteva secunde, mesajul inițial **DISCKIT** se afișează în partea de sus a ecranului, urmat de:

One drive found

indicînd că utilitarul **DISCKIT** este în exploatare și că AMSTRAD-ul a recunoscut utilizarea sistemului cu o singură unitate floppy (cea care este integrată în calculator).

Dacă sistemul dvs. cuprinde încă o unitate suplimentară de floppy, mesajul afișat va fi:

Two drives found

La baza ecranului veți vedea:

Copy

Format

Verify

Exit from program

7	
4	
1	
0	

Este vorba de meniul principal al programului **DISCKIT**. Numerele înscrise în interiorul căsuțelor corespund tastelor de funcțiuni situate la dreapta claviaturii (**f0**, **f1**, **f4** și **f7**). Selecția o puteți face cu una din tastele precizate. Acționați tasta [**f0**] pentru a ieși din programul **DISCKIT**.

Deoarece noi dorim formatarea unei dischete, acționați tasta [**f4**].

Remarcă. Formatarea unei dischete deja înregistrate are ca efect distrugerea vechiului conținut.

Pe ecran va apare un nou meniu, propunându-vă să optați pentru unul din formatele: **System format**; **Data format**; **Vendor format**.

System format
Data format
Vendor format
Exit menu

	9
	6
	3

Acționind una din tastele de funcțiuni (**f3**, **f6** sau **f9**), puteți selecționa unul din formatele dorite. Pentru moment, alegeți formatul **Data**, acționind în consecință tasta [**f6**].

Pentru ieșirea din modul **Format** acționați [·] situată sub **f3**, **f6** și **f9**.

După selectarea tastei de funcțiuni **f6** (în cazul în care sistemul nu posedă decît o unitate floppi), pe ecran se va afișa mesajul:

[Y] Format as Data
Any other key to exit menu

În acest moment, scoateți discheta CP/M, introduceți discheta pe care doriți să o formatați și tastați [Y].

Formatarea dischetei se efectuează pistă cu pistă (de la pista 0 la pista 39), numărul pistei în curs de formatare afișându-se în colțul de sus din stînga ecranului.

Nu puteți formata o dischetă a cărei gaură de protecție la scriere este deschisă. Dacă doriți, totuși acest lucru, vi se va adresa mesajul:

Disc write-protected
Insert disc to format
R-etry or C-ancel

. . . Tastați atunci [C] pentru a anula tentativa de scriere, retrageți discheta și introduceți o alta formatată.

Observație. Niciodată să nu închideți obturatorul de protecție la scriere a unei dischete al cărei conținut doriți să-l conservați. Este cazul dischetei sistem CP/M. Odată terminată formatarea dischetei, calculatorul vă cere să scoateți discheta din unitatea floppi și de a apăsa pe o tastă oarecare a claviaturii. După aceea, puteți introduce o altă dischetă pe care doriți să o formatați și apăsați din nou pe [Y] pentru a reveni în meniul principal al utilitarului **DISCKIT**.

Pentru reinițializarea sistemului, tastați [**CONTROL**], [**SHIFT**] și [**ESC**] în această ordine.

Observație. Păstrați întotdeauna o copie a dischetei CP/M.

Formatarea dischetelor într-un calculator cu două unități floppi. Urmați instrucțiunile prezentate mai sus, selectați opțiunea **Format** din meniul principal al programului utilitar **DISCKIT**, acționind tasta de funcțiuni [**f4**] și apoi selectați **Data format** cu ajutorul tastei de funcțiuni [**f6**].

Veți vedea cum, pe ecran, se afișează al treilea meniu de opțiuni:

Format A:

Format B:

Exit menu

	8	
	5	
	2	

Optind pentru **Format B** (tasta [f5]) puteți lăsa discheta sistem în unitatea A (fața 1 către exterior), introducând deci discheta pe care doriți s-o formatați în unitatea B. Tastați așadar [Y] pentru formatare sau oricare altă tastă pentru a reveni în meniul principal al utilitarului **DISCKIT**.

Dacă ați optat pentru **Format A** (tasta [f8]), va trebui să scoateți discheta sistem din unitatea A pe care s-o înlocuiți cu discheta de formatat.

Observație. Nu uitați să protejați discheta sistem împotriva unei distrugerii accidentale a informațiilor pe care le conține.

Salvarea unui program pe dischetă se realizează cu comanda **SAVE** al cărei format general este:

SAVE(nume fișier)[,(tip fișier)][,(parametrii binari)]

unde, <nume fișier> se compune din două părți sau zone. Prima, obligatorie, poate să conțină pînă la 8 caractere, litere sau cifre (spațiul și semnele de punctuație sînt interzise). A doua zonă, facultativă, poate conține pînă la 3 caractere diferite de spațiu sau semne de punctuație. Cele două zone sînt separate prin punct. În absența celei de-a doua zone specificate, sistemul consideră automat **BAS** pentru fișiere BASIC sau **BIN** pentru fișiere binare (în cod mașină);

<tip fișier> reprezintă tipul fișierului ce se salvează. Poate avea una din valorile: a – fișier ASCII (fișiere de prelucrări de texte, programe de aplicații etc.); p – fișier protejat (este interzisă listarea sa după încărcarea în memorie sau întreruperea sa prin [ESC]); b – fișiere binare (stocarea blocurilor de date RAM pe dischetă);

<parametrii binari> reprezintă: adresa de început a zonei de memorie de salvat, lungimea sa (nr. octeți), punctul de intrare etc.

În tabelul 3.1 sînt prezentate mai multe exemple de comenzi **SAVE** comentate.

Tabelul 3.1

Comandă	Funcțiune
SAVE "prog.1"	salvează fișierul în mod BASIC neprotejat
SAVE "prog.1",P	salvează fișierul în mod BASIC protejat
SAVE "prog.1",A	salvează fișierul în mod ASCII
SAVE "prog.1",B	salvează fișierul în mod binar
SAVE "pag. ecran", b, 49152, 16384	salvează conținutul ecranului (vidaj ecran) de la adresa de început a memoriei ecran – 49152 pe lungimea de 16384 memorie ecran.

Listarea catalogului (repertoriului) unei dischete

După ce ați salvat un program pe dischetă, este bine să verificați dacă acesta există într-adevăr fizic. Pentru aceasta, este necesar să citiți numai **CAT**alogul dischetei respective. Tastați:

CAT [RETURN]

și veți vedea cum pe ecran se listează în ordine alfabetică numele tuturor fișierelor prezente, precum și lungimea programului (arondată la Ko superior). Totodată se afișează identificatorul dischetei și al utilizatorului, precum și numărul de octeți disponibili.

Aplicație. Salvați programul exemplu pe o dischetă (neinițializată) sub numele PC3AMST. Verificați dacă programul a fost salvat.

Încărcarea unui program de pe dischetă

Programele de pe dischetă sînt încărcate în memoria calculatorului cu ajutorul comenzii **load** al cărei format general este:

Format general \
LOAD (nume fișier)[,(adresă)]

Cu opțiunea de adresă se încarcă un fișier binar la adresa indicată în locul adresei în care el se găsea în momentul salvării. Pentru un demaraj rapid se folosește comanda **run** al cărei format general este:

Format general
RUN (nume fișier)

Observație. Comanda **run** este singura comandă care permite execuția programelor protejate.

În tabelul 3.2 sînt prezentate două exemple de încărcare a unui program de pe dischetă în memoria calculatorului.

Tabelul 3.2

Comandă	Funcțiune
load "prog.1" [RETURN] run [RETURN]	încarcă în memorie de pe dischetă (prog.1)
run "prog.1" [RETURN]	idem, dar încărcarea este mai rapidă

TEST

Indicați o procedură de copiere a unui program BASIC de pe o dischetă pe o altă dischetă.

Dacă dispuneți de un sistem cu două dischete, puteți introduce discheta cu programul de încărcat în unitatea B de exemplu, iar discheta care primește programul în unitatea A.

```
| b [RETURN]
load "prog.1" [RETURN]
| a [RETURN]
save "prog.1" [RETURN]
```

Aplicație. Încărcați PC3AMST în memoria calculatorului cu una din comenzile studiate (vezi aplicația de la pag. 107).

Remarci

- Nu uitați niciodată să scoateți discheta din unitatea de floppy, înainte de a pune calculatorul sub tensiune sau după ce l-ați oprit. Riscați pierderea programelor și a datelor înregistrate.
- Faceți întotdeauna o copie a dischetelor care conțin programe prețioase. Este cazul dischetei CP/M furnizată odată cu calculatorul AMSTRAD.
- Verificați întotdeauna sistemul de protecție la scriere a dischetei pe care se află sistemul de operare CP/M.
- Nu atingeți suprafața magnetică a dischetei.
- Nu extrageți niciodată discheta în timpul citirii sau scrierii.
- Amintiți-vă că formatarea unei dischete presupune ștergerea vechiului conținut.

Execuția programului.

Reguli generale

Pentru lansarea în execuție a unui program BASIC-AMSTRAD, trebuie cunoscute și respectate următoarele **reguli generale**:

- comanda **RUN** realizează lansarea în execuție a programului curent existent în memoria calculatorului de la instrucțiunea cu cel mai mic număr de linie;
- comanda **RUN** (nr. linie) lansează în execuție programul curent existent în memoria calculatorului de la instrucțiunea care are numărul de linie egal cu (nr. linie);
- comanda **RUN** (șir alfanumeric) încarcă și execută un program BASIC sau un program obiect înregistrat pe dischetă. Această comandă permite accesul direct la programele BASIC protejate;
- instrucțiunea **STOP** întrerupe execuția unui program. Pentru reluarea execuției programului tasteți **CONT**;
- întreruperea momentană a execuției unui program se realizează prin acționarea tastei **[ESC]** o singură dată; reluarea execuției se face prin apăsarea unei alte taste iar oprirea completă a programului se realizează tastind încă o dată **[ESC]**;
- după introducerea datelor prin instrucțiunea **INPUT** se tastează **[RETURN]**;
- lista mesajelor de eroare se găsește în vol. 2, pag. 5.

Dacă programul exemplu nu se află în memoria calculatorului, vă invităm să-l încărcăm de pe dischetă (vezi aplicația de la pag. 108 pe care trebuia să o realizați singuri) ca după aceea să-l executați. Să urmărim împreună execuția programului.

Pe ecran a apărut "Raza=?", calculatorul așteptând ca dvs. să introduceți valoarea pe care o doriți. Vă propunem ca, de această dată, să tastezi 3.3, **[RETURN]**, adică un număr real. Între timp s-a afișat și capul de tabel sub care s-au imprimat (sub controlul aceleiași instrucțiuni **Zone**) valorile 3.3 (număr real) și 3 (număr întreg).

După ce a calculat aria (a) și volumul (v) rezervorului sferic de rază r (număr real), calculatorul AMSTRAD afișează rezultatul ca în figura:

Raza = ? 3.3	
RAZA (Nr. real)	RAZA (Nr. întreg)
3.3	3
R=3.3 A=136.847776	V=150.532554

Puteți relua execuția programului ori de câte ori doriți; încercați să introduceți pentru r și alte valori ca: 3, 0.8 etc.

Aplicație. Modificați programul exemplu, astfel încât să calculeze aria și volumul rezervorului sferic numai pentru valori întregi ale razei r .

TESTE

a) Se consideră următorul program BASIC:

```

10 REM PROGRAMUL CITEȘTE TREI
20 REM NOTE (N1, N2, N3) ALE UNUI
30 REM STUDENT, CALCULEAZĂ
   MEDIA (M)
40 REM ȘI TIPĂREȘTE NOTELE ȘI MEDIA
50 PRINT "INTRODUCEȚI PRIMA NOTĂ";
60 INPUT N1
70 PRINT "INTRODUCEȚI A DOUA
   NOTĂ";
90 PRINT "INTRODUCEȚI A TREIA
   NOTĂ";
100 INPUT N3
110 LET M=(N1+N2+N3)/3
120 PRINT
130 PRINT "MEDIA ESTE"; M
140 END
75 INPUT N2

```

În fig. 3.9 se prezintă dialogul cu calculatorul AMSTRAD.

```

RUN
INTRODUCETI PRIMA NOTA ? 8
INTRODUCETI A DOUA NOTA ? 0
INTRODUCETI A TREIA NOTA ? 10
MEDIA ESTE 6

```

Fig. 3.9

După fiecare semn de întrebare utilizatorul tastează valoarea variabilei apoi apasă pe tasta [RETURN].

După ce au fost introduse cele trei valori, calculatorul calculează media aritmetică și afișează rezultatul.

Ce valoare a introdus utilizatorul pentru variabila N1?

Dar pentru variabila N2?

RUN

```

INTRODUCETI PRIMA NOTĂ? 5
INTRODUCETI A DOUA NOTĂ? 0
INTRODUCETI A TREIA NOTĂ? 10
MEDIA ESTE 5

```

b) Scrieți un program BASIC-AMSTRAD pentru înmulțirea a două binoame:

$$(AX+B) \cdot (CX+D).$$

Valorile lui A, B, C, D se vor introduce dinamic. Pe ecran se vor afișa trei valori reprezentând coeficienții lui X^2, X și termenul liber.

```

10 PRINT "A, B, C, D";
20 INPUT A, B, C, D
30 PRINT A * C; A * D + B * C;
   B * D
40 END

```

run

```

A, B, C, D ? 1, 2, 3, 4
3 10 8

```

Ready



c) Precizați rezultatul execuției următorului program BASIC-AMSTRAD dacă lui A și B li se atribuie dinamic valorile 3, respectiv 2.

```

10 cls
20 print "cine este 'A' "
30 input a
40 print "cine este 'B' "
50 input b

```

```
60 cls
70 print a+"b"="a+b
80 end
```

R.

```
3+2=5
Ready
■
```

□ Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag. 203

Scrierea programului.

Reguli generale

Pentru scrierea unui program în limbajul BASIC-COMMODORE trebuie cunoscute și respectate următoarele **reguli** generale:

- fiecare linie program începe cu un număr de linie;
- o linie program poate conține una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- o variabilă este identificată printr-un șir de caractere care începe cu o literă, urmată de un număr oarecare de litere sau cifre dintre care numai primele două caractere sînt semnificative;
- variabilele admise sînt de următoarele tipuri: numerice reale, numerice întregi, șiruri de caractere;
- tipul unei variabile se poate stabili și prin caracterele speciale: "%0" (pentru variabilele numerice întregi); "\$" (pentru șiruri de caractere) ce se atașează numelui variabilei;
- numele unei variabile indexate este orice nume de variabilă acceptat de către BASIC-COMMODORE;
- valoarea minimă a indicelui unui tablou este zero;
- numărul de indici ai unei variabile indexate este de la 1-3;
- operatorii aritmetici admiși sînt: "+" (pentru adunare); "-" (pentru scădere); "*" (pentru înmulțire); "/" (pentru împărțire); "^" (pentru ridicare la putere);
- operatorii de relație admiși sînt: "=", "<"; ">"; "<="; "=>";
- operatorii logici admiși sînt: **NOT, AND, OR.**
- funcțiile matematice admise sînt: **COS, EXP, LOG, RND, SIN, SQR, TAN, ATN, INT, SGN;**
- funcțiile admise pentru șirurile de caractere sînt: **LEN, POS, MID\$, LEFT\$, RIGHT\$;**
- funcțiile de conversie admise sînt: **CHR\$, VAL, STR\$, ASC;**
- alte funcții (auxiliare) disponibile sînt: **TAB, SPC, POS, FRE, USR, SYS, WAIT;**
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF...THEN, ON...GOTO;**
- instrucțiunea de ciclare permisă este **FOR...NEXT;**
- instrucțiunile de intrare/ieșire permise sînt: **READ, DATA, RESTORE, INPUT, GET, PRINT, POKE; PEEK;**
- instrucțiunile admise pentru operațiile de intrare/ieșire pe disc sînt: **OPEN, CLOSE, PRINT, INPUT, GET, SAVE, LOAD;**
- limbajul nu are instrucțiuni grafice speciale. Se pot utiliza instrucțiunile **POKE** și **PEEK;**
- limbajul nu are instrucțiuni speciale pentru producerea sunetelor. Se pot utiliza instrucțiunile **POKE** și **PEEK.**

Aplicați aceste reguli la scrierea programului. Vă puteți ghida și după lista programului precedent. Pentru tabularea cerută la proiectarea programului puteți folosi una din instrucțiunile **PRINT TAB** sau **PRINT SPC** (identice cu cele de la calculatorul AMSTRAD). Ca și în programul precedent puteți

folosi variabile întregi și reale denumite cu același nume (numai primele două caractere sînt semnificative!).

La tipărirea rezultatului final, nu uitați să editați și valoarea razei pe care o citiți cu instrucțiunea **INPUT**.

Introducerea programului.

Reguli generale

Regulile care stau la baza introducerii și corectării interactive a programelor BASIC-COMMODORE sînt:

- linia program poate fi tastată numai după apariția cursorului "█";
- comanda **LIST** tipărește o parte sau tot programul aflat în memorie (vezi comanda **AMSTRAD**);
- comanda **NEW** elimină din memoria calculatorului programul rezident;
- comanda **CLR** inițializează valorile tuturor variabilelor.

Salvarea programelor pe un suport extern de memorie. Un program BASIC-COMMODORE poate fi salvat pe casetă sau dischetă.

Formatul general al comenzii este:

Format general
SAVE["<nume fișier>"][,a][,b]

Dacă **SAVE** se execută fără parametri, programul curent din memorie (RAM) este înregistrat pe casetă. Dacă se indică <nume fișier> programul curent se salvează (pe casetă) sub acel nume. Dacă salvarea se face pe dischetă, <nume fișier> este obligatoriu. Cel de-al doilea parametru indică perifericul. Dacă se indică 8, programul <nume fișier> se salvează pe dischetă. Dacă cel de-al doilea parametru (b) nu este folosit sau are valoarea 1, fișierul va fi înregistrat pe casetă. Dacă parametrul b are valoarea 1, atunci se mai poate indica și un al treilea parametru. Dacă și pentru cel de-al treilea parametru se fixează valoarea 1, la sfîrșitul fișierului se va imprima eticheta de End of Tape.

În tabelul 3.3 se prezintă cîteva exemple de utilizare a comenzii.

Tabelul 3.3

Comandă	Funcțiune
SAVE "EX1"	salvează un program pe casetă, sub numele EX1.
SAVE "EX1",8	salvează un program pe dischetă, sub numele EX1.
SAVE "EX1",1,1	salvează un program pe casetă, sub numele EX1, cu imprimarea unei etichete End of Tape.

Observație. Programele salvate cu **SAVE** se încarcă cu comanda **LOAD**.

Aplicație. Salvați pe o dischetă (inițializată), sub numele PC3COM, programul exemplu.

Încărcarea programelor de pe un suport extern

Un program BASIC-COMMODORE poate fi încărcat în memoria calculatorului de pe un suport extern (casetă, dischetă) cu comanda **LOAD**. Formatul general al comenzii este:

Format general
LOAD ["(nume program)"] [,a]

Parametrul "a" specifică tipul perifericului (casetă/dischetă). Dacă programul urmează să fi încărcat de pe o dischetă, a ia valoarea 8. De menționat că (nume program) este necesar numai pentru programele salvate pe dischetă. Dacă în loc de (nume program) se indică asterisc "*", de pe dischetă se va încărca (în memoria calculatorului) numai primul program.

În tabelul 3.4 se prezintă mai multe exemple de utilizare a comenzii **LOAD**.

Tabelul 3.4

Comandă	Funcțiune
LOAD	încarcă de pe casetă următorul program.
LOAD "EX1"	încarcă de pe casetă programul EX1.
LOAD "*" ,8	încarcă de pe dischetă primul program găsit.
LOAD "EX1" ,8	încarcă de pe dischetă programul EX1.

Remarci

- Când se execută **LOAD** pentru un program înregistrat pe casetă se afișează mesajele:

```
PRESS PLAY ON TAPE
SEARCHING FOR nume program
FOUND nume program
```

- Odată cu încărcarea programului în memoria calculatorului se afișează mesajul:
LOADING

urmat de:

```
OK
READY
```

atunci când procedura de încărcare **LOAD** este completă.

Aplicație. Încărcați în memoria RAM a calculatorului programul P3COM înregistrat pe dischetă.

Pentru verificarea modului în care a fost realizată salvarea unui program se utilizează comanda **VERIFY** al cărei format general este:

Format general
VERIFY [(nume program) , (dispozitiv)]

Dacă în urma verificării celor două programe (din memorie și de pe suportul extern) rezultă că cele două programe sînt identice, se afișează mesajul:

**OK
READY**

în caz contrar, apare mesajul:

**? VERIFY
READY**

Execuția programului.

Reguli generale

Pentru lansarea în execuție a unui program BASIC-COMMODORE trebuie cunoscute și respectate următoarele **reguli** generale:

- Comanda **RUN** al cărei format general este:

Format general
RUN [(nr. linie)]

lansează în execuție programul curent din memoria internă a calculatorului, de la prima linie program (în absența parametrului (nr. linie)) sau de la cea cu numărul de linie egal cu (nr. linie);

Observație. Dacă într-o comandă **RUN** se specifică (nr. linie) dar linia cu numărul respectiv nu există în program, pe ecran se afișează mesajul:

? UNDEF'D STATEMENT ERROR

- la întîlnirea instrucțiunii **STOP**, execuția programului se oprește temporar cu mesajul:

BREAK IN XXX

unde **XXX** este numărul de linie al instrucțiunii **STOP**. Pentru reluarea execuției tastezi **[CONT]**;

- execuția unui program poate fi oprită prin acționarea tastei **[BREAK]**.

Din acest moment, nimeni nu vă mai oprește să conversați cu **COMMODORE**. Răspundeți concret la întrebarea ce vi se adresează ("RAZA=?") și analizați răspunsul primit. Oricare ar fi rezultatul, continuați!

Aplicație. Să se determine coordonatele centrului de greutate al plăcii din figura de mai jos, dacă dimensiunile sînt următoarele: $AB=20$ cm; $BD=24$ cm; $ED=10$ cm; $AN=2$ cm; $NL=18$ cm; $LK=20$ cm; $FK=8$ cm.

Indicație. Se împarte placa în trei arii elementare și se alege sistemul de axe din figură.

Specificațiile de programare și documentația de proiectare sînt prezentate în modulele de analiză și proiectare structurată, fig. 3.10.

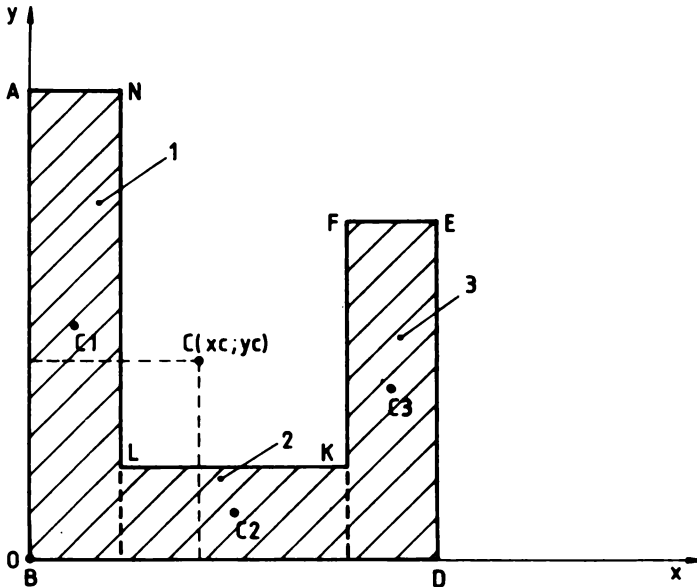


TABELA DE VARIABILE

Variabile de intrare	Variabila de stare	Variabile de ieșire
A1 : aria primului corp	A : aria totală	X : abscisa centrului de greutate
A2 : aria celui de-al doilea corp		Y : ordonata centrului de greutate
A3 : aria celui de-al treilea corp		
X1 : abscisa centrului de greutate al primului corp		
X2 : abscisa centrului de greutate al celui de-al doilea corp		
X3 : abscisa centrului de greutate al celui de-al treilea corp		
Y1 : ordonata centrului de greutate al primului corp		
Y2 : ordonata centrului de greutate al celui de-al doilea corp		
Y3 : ordonata centrului de greutate al celui de-al treilea corp		

a)

SPECIFICAȚII DE PROGRAMARE

Descrierea programului

Programul citește de la terminal ariile, abscisele și ordonatele centrelor de greutate ale celor trei corpuri, calculează și afișează coordonatele centrului de greutate al plăcii.

Intrări

Ariile, abscisele și ordonatele centrelor de greutate ale celor trei câmpuri care compun placa.

Ieșiri

Abscisa și ordonata centrului de greutate al plăcii.

Lista de funcțiuni ale programului

1. Citește date intrare
2. Calculează abscisa centrului de greutate.
3. Calculează ordonata centrului de greutate.
4. Afișează rezultatul.
5. Stop.

b)

Fig. 3.10. Modulele de analiză și proiectare structurată: a) tabela de variabile; b) specificații de programare;

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

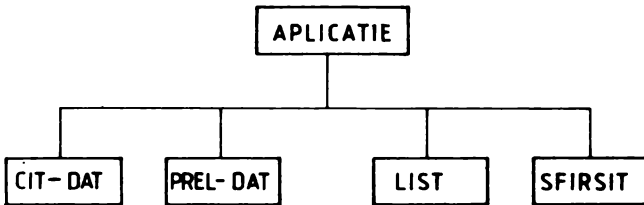
Modul	Funcțiuni
CIT-DAT	1
PREL-DAT	2,3
LIST	4
SFIRȘIT	5

c)

DATELE DE INTRARE

A1	A2	A3	X1	X2	X3	Y1	Y2	Y3
40	40	20	1	12	23	10	1	5

d)



e)

PSEUDOCODUL

APLICAȚIE	SEQ
CIT-DAT	SEQ
	INPUT A1, A2, A3
	INPUT X1, X2, X3
	INPUT Y1, Y2, Y3
CIT-DAT	END
PREL-DAT	SEQ
	A=A1+A2+A3
	X=(A1*X1+A2*X2+A3*X3)/A
	Y=(A1*Y1+A2*Y2+A3*Y3)/A
PREL-DAT	END
	PRINT "X="; X; "Y="; Y
APLICAȚIE	END

f)

Fig. 3.10 c) alocarea funcțiilor de prelucrare; d) datele de intrare; e) diagrama de structură; f) pseudocodul;

```

10 REM PROGRAMUL CALCULEAZĂ CO
   ORDONATELE
20 REM CENTRULUI DE GREUTATE AL
30 REM UNEI PLĂCI PLANE
40 PRINT "A1, A2, A3";
50 INPUT A1, A2, A3;
60 PRINT "X1, X2, X3";
70 INPUT X1, X2, X3
80 PRINT "Y1, Y2, Y3";
90 INPUT Y1, Y2, Y3
100 A=A1+A2+A3
110 X=(A1*X1+A2*X2+A3*X3)/A
120 Y=(A1*Y1+A2*Y2+A3*Y3)/A
125 PRINT
130 PRINT "X="; X, "Y="; Y
140 END
150 STOP
RUN
A1, A2, A3 ?
X1, X2, X3 ?
Y1, Y2, Y3 ?
X=9.80           Y=5.40
Ready
    
```

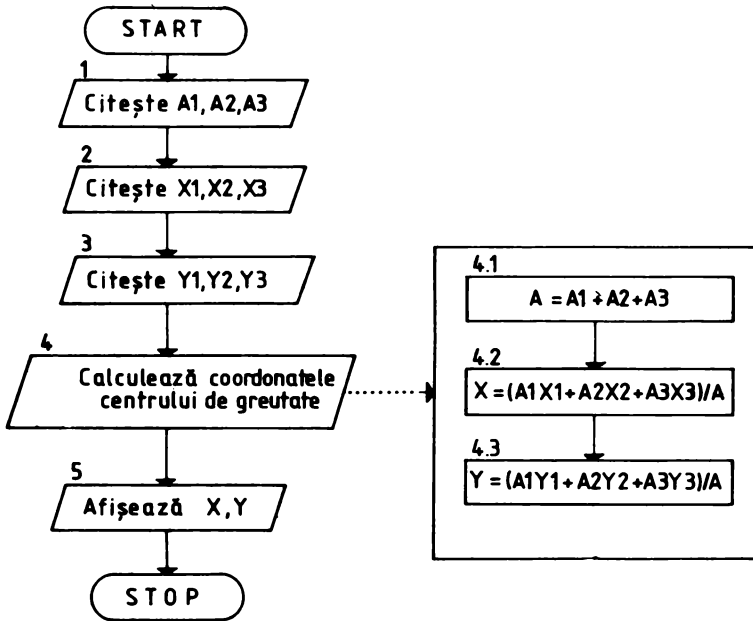


Fig. 3.10. g) schema logică.

TEST

Două conductoare foarte lungi, paralele, aflate la distanța $d=10$ cm unul de celălalt, sînt parcurse de curenți de același sens, de intensități $I_1=5A$ și $I_2=10A$. Să se calculeze inducția magnetică a cimpului rezultat într-un punct situat la jumătatea distanței dintre cele două conductoare.

Indicație. Se va utiliza formula de calcul $B = \frac{\mu_0}{\pi d} (I_2 - I_1)$. Valorile curenților și distanței dintre conductori se vor citi dinamic. Constanta de permeabilitate magnetică a vidului (μ_0) are valoarea: $\mu_0 = 4\pi \cdot 10^{-7} N/A^2$.

```

5 PRINT "D, I2, I1";
10 INPUT D, I2, I1
20 MIU=(4*PI)/10↑7
30 B=(MIU/PI*D)*(I2-I1)
35 PRINT
  
```

```

40 PRINT "INDUCȚIA MAGNETICĂ="; B; "T"
50 END
RUN
D, I2, I1? 0.1, 10.5
INDUCȚIA MAGNETICĂ=
  
```

□ Codificarea în limbajul BASIC-80

vol. 2, pag. 203

Scrierea programului. Reguli generale

Pentru scrierea unui program în limbajul BASIC-80 trebuie cunoscute și respectate următoarele **reguli** generale:

- fiecare linie program începe cu un număr de linie;
- numărul liniei este cuprins între 0 și 65529;

- o linie program poate să conțină una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- numele variabilelor numerice sînt formate dintr-un număr nelimitat de caractere (litere, cifre, caracterul punct) care încep cu o literă, dar numai primele 40 de caractere sînt semnificative;
- variabilele BASIC-80 pot fi: șir, întregi, simplă precizie sau dublă precizie;
- tipul unei variabile se poate stabili și prin caracterele speciale: "\$" (variabilă de tip șir); "%/" (variabilă întregă); "!" (variabilă simplă precizie); "#" (variabilă dublă precizie) ce se atașează numelui variabilei;
- o variabilă poate fi declarată și explicit cu ajutorul instrucțiunilor **DEFINT, DEFSTR, DEFSNG, DEFDBL**;
- numele unei variabile indexate este orice nume de variabilă acceptat de BASIC-80;
- numărul maxim pentru dimensiunile unei variabile indexate este 255;
- numărul maxim de elemente/dimensiune este 32767;
- operatorii aritmetici admiși sînt: "+", "-", "*", "/", "↑";
- operatorii de relație admiși sînt: "=", "<", ">", "<=", "=>", "==", ">=", "<>", "<";
- operatorii logici admiși sînt: **NOT, AND, OR, XOR, IMP** (implicație), **EQV** (echivalență);
- funcțiile aritmetice admise sînt: **ABS, INT, SQR, EXP, LOG, RND, SGN, COS, SIN, TAG, ATN**;
- funcțiile care operează cu valori numerice și au ca rezultat șiruri sînt: **CHR\$, HEX\$, OCT\$, STR\$, SPC, SPACE\$, STRING\$, TAB, MKI\$, MKS\$, MKD\$**;
- funcțiile care operează cu șiruri și au ca rezultat valori numerice sînt: **ASC, LEN, INSTR**;
- funcțiile care operează cu șiruri și au ca rezultat șiruri sînt: **LEFT\$, RIGHT, MID\$**;
- funcțiile de conversie numerică admise sînt: **CDBL, CINT, CSNG, CVI, CVS, CVD**;
- funcțiile de intrare admise sînt: **INKEY\$, INPUT\$, INP, POS, LPOS, PEEK**;
- alte funcții disponibile sînt: **FREE, USR, VARPTR, LOC, EOF, SWAP**;
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, IF ... THEN ... ELSE, IF ... GOTO ..., ON ... GOTO, ON ERROR GOTO**;
- instrucțiunile de ciclare permise sînt: **FOR ... NEXT, WHILE ... WEND**;
- instrucțiunile de intrare/ieșire permise sînt: **INPUT, LINE INPUT, PRINT, PRINT USING, LPRINT, LPRINT USING, WRITE, READ, DATA, RESTORE, OUT, POKE**;
- instrucțiunile pentru operațiile de intrare/ieșire pe disc sînt: **SAVE, LOAD, OPEN, CLOSE, INPUT, LINE INPUT, PRINT, PRINT USING, WRITE, GET, PUT, KILL, NAME**;
- comenzile suplimentare în BASIC GRAPHIC (GBASIC) admise sînt: **VIEWPORT, WINDOW, MOVE, RMOVE, DRAW, RDRAW, ROTATE**.

Vom aplica aceste reguli la scrierea programului.

```

5 PRINT CHR$(24)
10 REM Programul citește raza (R)
15 REM a unui rezervor sferic
20 REM calculează aria (A),
   volumul (V) și
30 REM tipărește: Raza, Aria,
   Volumul
35 PI=3.14159
40 PRINT "Raza=";
45 INPUT R
50 R0=R

60 PRINT "RAZA (Nr. real)"; SPC
   (7); "RAZA (Nr. întreg)
65 PRINT
70 PRINT R; SPC(15); R0
80 A=4*PI*R^2
90 V=4*PI*R^3/3
100 PRINT
110 PRINT "R="; R; "A="; A;
   "V="; V
120 PRINT
130 STOP
140 END

```

Prima linie am scris-o pentru ștergerea ecranului (vezi conversația 2), iar în linia 35 am definit noi valoarea constantei PI (vezi conversația 2). Cum și în BASIC-80 distingem variabile numerice întregi și reale, am procedat la evidențierea prin program a acestei importante facilități introducînd instrucțiunile din liniile 50, 60, 70. Pentru tabularea impusă prin proiectare am utilizat în cadrul instrucțiunilor **PRINT** (liniile 60 și 70) funcția **SPC** care

are aceeași semnificație ca și la BASIC-AMSTRAD. Tot atât de bine putem utiliza și funcția **TAB** scriind în acest sens instrucțiunile din liniile 60 și 70 sub forma:

```
60 PRINT "RAZA (Nr. real)"; TAB (21); "RAZA (Nr. întreg)"
70 PRINT R; TAB (21); R%
```

Instrucțiunile **STOP** și **END** termină execuția programului redând controlul sistemului.

Observație. În versiunea extinsă BASIC-80 comentariile se pot pune între ghilimele (exemplu: 10 R=3 'inițializare').

Instrucțiunea LPRINT

Prezentăm în continuare o variantă simplificată a programului:

```
10 REM "Programul citește raza (R) a          40 PRINT "RAZA=";
unui rezervor sferic",                      50 INPUT R
20 REM "calculează aria (A), volumul (V)    60 LET A=4*PI*R^2
și tipărește:"                               70 LET V=4*PI*R^3/3
30 REM "Raza, Aria, Volumul"               80 LPRINT "R="; R; "A="; A;
35 PI=3.14159                               "V="; V
                                           90 END
```

în care am urmărit ca afișarea rezultatelor să se facă la imprimanta sistemului. În acest sens am scris instrucțiunea 80:

```
80 LPRINT "R="; R; "A="; A; "V="; V
```

înlocuind pe **PRINT** cu **LPRINT**. **LPRINT** tipărește valorile expresiilor din listă la imprimantă. Formatul general este:

Format general

<nr. linie>**LPRINT**{<listă exp.>}

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii și corectării interactive a programelor BASIC-80 sînt:

- linia program poate fi tastată numai după apariția cursorului;
- pentru introducerea unui cuvînt cheie BASIC în calculator puteți utiliza indiferent MAJUSCULE sau minuscule;
- pentru ștergerea memoriei înainte de introducerea unui nou program se utilizează comanda **NEW**;
- comanda **CLEAR** inițializează toate variabilele numerice cu zero;
- comanda **LIST** listează o parte sau tot programul aflat în memorie. Formatul general al comenzii este:

Format general

LIST [{nr. linie}]

 Format general

LIST [*nr. linie 1*][-[*nr. linie 2*]]]

Dacă *nr. linie* este omis, programul se listează integral de la linia cu cel mai mic număr. Dacă se indică *nr. linie* se afișează numai linia cu acest număr! Dacă este specificat numai *nr. linie 1* se listează toate liniile program cu numărul mai mare sau egal cu acesta. Dacă se specifică numai *nr. linie 2* se listează toate liniile de la începutul programului până la linia respectivă, inclusiv. În sfârșit, dacă ambele numere de linii sint specificate se listează zona de program cuprinsă între ele, inclusiv.

În tabelul 3.5 sint prezentate câteva exemple de utilizare a comenzii;

Tabelul 3.5

Comandă	Funcțiune
LIST	listează programul începind cu linia cu cel mai mic număr
LIST 50	afișează linia 50
LIST -250	listează toate liniile de la începutul programului până la linia 250 inclusiv
LIST 150-250	listează toate liniile program cuprinse cu numerele 150 până la 250 inclusiv

- comanda **LLIST** listează o parte sau tot programul la imprimantă (cu 132 caractere pe linie); Formatul general este:

 Format general

LLIST [*nr. linie 1*][-[*nr. linie 2*]]]

- listarea programului se poate intrerupe prin **CTRL/C**;
- pentru ca linia să fie prelucrată se va acționa tasta **[RETURN]**;
- ștergerea ultimului caracter tastat se realizează prin **CTRL/H**;
- tabularea din 8 în 8 coloane se realizează prin **CTRL/I**;
- **CTRL/R** reafișează linia tastată;
- **CTRL/U** șterge linia curentă;
- pentru tastarea majusculor și a simbolurilor superioare înscrise pe taste, mențineți apăsată tasta **[SHIFT]**;
- **AUTO** generează automat un număr de linie. Formatul general este:

 Format general

AUTO[*nr. linie*][,*increment*]]

Valorile implicite pentru *nr. linie* și *increment* sint 10. Dacă un număr de linie există deja, se va afișa caracterul "*" semnalind că linia ce urmează va înlocui o linie existentă. În tabelul 3.6 se prezintă câteva exemple de utilizare a comenzii;

Tabelul 3.6

Comandă	Funcțiune
AUTO	generează numerele de linie: 10, 20, 30, 40, ...
AUTO 18	generează numerele de linie: 18, 28, 38, ...
AUTO 200, 50	generează numerele de linie: 200, 250, 300, ...

Observație. Ieșirea din modul „**AUTO**” se realizează prin **CTRL/C**.

- comanda **DELETE** șterge linii din program. Formatul general este:

Format general
DELETE [(nr. linie)][-(nr. linie)]

Observație. Dacă linia pe care dorim s-o ștergem nu există în program, se afișează un mesaj de eroare.

- comanda **EDIT** permite modificarea unei porțiuni dintr-o linie program fără a mai reintroduce întreaga linie. Formatul general este:

Format general
EDIT (nr. linie)

Observație. Intrarea în mod editare pe linia curentă se poate face și cu **CTRL/A**. În tabelul 3.7 este prezentat subșetul de comenzi de editare.

Tabelul 3.7

Comandă	Funcțiune
" "	mută cursorul la dreapta.
<RUBOUT>	mută cursorul la stînga.
I (TEXT) \$	inserează text pe poziția curentă a cursorului.
X	extinde linia; mută cursorul la sfîrșitul liniei unde se poate insera <TEXT> <CR>.
nD	șterge n caractere la dreapta cursorului.
H	șterge toate caracterele de la dreapta cursorului.
nS (ch)	caută a n-a apariție a caracterului (ch) și poziționează cursorul înainte de el.
nK (ch)	șterge a n-a apariție a caracterului (ch).
C (ch)	înlocuiește următorul caracter cu (ch).
<CR>	se salvează modificările făcute în linie și se iese din modul EDIT ; se tipărește restul liniei
E	idem <CR> minus tipărire
Q	se dă controlul interpretorului BASIC-80 după ce se salvează orice modificare făcută cu EDIT .
L	listează restul liniei și poziționează cursorul la începutul liniei.
A	permite o nouă editare a unei linii.

Salvarea unui program BASIC pe dischetă

Pentru salvarea unui program BASIC-80 pe o dischetă, se folosește comanda **SAVE** al cărei format general este:

Format general
SAVE "<nume fișier>" [, A , P]

Dacă <nume fișier> există deja pe dischetă, programul salvat va fi scris peste conținutul vechiului fișier. Parametrul **A** determină salvarea programului în format ASCII, iar parametrul **P** conduce la protejarea programului.

Aplicație. Salvați programul pe o dischetă (inițializată!)* sub numele PC380. Verificați (vezi comanda **DIR** sub CP/M) dacă programul a fost salvat.

Încărcarea unui program de pe dischetă

Programele de pe dischetă se încarcă în memoria calculatorului cu ajutorul comenzii **LOAD** al cărei format general este:

Format general
LOAD "<nume fișier>" [, R]

LOAD încarcă un fișier de tip **BAS** (sub CP/M) în memorie. Comanda **LOAD** închide înaintea încărcării toate fișierele deschise anterior și șterge din memorie toate variabilele și liniile program rezidente. **LOAD** cu opțiunea **R** încarcă și execută programul de pe dischetă.

Observație. Opțiunea **R** poate fi folosită și pentru înlănuirea programelor ori a segmentelor aparținând aceluiași program.

Aplicație. Încărcați în memorie, de pe dischetă, programul PC380.

Pentru adăugarea unui program înregistrat pe dischetă la programul curent din memorie puteți utiliza instrucțiunea **MERGE** al cărei format general este:

Format general
MERGE "<nume fișier>"

Dacă o linie din <nume fișier> are același număr cu o linie din programul rezident în memorie, linia din <nume fișier> înlocuiește linia din memorie.

Observație. După executarea unei instrucțiuni **MERGE**, se intră în modul de lucru direct.

Execuția programului. Reguli generale

Pentru lansarea în execuție a unui program BASIC-80, trebuie cunoscute și respectate următoarele **reguli** generale:

- comanda **RUN** execută un program aflat în memorie. Formatul general este:

Format general
RUN [<nr. linie>]

Dacă se specifică <nr. linie> atunci se începe execuția cu acea linie;

- comanda **RUN** încarcă un fișier program de pe dischetă și îl execută. Formatul general este:

* Calculatoarele M 118, JUNIOR și TPD folosesc dischete de 8" sau 5".

Format general
RUN "(nume fișier)"[R]

RUN închide toate fișierele (deschise) și șterge conținutul memoriei înainte de a încărca un program de pe dischetă. Opțiunea **R** permite păstrarea tuturor fișierelor de date deschise anterior;

Observație. (nume fișier). reprezintă numele din instrucțiunea **SAVE** cu care s-a salvat programul pe dischetă.

- instrucțiunea **STOP** încheie execuția programului și redă controlul interpretorului BASIC; Formatul general este:

Format general
STOP

- instrucțiunea **STOP** nu determină închiderea fișierelor. Ori de câte ori se întâlnește un **STOP** în program se va afișa mesajul:

Break in line nnnn

- întreruperea momentană a execuției unui program se realizează prin **CTRL/C**;
- **CTRL/O** suspendă o așteptare de intrare/ieșire (I/E) și continuă execuția programului. Un nou **CTRL/O** activează I/E;
- pentru urmărirea modului de execuție a unui program se utilizează instrucțiunile **TRON** și **TROFF** ale căror formate generale sînt:

Format general
TRON

Format general
TROFF

TRON determină tipărirea între "[]" a numerelor liniilor executate. Instrucțiunea **TROFF** anulează pe **TRON**;

- după introducerea datelor prin instrucțiunea **INPUT** se tastează [**RETURN**].

Executați programul PC380 (de pe dischetă!) cu una din comenzile învățate. Conversația pe care o purtați cu calculatorul este cea obișnuită. La apariția mesajului "RAZA=?" tastați valoarea razei pentru care doriți să calculați aria și volumul rezervorului (de exemplu 3.5), după care reflectați puțin la „replica” partenerului dvs. de conversație. Dacă pentru A și V vi s-au comunicat valorile 153.938 respectiv 179.594 înseamnă că dialogul dvs. cu cele trei personaje: M 118, JUNIOR și TPD și-a atins scopul. Felicitări! În caz contrar, reluați conversația.

```
R = 3.5  A = 153.938  V = 179.594
R = 1.78  A = 39.8153  V = 23.6237
R = 10.2  A = 1307.4  V = 4445.17
R = 15  A = 2827.43  V = 14137.2
```

Fig. 3.11

Vă prezentăm în fig.3.11 rezultatul execuției programului variantă (vezi pag. 118) obținut la imprimantă.

Ce puteți spune despre forma de editare a celor trei valori? Pare dezordonată, nu-i așa? Lipsesc alinierea! Vom remedia aceste neajunsuri în conversația imediat următoare.

Aplicație. Scrieți un program BASIC care rezolvă următoarele sisteme liniare de două ecuații cu două necunoscute.

a) $\begin{cases} 2x+5y=8 \\ 3x-2y=-7 \end{cases}$

b) $\begin{cases} 3x-2y=1 \\ 5x+y=6 \end{cases}$

c) $\begin{cases} 12x+5y=0 \\ 8x+10y=8 \end{cases}$

Indicație. Pentru rezolvarea acestor sisteme se utilizează formulele de calcul:

$$x = \frac{b_2k_1 - b_1k_2}{a_1b_2 - a_2b_1}; \quad y = \frac{a_1k_2 - a_2k_1}{a_1b_2 - a_2b_1}$$

unde:

$a_1, b_1, k_1, a_2, b_2, k_2$ au semnificația de mai jos:

$$\begin{cases} a_1x + b_1y = k_1 \\ a_2x + b_2y = k_2 \end{cases}$$

```
10 print "a1, b1, k1";
20 input a1, b1, k1
30 print "a2, b2, k2";
40 input a2, b2, k2
50 d=a1*b2-a2*b1
60 x=(b2*k1-b1*k2)/d
70 y=(a1*k2-a2*k1)/d
75 print
80 print "X="; x, "Y="; y
```

```
run
a1, b1, k1? 3, -2, 1
a2, b2, k2? 5, 1, 6
X=1 Y=1
run
```

```
a1, b1, k1? 12, 5, 0
a2, b2, k2? 8, 10, 8
X=-0.5 Y=1.2
ready
```

```
run
a1, b1, k1? 2, 5, 8
a2, b2, k2? 3, -2, -7
X=-1 Y=2
```

d) Să se calculeze valoarea expresiei:

$$E = \frac{a+b-1}{a+b} + \frac{1}{2} \left(\frac{a+b-1}{a+b} \right)^2 + \frac{1}{3} \left(\frac{a+b-1}{a+b} \right)^3 + \frac{1}{4} \left(\frac{a+b-1}{a+b} \right)^4 + \frac{1}{5} \left(\frac{a+b-1}{a+b} \right)^5 + \frac{1}{6} \left(\frac{a+b-1}{a+b} \right)^6$$

în care valorile lui a și b se vor introduce dinamic.

```
10 PRINT "A, B";
20 INPUT A, B
30 C=1-1/(A+B)
40 E=C+C^2/2+C^3/3+C^4/4+
+C^5/5+C^6/6
50 PRINT "E="; E
60 END
```

```
RUN
A, B? 2,3
E=
```

□ Codificarea în limbajul BASIC-PLUS

vol. 2, pag. 204

Scrierea programului. Reguli generale

Pentru scrierea unui program în limbajul BASIC-PLUS trebuie cunoscute și respectate următoarele **reguli** generale:

- fiecare linie program începe cu un număr de linie;
- numărul liniei este cuprins între 1 și 32767;
- o linie program poate să conțină una sau mai multe instrucțiuni separate între ele prin două puncte (":");
- numele unei variabile este format dintr-o literă sau dintr-o literă urmată de a singură cifră;
- variabilele admise sînt de următoarele tipuri: variabile simple numerice întregi, variabile simple numerice reale, variabile simple nenumerice (șir de caractere), variabile indexate numerice (întregi și reale), variabile indexate nenumerice (șir de caractere);
- variabila simplă întregă este identificată prin orice literă (urmată opțional de o cifră) urmată de %;
- variabila simplă reală este identificată printr-o literă urmată opțional de o cifră;
- variabila simplă șir este identificată printr-o literă (urmată opțional de o cifră) urmată de caracterul \$;
- variabila indexată întregă este identificată prin orice nume de variabilă simplă întregă;
- variabila indexată reală este identificată prin orice nume de variabilă simplă reală;
- variabila indexată șir este identificată prin orice nume de variabilă simplă șir;
- operatorii aritmetici admiși sînt: "+" (pentru adunare); "-" (pentru scădere); "*" (pentru înmulțire); "/" (pentru împărțire); "**" (pentru ridicare la putere);
- operatorii de relație admiși sînt: "="; "<"; "<="; ">"; ">="; "<>" (diferit); "==" (aproximativ egal);
- operatorii logici admiși sînt: **NOT, AND, OR, XOR, EQV, IMP;**
- funcțiile matematice admise sînt: **ABS, ATN, COS, EXP, FIX, INT, LOG, LOG 10, NRE** (constantă reală specială $e=2.71828$), **PI, RAD** (constantă reală specială $\sqrt{2}=1.41421$), **RND, SGN, SIN, SQR;**
- funcțiile nenumerice standard admise sînt: **ASCII, CHR\$, DATE\$, INSTR, LEFT, LEN, MID, NUM\$, RIGHT, SPACES, STRING\$, TIMES, VAL, CVT %/\$, CVT\$ %/, CVT\$, CVT\$F;**
- instrucțiunile de salt recunoscute de interpretor sînt: **GOTO, ON...GOTO, IF... THEN... ELSE, IF... GOTO**, modificatorul **IF**;
- instrucțiunile de ciclare permise sînt: **FOR...NEXT, FOR WHILE, FOR UNTIL**, modificatorul **WHILE**, modificatorul **UNTIL**, modificatorul **FOR**;
- instrucțiunile de intrare/ieșire permise sînt: **INPUT, INPUT LINE, READ, DATA, RESTORE, PRINT, PRINT USING, MAT READ, MAT INPUT, MAT PRINT;**
- instrucțiunile pentru intrare/ieșire pe disc sînt: **OPEN, CLOSE, INPUT, INPUT LINE, PRINT, PRINT...USING, MAT INPUT, OPEN...AS VIRTUAL, GET, PUT, SAVE, UNSAVE, LOAD, APPEND, LENGTH, REPLACE.**

Vom aplica aceste reguli la scrierea programului.

Ceea ce mai poate reține atenția la acest program este utilizarea lui **TAB (40)** în cadrul instrucțiunilor **PRINT** din liniile 180 și 190 pentru o mai largă spațiere a: capului de tabel (denumirea celei de-a doua coloane se imprimă în coloana 40 și nu în 20 cum eram obișnuți) și a valorilor lui **R, R⁰/₀**, ultima imprimindu-se începînd tot din coloana 40.

Și, încă o remarcă foarte importantă pe care trebuie s-o facem! Variabilele numerice $R^0/0$ (de tip întreg) și R (de tip real), deși au același nume sînt de tipuri diferite (vezi liniile 150, 160, 190).

Observație. În BASIC-PLUS comentariile se pot introduce și prin simbolul "!" (vezi Conversația 4).

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii și corectării interactive a programelor BASIC-PLUS sînt:

- linia program poate fi tastată numai după apariția prompterului " : " ;
- o instrucțiune BASIC-PLUS poate ocupa una sau mai multe linii program, trecerea de pe o linie pe alta indicîndu-se prin caracterul "8" plasat pe ultima poziție a liniei care se continuă;
- ștergerea unei linii program înainte de a se fi tastat [RETURN] se realizează prin acționarea simultană a tastelor [CTRL], [U];
- în timpul tastării unei linii program anularea unor caractere eronate se realizează prin acționarea tastei [DELETE (RUB OUT)];
- comanda DELETE elimină din program un grup de linii. Formatul general este:

Format general

DEL [ETE] <nr. linie> | -<nr. linie> | <nr. linie 1>-<nr. linie 2>

Dacă se indică <nr. linie> se elimină toate liniile program cu numărul mai mare sau egal cu acesta. Dacă se specifică - <nr. linie> se elimină toate liniile din program pînă la linia respectivă, inclusiv. Dacă se indică <nr. linie 1> - <nr. linie 2> se elimină zona de program cuprinsă între ele inclusiv. În tabelul 3.8 sînt prezentate cîteva exemple de utilizare a comenzii DELETE.

Tabelul 3.8

Comanda	Funcțiune
: DELETE 80	elimină toate liniile program cu numărul ≥ 80 .
: DEL -80	elimină toate liniile din program pînă la linia 80, inclusiv.
: DEL 200-850	elimină din program toate liniile cuprinse între 200 și 850 inclusiv.

- comanda LIST tipărește o parte sau tot programul aflat în memorie. Formatul general al comenzii este:

Format general

LIST[T] [<nr. linie> | -<nr. linie> | <nr. linie 1> - <nr. linie 2>]

Dacă <nr. linie> este omis, programul se listează integral de la linia cu cel mai mic număr. Dacă se indică <nr. linie>, se afișează toate liniile program cu numărul mai mare sau egal cu acesta. Dacă se indică - <nr. linie> se listează toate liniile program pînă la linia respectivă, inclusiv. Dacă se indică <nr. linie 1> - <nr. linie 2> se listează zona de program cuprinsă între ele, inclusiv. În tabelul 3.9 sînt prezentate cîteva exemple de utilizare a comenzii.

Tabelul 3.9

Comanda	Funcțiune
: LIST 20	listează toate liniile program cu numărul ≥ 20 .
: LIST -800	listează toate liniile din program pînă la linia 800, inclusiv.

- comanda **NEW** elimină din memoria calculatorului programul rezident. Formatul general este:

Format general

NEW [NO LIST] [(nume program)]

Prezența parametrului opțional **NO LIST** (elimină informațiile privind textul sursă al programului) împiedică listarea programului. (nume program) reprezintă numele (conține cel mult 9 caractere) sub care se introduce noul program. În tabelul 3.10 se prezintă două exemple în acest sens;

Tabelul 3.10

Comanda	Funcțiune
: NEW	elimină din memoria internă programul rezident; (nume program) al programului ce urmează a fi introdus va fi un nume standard BASIC.
: NEW M3BPLUS	elimină din memoria internă programul rezident; (nume program) al programului ce urmează a fi introdus va fi: M3BPLUS.

- comanda **RENAME** modifică numele asociat programului utilizator. Formatul general este:

Format general

REN [AME] [NO LIST] [(nume program)]

Comanda modifică (opțional) și acțiunea parametrului **NOLIST** (v. tabelul 3.11).

Tabelul 3.11

Comanda	Funcțiune
: RENAME NO LIST M3BPLUS	asociază programului utilizator numele M3BPLUS și împiedică listarea programului.
: RENAME	asociază programului curent numele standard BASIC.

Observație. (nume program) conține cel mult 9 caractere și are aceeași semnificație ca la comanda **NEW**.

Salvarea programelor (din memoria internă) pe un suport extern

Puțină teorie... Un program BASIC-PLUS poate fi salvat pe bandă magnetică, disc magnetic etc. Comanda cu care se salvează programul **-SAVE** are formatul general:

Format general

SAV[E] [(specificator program)]

Așadar, **SAVE** (comandă de bibliotecă!) salvează programul curent utilizator existent în memoria internă pe un suport extern precizat prin

parametrul (specificator program). Formatul general al specificatorului de program este:

Format general
(suport): [(grup), (membru)] (nume) (tip); (versiune)

Pentru moment facem precizarea că (suport) reprezintă unitatea periferică fizică sau logică pe care s-a încărcat fișierul (DK – disk; MM – bandă magnetică; TT – terminal; LP – imprimantă etc.), (tip) precizează natura fișierului: **BAC** – fișier program în format sursă; **DAT** – fișier de date; **TMP** – fișier temporar, iar (nume) reprezintă numele fișierului alcătuit din 9 caractere alfanumerice.

Fișierul creat pe suportul extern primește (nume) din (specificator fișier), în caz că acesta a fost precizat, sau numele său intern, în caz contrar. De notat că programul se salvează conform parametrului (tip) indicat.

În tabelul 3.12 se prezintă câteva exemple de utilizare a comenzii **SAVE**.

Tabelul 3.12

Comandă	Funcțiune
: SAVE	salvează programul curent utilizator într-un fișier aflat pe discul sistem, sub numele său intern și în format sursă.
: SAVE DK 2 : P3PLUS.BAC	salvează programul utilizator P3PLUS existent în memoria internă pe discul DK2, în format sursă.
: SAVE LP :	listează programul curent utilizator la imprimantă.

Pentru a șterge de pe suportul extern un program existent (salvat cu comanda **SAVE**) se folosește comanda **UNSAVE**:

Format general
UNS [AVE] (specificator program)

Parametrul (specificator program) trebuie să conțină în mod obligatoriu numărul de versiune al fișierului. În caz contrar, comanda nu va fi acceptată.

Aplicație. Salvați programul existent în memoria internă a calculatorului într-un fișier program M3BPLUS, pe discul sistem, în format sursă.

Încărcarea programelor de pe un suport extern în memoria internă

Pentru a încărca în memoria internă a calculatoarelor INDEPENDENT și CORAL un program aflat pe un suport extern se utilizează comanda de bibliotecă **LOAD** al cărei format general este:

Format general
LOA[D] [NO LIST] (specificator program)

Încărcarea programului se realizează în deplină concordanță cu informațiile completate în parametrul <specificator program>.

Comanda **LOAD** îndeplinește următoarele funcțiuni: 1) încarcă programul BASIC de pe suportul extern în memoria internă; 2) înlocuiește numele programului curent utilizator cu numele indicat în <specificator program>; 3) șterge din memorie programul BASIC curent.

De notat că parametrul **[NOLIST]** al comenzii se activează sau nu funcție atît de comanda respectivă, cît și de caracteristica programului indicat în <specificator program>. Mai precizăm că orice program identificat în <specificator program> cu **BAC** sau orice alt tip (**DAT**, **TMP**) se consideră program sursă BASIC. În sfîrșit, o sugestie! Înainte de a tasta comanda **LOAD**, salvați pe suport extern programul curent utilizator aflat în memoria internă a calculatorului.

Aplicație. Încărcați de pe discul sistem în memoria internă programul sursă M3BPLUS.

Execuția programului. Reguli generale

Pentru lansarea în execuție a unui program BASIC-PLUS trebuie cunoscute și respectate [51] următoarele **reguli** generale:

- comanda **RUN** al cărei format general este:

Format general

RUN [NO LIST] [(specificator program)] [, (nr. linie)]

lansează în execuție programul utilizator existent în memoria internă a calculatorului sau pe orice alt suport de memorie externă (bandă, disc etc.), de la prima linie program (în absența parametrului (nr. linie)) sau de la cea avînd numărul de linie egal cu (nr. linie).

- comanda **RUN** (specificator program) îndeplinește funcțiunile: 1) elimină din memorie programul utilizator curent (vezi comanda **NEW**); 2) încarcă în memorie programul (fișierul) indicat prin parametrul <specificator program> (vezi comanda **LOAD**); 3) lansează în execuție programul încărcat;
- la întîlnirea instrucțiunii **STOP**, execuția programului se oprește temporar cu mesajul „STOP AT LINE n”, n fiind numărul de linie al instrucțiunii;
- la întîlnirea instrucțiunii **END**, execuția programului este oprită prin mesajul:

END OF PROGRAM AT LINE n

n fiind numărul de linie al instrucțiunii **END**;

- comanda **CONTINUE** al cărei format general este:

Format general

CON[ITINUE]

lansează în execuție programul utilizator (din punctul de întrerupere datorat ultimei instrucțiuni **STOP**);

- comanda **HELP** (vezi Conversația 11) explicitează mesajul de eroare ce apare în timpul introducerii sau execuției unui program BASIC;
- mesajele de eroare sînt prezentate în vol. 2, pag. 7;
- după introducerea datelor prin instrucțiunea **INPUT** se tastează **[RETURN]**;
- dacă în timpul execuției programului se acționează simultan tastele **[CTRL]** și **[Z]** programul se oprește din execuție cu un mesaj de eroare.

Începeți conversația. Atenție la ceea ce răspundeți... prietenului pe care îl aveți în față dvs. (INDEPENDENT sau CORAL) la întrebarea: RAZA=? Vă sugerăm următoarele... posibilități: 3; 3.33; 3.0; 3.2 etc. Nu uitați ca în finalul conversației, plăcute de altfel, să analizați și dvs. răspunsurile partenerului de conversație. În ceea ce ne privește (vezi fig. 3.12) vă asigurăm că sînt excelente.

```

:RUN
RAZA = ?3

RAZA ( NR.REAL )          RAZA ( NR.INTREG )
  3                          3

R = 3          A = 113.097  V = 113.097

STOP AT LINE 250
:RUN
RAZA = ?3.2

RAZA ( NR.REAL )          RAZA ( NR.INTREG )
  3.2                      3

R = 3.2        A = 128.68  V = 137.258

STOP AT LINE 250

```

Fig. 3.12.

□ Codificarea în limbajul ABASIC

vol. 2, pag. 204

Scrierea programului. Reguli generale.

Pentru scrierea unui program în limbajul ABASIC trebuie cunoscute și respectate următoarele **reguli generale**:

- Fiecare linie program începe cu un număr de linie;
- numărul liniei este cuprins între 1 și 32 000;
- o linie program poate să conțină una sau mai multe instrucțiuni separate între ele prin două puncte (" : ");
- o variabilă este identificată printr-un șir de caractere ce începe cu o literă, urmată de un număr oarecare de litere sau cifre;
- variabilele admise sînt de următoarele tipuri: numerice reale, numerice întregi, șiruri de caractere;
- tipul unei variabile se poate stabili și prin caracterele speciale: "I" sau " # " (pentru variabile numerice reale); "% " (pentru variabile numerice întregi); "\$ " (pentru șiruri de caractere) ce se atașează numelui variabilei;
- o variabilă poate fi declarată și implicit cu ajutorul instrucțiunilor admise: DEFINT, DEFDBL, DEFSTR;
- dacă precizarea tipurilor implicite nu este făcută, de utilizator, toate variabilele sînt considerate numerice reale;
- numele unei variabile indexate este orice nume de variabilă acceptat de către ABASIC;
- numărul de dimensiuni ale matricilor declarate este limitat doar de lungimea liniei ABASIC (80 de caractere);
- operatorii aritmetici admiși sînt: "+" (pentru adunare); "-" (pentru scădere); "*" (pentru înmulțire); "/" (pentru împărțire); "**" sau "^" (pentru ridicarea la putere);

- operatorii de relație admiși sint: "="; "<"; ">"; "<="; ">="; "<>" (diferit) și "==" (aproximativ egal);
- operatorii logici admiși sint: **NOT**, **AND**, **OR**, **XOR**, (SAU exclusiv), **EQV** (echivalență) și **IMP** (implicație);
- funcțiile matematice admise sint: **SIN**, **COS**, **TAN**, **TAG**, **ATN**, **PI** (trigonometrice), **SQR**, **EXP**, **LOG**, **LOG 10**, **LG**, **INT**, **ABS**, **SIGN**, **SGN**, **FIX**, **CINT** (algebrice), **RND**, **RANDOMIZE** (generarea de nume aleatoare);
- funcțiile admise pentru șiruri de caractere sint: **LEN**, **TRM\$**, **POS**, **INSTR**, **SEG\$**, **MID\$**, **LEFT\$**, **RIGHT\$**, **STRING\$**, **SRG\$**, **SPACE\$**, **SPC\$**;
- funcțiile de conversie admise sint: **CHR\$**, **CHRA\$**, **VAL**, **STR\$**, **CHRE\$**, **EBC**, **ASC**, **HEX\$**, **OCT\$**;
- alte funcții (auxiliare) disponibile sint: **TAB**, **CPOS**, **BEL\$**, **ERR**, **ERL**, **FREE**, **FRE**, **EOF**, **DATE\$**, **DAY\$**, **TIMES\$**;
- instrucțiunile de salt recunoscute de interpretor sint: **GOTO**, **ON GOTO**, **IF ... THEN ... ELSE**, **IF ... GOTO**, **IF ... THEN ... ELSE ... ENDIF**;
- instrucțiunile de ciclare permise sint: **FOR ... NEXT**, **WHILE ... ENDWHILE**, (**WEND**);
- instrucțiunile de intrare/ieșire permise sint: **INPUT**, **LINPUT**, (**LINE INPUT**, **INPUT LINE**), **READ**, **DATA**, **RESTORE**, **PRINT**, **PRINT USING**, **WRITE**;
- instrucțiunile admise pentru operațiile de intrare/ieșire pe disc sint: **OPEN**, **CLOSE**, **INPUT**, **LINPUT**, **PRINT**, **PRINT USING**, **WRITE**, **SAVE**, **LOAD**.

Vom aplica aceste reguli la scrierea programului. Sinceri să fim, mari secrete nu mai avem să vă mărturisim. Comentariile se introduc, de asemenea, cu **REM** sau prin simbolul "!", o variabilă numerică de tip întreg și una de tip real pot fi definite cu același nume, **PRINT** cu **INPUT** au semnificația știută, iar **TAB** (utilizat într-o instrucțiune **PRINT**) setează, de asemenea, poziția curentă în linia de ieșire, la valoarea dată de expresia din paranteză.

Prezentăm în continuare o variantă simplificată a acestui program, dvs. revenindu-vă sarcina, ca aplicație, să adăugați și instrucțiunile care evidențiază variabilele (inclusiv valori) de tip real și întreg.

```

10 REM 'PROGRAMUL CITEȘTE RAZA (R) A UNUI REZERVOR SFERIC'
20 REM 'CALCULEAZĂ ARIA (A), VOLUMUL (V) ȘI TIPĂREȘTE:RAZA,ARIA,VOLUMUL'
40 PRINT 'RAZA=';
50 INPUT R
60 LET A=4*PI*R^2
70 LET V=4*PI*R^3/3
80 PRINT 'R = ';R;' A = ';A;' V = ';V
90 END

```

Introducerea programului. Reguli generale

Regulile generale care stau la baza introducerii și corectării interactive [36] a programelor ABASIC sint:

- linia program poate fi tastată numai după apariția prompterului ">";
- comanda **DELETE** elimină din program un grup de linii (vezi comanda **BASIC-PLUS**);
- comanda **LIST** tipărește o parte sau tot programul aflat în memorie (vezi comanda **BASIC-PLUS**);
- comanda **NEW** elimină din memoria calculatorului programul rezident (vezi comanda **BASIC-80**);
- comanda **CLEAR** inițializează valorile tuturor variabilelor;

- comenzile **RENUM** și **XCHANGE** permit renumerotarea liniilor program. Formatele generale sînt:

Format general
RENUM <nr. linie început>, <pas>
Format general
XCHANGE <nr. linie început>, <pas>

- comanda **EDIT** permite modificarea unei linii. Formatul general este:

Format general
EDIT <nr. linie>, <șir de substituit>, <șir de intrîdus>

- comanda **AUTO** are funcțiunea cunoscută;
- trecerea din starea **BASIC** în starea **EDITOR** se realizează prin comanda **EXIT**.

Salvarea și încărcarea programelor

Un program **ABASIC** poate fi salvat ca fișier **ARIEL** cu comanda **SAVE**. Formatul general al comenzii este:

Format general
SAVE <nume fișier>

Programul poate fi încărcat în memorie, dintr-un alt fișier sau direct de la terminal.

Ștergerea unui program (fișier **ARIEL**) se realizează cu instrucțiunea **KILL** al cărei format general este:

Format general
KILL <nume fișier>

Cu comanda **LOAD** al cărei format general este:

Format general
LOAD <nume fișier>

se realizează încărcarea unui fișier <nume fișier> precizat în comanda **SAVE**. Programul curent și tabela de variabile definite anterior sînt șterse, iar fișierele active sînt închise. Se intră în starea de așteptare a unei comenzi utilizator. Numerele de linii ale editorului **ARIEL** devin numere de linii **ABASIC**.

Observație. Dacă la crearea fișierului s-au introdus și numerele de linii, acestea sînt preferate numerelor de linii ale editorului.

Aplicație. Salvați programul existent în memoria internă a calculatorului într-un fișier program **F3ABAS**, pe discul sistem. În pasul imediat următor, încărcăți programul în memoria internă a calculatorului.

Execuția programului. Reguli generale

Pentru lansarea în execuție a unui program ABASIC trebuie cunoscute și respectate [52] următoarele **reguli generale**:

- comanda **RUN** al cărei format general este:

Format general
RUN [(nr. linie)]

lansează în execuție programul de la început sau de la linia indicată (nr. linie);

- comanda **RUN**, cu formatul general:

Format general
RUN [(nume fișier) [, (nr. linie)]]

incarcă și lansează în execuție programul curent sau programul (fișier ARIEL) precizat prin (nume fișier) de la prima linie program (în absența parametrului (nr. linie)) sau de la cea avind numărul de linie egal cu (nr. linie);

- comanda **MERGE** permite interclasarea a două fișiere. Formatul general este:

Format general
MERGE (nume fișier) [, (nr. linie)]

- la întâlnirea instrucțiunii **STOP [END]** execuția programului se oprește temporar cu mesajul:

STOPPED AT LINE : n

unde n este numărul de linie al instrucțiunii **STOP**.

- execuția unui program poate fi oprită la cererea explicită a utilizatorului (se tastează **[BREAK]**) sau în timpul execuției unui program (se tastează **[CTRL/C]**);

Și acum, vă invităm să conversați prin intermediul programului F3 ABAS cu calculatorul FELIX-C. Știți și ce întrebare veți primi, doar dvs. ați programat-o! V-ați pregătit din timp răspunsul? Nu uitați că imediat după ce ați dat "replica" trebuie să tastați **[RETURN]**.

În sfârșit, "ultima verba" al calculatorului: STOPPED AT LINE: 90.

Aplicație. Se consideră placa plană omogenă din fig. 3.13 avind dimensiunile date în centimetri. Să se determine coordonatele centrului de greutate.

Indicație. Se împarte placa în trei arii elementare și anume: aria triunghiulară BDC ($A_1=1\ 000\text{ cm}^2$), aria dreptunghiulară ABDO ($A_2=1\ 200\text{ cm}^2$) din care s-a decupat semicercul AEO ($A_3=-628\text{ cm}^2$). Alegind sistemul de axe de coordonate din figură rezultă: $X_1=46,66\text{ cm}$; $Y_1=13,3\text{ cm}$; $X_2=15\text{ cm}$; $Y_2=20\text{ cm}$; $X_3=8,5\text{ cm}$; $Y_3=20\text{ cm}$. Se poate utiliza programul BASIC de la pag. 115, vol. 1. În fig. 3.14 sint prezentate rezultatele execuției programului.

TESTE

a) Scrieți un program ABASIC care calculează

$$E=A*B+C\uparrow D+3$$

pentru diferite valori de intrare ale lui A, B, C și D.

10 PRINT "A, B, C, D";

20 INPUT A, B, C, D

30 E=A*B+C\uparrow D+3

40 PRINT "E="; E

50 END

RUN

A, B, C, D ? 10, 20, 3, 2, 3

E=212

STOPPED AT LINE : 50

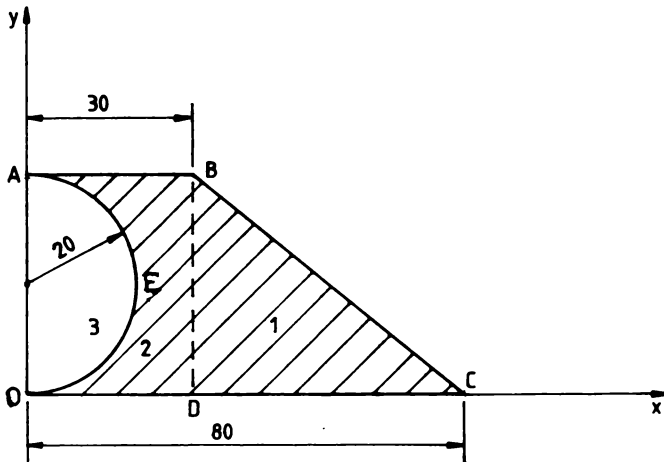


Fig. 3.13.

```

run
A1, A2, A3 ? 1000, 1200, -628
X1, X2, X3 ? 46.66, 15, 8.5
Y1, Y2, Y3 ? 13.3, 20, 20

X = 37.7408821           Y=15.7379135

Ready
    
```

Fig. 3.14.

b) Scrieți un program ABASIC care, folosind două declarații de tip **INPUT**, are ca rezultat:

```

b1) RUN
VALOAREA LUI X ? 8
VALOAREA LUI Y ? 6
X+Y=14
    
```

```

b2) RUN
VALOAREA LUI X ? 9
VALOAREA LUI Y ? 3
X-Y=6
    
```

```

b3) RUN
VALOAREA LUI X ? 3
VALOAREA LUI Y ? 7
X*Y=21
    
```

```

b4) RUN
VALOAREA LUI X ? 8
VALOAREA LUI Y ? 2
8/2=4
    
```

```

10 PRINT "VALOAREA LUI X";
20 INPUT X
30 PRINT "VALOAREA LUI Y";
40 INPUT Y
50 PRINT "X+Y"; X+Y
60 PRINT "X-Y"; X-Y
70 PRINT "X*Y"; X*Y
80 PRINT "X/Y"; X/Y
90 END
    
```

c) Care din programele de mai jos (c_1, c_2, c_3, c_4, c_5) calculează corect produsul dintre "A" și "B"?

```

c1) 10 PRINT "A=";
21 INPUT A
63 PRINT "B=";
69 INPUT B
93 P=A*B
411 PRINT "P="; P
600 STOP
    
```

```

c2) 8 PRINT "A=";
      33 INPUT A
      42 INPUT B
      59 P=A*B
      50 PRINT P
      60 STOP

c3) 20 INPUT A
      30 INPUT B
      25 P=A*B
      50 PRINT P
      60 STOP

c4) 50 PRINT "A=";
      200 INPUT A

```

```

      290 PRINT "B=";
      300 INPUT B
      210 INPUT P=A*B
      290 PRINT "P="; P
      320 STOP

c5) 50 PRINT "A="
      200 INPUT A
      140 PRINT "B=";
      150 INPUT B
      506 P=A*B
      1000 PRINT "P="; P
      1100 STOP

R. c1; c5

```

TEMA 3

Răspundeți prin DA sau NU la următoarele întrebări:

- În rezolvarea informatică a unei probleme, întotdeauna se pornește cu datele de ieșire.
- Tabela de variabile conține variabile de intrare, de stare și de ieșire.
- Este important ca schemele și specificațiile de programare să fie verificate împreună cu persoana care a solicitat programul BASIC.
- Diagrama de structură, pseudocodul și schema logică reprezintă principalele instrumente cu care se proiectează un program.
- Pseudocodul trebuie să fie precis spre a putea fi înțeles de programatori.
- Schema logică reprezintă o metodă grafică de documentare a programului.
- Instrucțiunea **REM** poate apare oriunde în cadrul unui program BASIC.
- În BASIC-aMIC se recomandă ca toate comentariile să se introducă între apostrofuri.
- Pe o linie multiinstrucțiune, instrucțiunea **REM** trebuie să fie ultima instrucțiune a liniei.
- În BASIC-PRAE, în locul lui **REM** se poate utiliza și apostroful.
- În BASIC HC-85, TIM S, SPECTRUM în locul lui **REM** se poate utiliza și apostroful.
- În BASIC-AMSTRAD, **REM** poate fi înlocuit printr-un apostrof.
- În BASIC-COMMODORE, **REM** nu poate fi înlocuit printr-un apostrof.
- În BASIC-80, comentariile pot fi precedate de un apostrof.
- În versiunea extinsă BASIC-80, comentariile se pun între ghilimele.
- În BASIC-PLUS, comentariile se introduc prin **REM** sau prin simbolul "!".
- În ABASIC, comentariile se introduc prin **REM** sau prin simbolul "!".
- Variabilele pot primi valori prin atribuire propriu-zisă.

- Variabilele pot primi valori prin introducere dinamică.
- Se recomandă utilizarea instrucțiunii **INPUT** ori de câte ori datele necesare execuției unui program sînt foarte numeroase.
- Ori de câte ori se execută instrucțiunea **INPUT**, calculatorul aMIC afișează pe ecran simbolul: " : ".
- Cu o instrucțiune **INPUT** se pot atribui una, două sau mai multe valori variabilelor din cadrul unui program BASIC.
- Instrucțiunea **PRINT TAB** se poate folosi într-un program BASIC-aMIC.
- Într-un program BASIC-PRAE pot fi utilizate instrucțiunile **PRINT TAB** și **PRINT SPC**.
- Instrucțiunea **PRINT TAB** se poate folosi într-un program BASIC HC-85, TIM S, SPECTRUM.
- Instrucțiunile **PRINT TAB** și **PRINT SPC** pot apare în orice program BASIC-COMMODORE.
- Instrucțiunea **ZONE** este specifică numai calculatoarelor personale AMSTRAD.
- Instrucțiunile **PRINT TAB** și **PRINT SPC** nu pot apare în programele BASIC-AMSTRAD.
- Instrucțiunile **PRINT TAB** și **PRINT SPC** pot apare în orice program BASIC-80.
- Instrucțiunea **PRINT TAB** poate fi folosită în orice program BASIC-PLUS.
- Instrucțiunea **PRINT SPC** poate fi folosită în orice program BASIC-PLUS.
- Instrucțiunile **PRINT TAB** și **PRINT SPC** pot apare în orice program ABASIC.
- Programele BASIC-aMIC pot fi salvate numai pe dischetă.
- În cadrul comenzii **SAVE** de salvare a unui program BASIC-aMIC pe casetă, numele programului este constituit din patru caractere hexazecimale.
- În cadrul comenzii **ASAVE** de salvare a unui program BASIC-PRAE pe casetă, numele programului este constituit din oricite caractere cuprinse între ghilimele.
- Programele BASIC HC-85, TIM S, SPECTRUM pot fi salvate numai pe dischetă.
- Programele BASIC-AMSTRAD pot fi salvate pe casetă și pe dischete de 3".
- Programele BASIC-COMMODORE pot fi salvate pe casetă și pe dischete de 5".

Înlocuiți cuvintele care lipsesc din următoarele propoziții:

- a) Procesul de alcătuire a unui program constă din următoarele faze: analiză, ,
- b) În faza de analiză se realizează tabela de variabile și
- c) În faza de proiectare se realizează

- d) Dintre virtuțile diagramei de structură reținem:
- e) Pseudocodul se pretează la documentația programării
- f) Acțiunile primitive sînt acțiunile care, spre a putea fi înțelese și executate de către calculator
- g) Acțiunile neprimitive sînt opusul acțiunilor
- h) Lista tuturor acțiunilor primitive constituie:
- problemei de rezolvat.
- i) În timpul codificării unui program au loc trei pași:
- j) În faza testare și depanare a unui program se constată
- k) Instrucțiunea **INPUT** permite utilizatorului să alocă diferite valori variabilelor, ori de cîte ori un program este executat, fără ca programul să fie modificat. Cînd calculatorul ajunge la instrucțiunea **INPUT** din cadrul unui program, afișează semnul întrebării (de regulă) și așteaptă ca utilizatorul să introducă valoarea ce trebuie alocată variabilei. Utilizatorul trebuie să introducă o valoare (după semnul întrebării) și apoi să acționeze tasta **[RETURN]**.

În fig. 3.15 se prezintă lista unui program BASIC și rezultatul execuției acestuia.

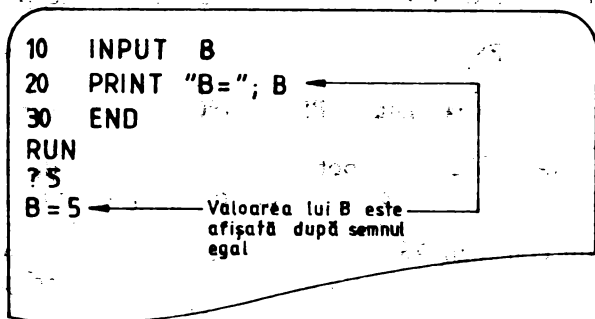


Fig. 3.15.

După ce am tastat **RUN** și am acționat tasta **[RETURN]**, calculatorul a afișat După aceea, am tastat, ceea ce reprezintă valoarea variabilei de intrare Calculatorul afișează apoi șirul de caractere "B=" urmat de lui B.

Programul poate fi din nou executat, dînd o valoare diferită variabilei B. Precizați modul de conversație cu calculatoarele: aMIC, PRAE, HC-85, TIM S, SPECTRUM, AMSTRAD, COMMODORE M-118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX-C dacă utilizatorul introduce pentru B valoarea 20?

RUN

.....

.....

l) Adăugați la programul precedent instrucțiunea:

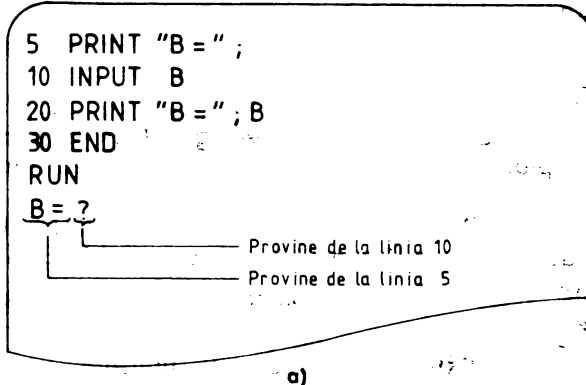
5 PRINT "B=";

Ați remarcat punctul și virgula de la sfârșitul instrucțiunii?

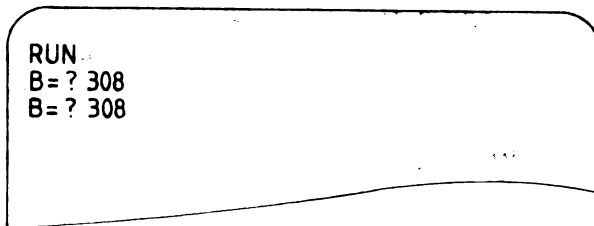
Când se folosește ";" la sfârșitul unei instrucțiuni **PRINT**

În fig. 3.16(a) se prezintă noul program și începutul execuției acestuia.

Acum știm sigur ce dorește calculatorul – o valoare pentru variabila de intrare B. Vom atribui lui B valoarea 308, o vom scrie după semnul întrebării și vom apăsa tasta **[RETURN]** (v. fig. 3.16(b)).



a)



b)

Fig. 3.16 a-b.

Precizați eroarea din fig. 3.16(b) cât și modul de conversație cu toată gama de calculatoare studiate, dacă utilizatorul introduce pentru B valoarea 902.

RUN

m) Când un program care conține o instrucțiune **INPUT** cu mai multe variabile este executat pe calculatoarele prima valoare afișată de utilizator (după semnul întrebării) este destinată variabile ce apare în instrucțiunea **INPUT**; a valoare tipărită de utilizator va fi destinată celei de-a doua variabile din instrucțiunea **INPUT** etc. Cele două variabile din programul care conține instrucțiunea **INPUT**, precum și valorile introduse de utilizator în timpul execuției programului, trebuie să fie separate de

n) Se consideră următorul program BASIC:

```

10 PRINT "VALORILE LUI X ȘI Y";
20 INPUT X, Y
30 PRINT "X="; X; Y="; Y
40 END

```

Dorim să atribuim lui X și Y valorile 3, respectiv 8. Pentru aceasta, tastăm după care

a) Ori de câte ori se execută instrucțiunea **INPUT**, calculatoarele , afișează pe ecran semnul întrebării.

p) În momentul execuției unei instrucțiuni **INPUT**, calculatorul așteaptă (răbdător!) pină tastați valoarea pe care doriți să o introduceți și pină cind veți acționa tasta

r) În instrucțiunea:

30 **INPUT**, x, y, z,

prima și ultima virgulă sînt plasate **corect/greșit**, în timp ce a doua și a treia virgulă sînt plasate **corect/incorect**. Virgula separă numai

s) Instrucțiunea **REM** poate fi înlocuită prin apostrof în BASIC , dar nu poate fi înlocuită prin apostrof în BASIC

t) În **BASIC** comentariile pot fi introduse prin **REM** și prin simbolul "!".

u) Instrucțiunea **PRINT TAB** poate fi utilizată în orice program BASIC dar nu poate fi folosită într-un program BASIC

v) Instrucțiunea **PRINT SPC** poate să apară în orice program BASIC dar nu poate fi utilizată într-un program BASIC

w) Programele BASIC-aMIC, BASIC-PRAE pot fi salvate pe dar nu pot fi salvate pe

x) Programele BASIC pot fi salvate și pe dischetă și pe casetă.

Să se scrie cite un program BASIC pentru fiecare din problemele de mai jos:

a) Să se calculeze momentele de inerție axiale ($I_Z=I_Y$) ale unei secțiuni circulare de rază R.

Indicație. Se utilizează formula $I_y=I_z=\frac{\pi d^4}{64}$, unde d este diametrul secțiunii. Valoarea lui d se va introduce dinamic.

b) Să se calculeze momentele de inerție axiale ale unei suprafețe inelare avînd diametrul exterior D și diametrul interior d.

Indicație. Se va utiliza formula de calcul $I_z=I_y=\frac{\pi}{64}(D^4-d^4)$. Valorile celor două diametre se vor introduce dinamic (exemplu: D=40 cm; d=20 cm).

c) Prin vîrfurile unui pătrat cu latura $a=10$ cm trec patru conductoare rectilinii, foarte lungi, perpendiculare pe planul figurii 3.17, parcurse de curent în sensul indicat pe figură. Dacă intensitățile curenților au valorile $I_1=1A$, $I_2=2A$, $I_3=1A$ să se determine inducția magnetică B_4 în vîrful patru al pătratului.

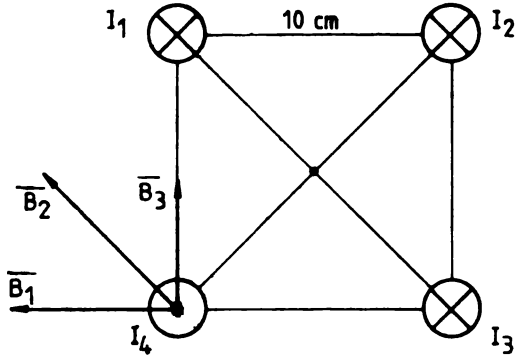


Fig. 3.17.

Indicație. Se va utiliza formula de calcul $B_4 = \frac{\mu_0}{2\pi a} (\sqrt{I_1^2 + I_3^2} + I_2/\sqrt{2})$. Valorile celor trei curenți și latura pătratului se vor introduce dinamic; $\mu_0 = 4\pi \cdot 10^{-7} \text{ N/A}^2$.

d) Scrieți un program BASIC care rezolvă următoarele sisteme liniare de trei ecuații cu trei necunoscute.

$\begin{aligned} &2x + 4y + z = 5 \\ \text{d}_1) \quad &x + y + z = 6 \\ &2x + 3y + z = 6 \end{aligned}$	$\begin{aligned} &x + 2y - z = -3 \\ \text{d}_2) \quad &3x + y + z = 4 \\ &x - y + 2z = 6 \end{aligned}$	$\begin{aligned} &x - y = 5 \\ \text{d}_3) \quad &y - z = -6 \\ &2x - z = 2 \end{aligned}$
--	--	--

Indicație. Pentru sistemul linear de trei ecuații cu trei necunoscute

$$\begin{aligned} a_1x + b_1y + c_1z &= k_1 \\ a_2x + b_2y + c_2z &= k_2 \\ a_3x + b_3y + c_3z &= k_3 \end{aligned}$$

x, y și z (necunoscutele) au expresiile:

$$\begin{aligned} x &= \frac{b_2c_3k_1 + b_1c_2k_3 + b_3c_1k_2 - b_2c_1k_3 - b_3c_2k_1 - b_1c_3k_2}{\Delta_3} \\ y &= \frac{a_1c_3k_2 + a_3c_2k_1 + a_2c_1k_3 - a_3c_1k_2 - a_1c_2k_3 - a_2c_3k_1}{\Delta_3} \\ z &= \frac{a_1b_2k_3 + a_3b_1k_2 + a_2b_3k_1 - a_3b_2k_1 - a_1b_3k_2 - a_2b_1k_3}{\Delta_3} \end{aligned}$$

unde:

$$\Delta_3 = a_1b_2c_3 + a_3b_1c_2 + a_2b_3c_1 - a_3b_2c_1 - a_1b_3c_2 - a_2b_1c_3$$

e) Asupra unui punct material acționează un sistem de forțe alcătuit din cinci forțe concurente (vezi fig. 3.18). Să se determine rezultanta acestui sistem de forțe.

Indicație. Se va utiliza relația de calcul $\vec{R} = \sqrt{R_x^2 + R_y^2}$, unde $R_x = F_{1x} + F_{2x} + \dots + F_{5x}$ iar $R_y = F_{1y} + F_{2y} + \dots + F_{5y}$. R_x și R_y reprezintă suma algebrică a proiecțiilor forțelor pe axele Ox respectiv Oy . Valorile lui F_{1x}, \dots, F_{5x} și ale lui F_{1y}, \dots, F_{5y} se vor introduce dinamic.

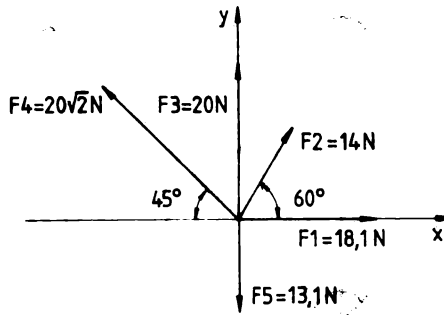


Fig. 3.18.

f) Un mobil efectuează o mișcare oscilatorie armonică. Știind că pentru elongațiile $x_1=2$ cm și $x_2=3$ cm, mobilul are vitezele $v_1=5$ m·s⁻¹ și respectiv $v_2=4$ m·s⁻¹, să se calculeze amplitudinea și perioada mișcării oscilatorii a mobilului.

Indicație. Se vor utiliza formulele de calcul:

$$A = \sqrt{\frac{x_1^2 v_2^2 - x_2^2 v_1^2}{v_2^2 - v_1^2}}; \quad T = 2\pi \sqrt{\frac{x_1^2 - x_2^2}{v_2^2 - v_1^2}}$$

Valorile vitezelor și elongațiilor se vor introduce dinamic.

Specificațiile de programare și documentația de proiectare sînt ilustrate în modulele de analiză și proiectare structurată; fig. 3.19.

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieșire
V1 – viteza corespunzătoare elongației X1	C : diferența pătratelor vitezelor	A : amplitudinea mișcării oscilatorii
V2 – viteza corespunzătoare elongației X2		T : perioada mișcării oscilatorii
X1 : prima elongație		
X2 : a doua elongație		

a

SPECIFICAȚII DE PROGRAMARE

Descrierea programului

Programul citește de la terminal elongațiile mișcării oscilatorii și vitezele mobilului, calculează amplitudinea și perioada mișcării oscilatorii și afișează rezultatul.

Intrări

Două elongații și două viteze.

Ieșiri

Perioada și amplitudinea

Lista de funcțiuni ale programului

1. Citește date intrare
2. Calculează amplitudinea mișcării oscilatorii
3. Calculează perioada mișcării oscilatorii
4. Afișează rezultatul
5. Stop.

Fig. 3.19. a–g. Modulele de analiză și proiectare structurată: a) tabela de variabile; b) specificații de programare;

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

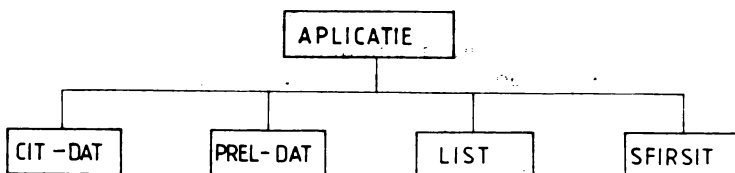
Modul	Funcțiuni
CIT-DAT	1
PREL-DAT	2,3
LIST	4
SFIRȘIT	5

c)

DATELE DE INTRARE

X1	X2	V1	V2
2	3	5	4

d)



e)

PSEUDOCOCUL

```

APLICAȚIE          SEQ
CIT-DAT           SEQ
                    INPUT X1, X2
                    INPUT V1, V2

CIT-DAT           END
PREL-DAT         SEQ
                    C=V2 * V2-V1 * V1
                    A=((X1 ↑ 2 * V2 ↑ 2-X2 ↑ 2 * V1 ↑ 2)/C) ↑ .5
                    T=2 * PI * ((X1 ↑ 2-X2 ↑ 2)/C) ↑ .5

PREL-DAT         END
                    PRINT "A="; A, "T="; T

APLICAȚIE         END
    
```

f)

Fig. 3.19 c) alocarea funcțiilor de prelucrare; d) datele de intrare; e) diagrama de structură; f) pseudocodul;

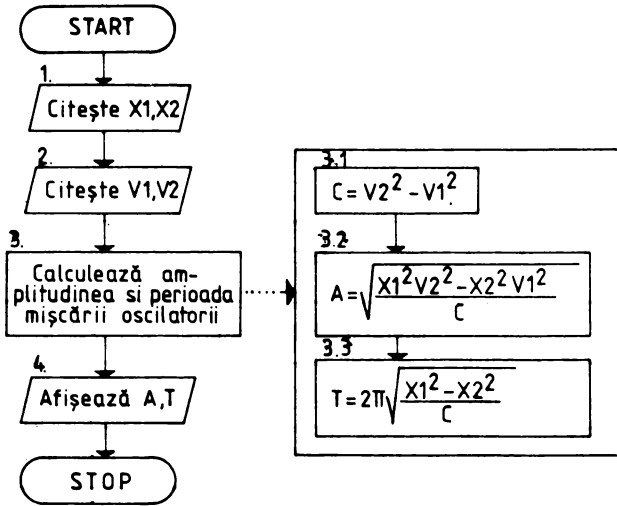


Fig. 3.19

g) schema logică.

g) Se consideră placa plană omogenă din figura 3.20, avînd dimensiunile date în centimetri. Să se determine coordonatele centrului de greutate.

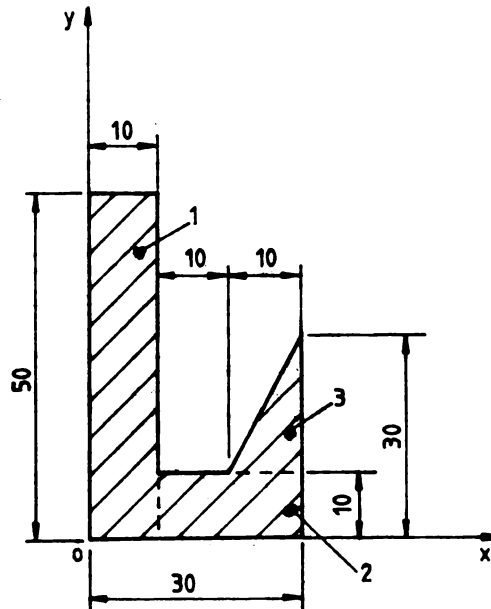


Fig. 3.20.

Indicație. Se împarte placa în trei arii elementare și se alege sistemul de axe din figură.

Datele de intrare sint prezentate în tabelul 3.13.

Tabelul 3.13

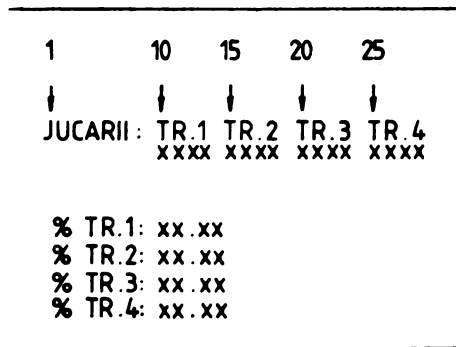
DATELE DE INTRARE								
A1	A2	A3	X1	X2	X3	Y1	Y2	Y3
500	200	100	5	20	26,666*	25	5	13,333**

* $20 + \frac{2}{3} \cdot 10$

** $10 + \frac{1}{3} \cdot 10$

□ **Să se scrie un program BASIC care citește de la terminal cantitățile lunare de jucării (pentru un articol) fabricate de întreprinderea "URSULEȚUL", calculează cantitățile totale trimestriale și procente corespunzătoare din cantitatea totală anuală.**

Specificațiile de programare și documentația de proiectare sint ilustrate în modulele de analiză și proiectare structurată, fig. 3.21.



a)

DATELE DE INTRARE*											
A1	A2	A3	B1	B2	B3	C1	C2	C3	D1	D2	D3
10	20	30	5	6	7	15	20	25	30	5	8

* mii bucăți

b)

Fig. 3.21 a–h. Modulele de analiză și proiectare structurată: a) formatul datelor de ieșire; b) datele de intrare;

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieșire
A1 : livrări ianuarie	T1 : livrări trim. I	P1 : procent trim. I
A2 : livrări februarie	T2 : livrări trim. II	P2 : procent trim. II
A3 : livrări martie	T3 : livrări trim. III	P3 : procent trim. III
B1 : livrări aprilie	T4 : livrări trim. IV	P4 : procent trim. IV
B2 : livrări mai	T : total livrări	
B3 : livrări iunie		
C1 : livrări iulie		
C2 : livrări august		
C3 : livrări septembrie		
D1 : livrări octombrie		
D2 : livrări noiembrie		
D3 : livrări decembrie		

c)

SPECIFICAȚII DE PROGRAMARE

Descrierea programului

Programul citește de la terminal cantitățile lunare de jucării fabricate, calculează cantitățile totale trimestriale și procente corespunzătoare din cantitatea totală anuală și afișează rezultatele.

Intrări

Cantitățile lunare de jucării fabricate.

Ieșiri

Cantitățile trimestriale de jucării fabricate, exprimate în procente.

Lista de funcțiuni ale programului

1. Citește date intrare
2. Calculează cantitățile totale trimestriale.
3. Calculează cantitatea totală anuală exprimată în procente
4. Calculează cantitățile trimestriale de jucării fabricate
5. Afișează rezultatul
6. Stop

d)

ALOCAREA FUNCȚIUNILR DE PRELUCRARE

Modul	Funcțiuni
CIT-DAT	1
PREL-DAT	2, 3, 4
LIST	5
SFIRȘIT	6

e)

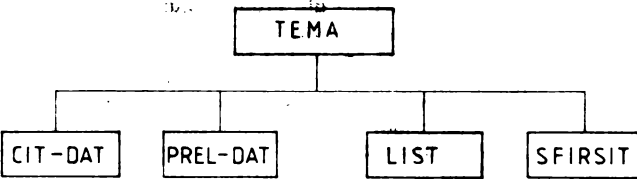
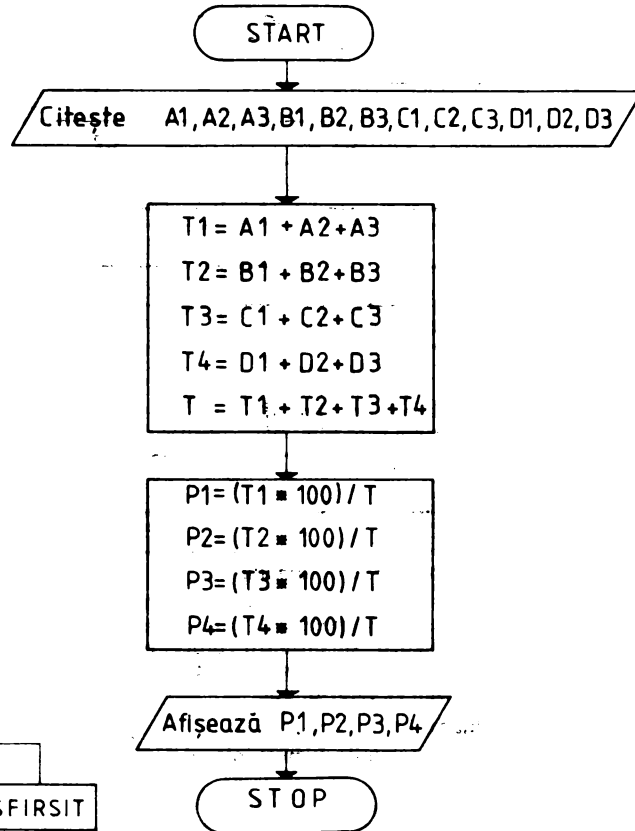
c) tabela de variabile; d) specificații de programare; e) alocarea funcțiunilor de prelucrare;

PSEUDOCODUL

```

TEMA3-4      SEQ
PREL-DAT     INPUT A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3
              SEQ
              T1=A1+A2+A3
              T2=B1+B2+B3
              T3=C1+C2+C3
              T4=D1+D2+D3
              T =T1+T2+T3+T4
              P1=(T1 * 100)/T
              P2=(T2 * 100)/T
              P3=(T3 * 100)/T
              P4=(T4 * 100)/T
PREL-DAT     END
TEMA3-4      PRINT P1, P2, P3, P4
              END
    
```

f)



g)

h)

f) pseudocodul; g) diagrama de structură; h) schema logică.

□ Să se scrie un program BASIC care citește de la terminal consumurile trimestriale ale unui articol, calculează și afișează procente corespunzătoare consumurilor trimestriale din consumul total anual (vezi și L. Dumitrașcu, Învățăm COBOL... conversind cu calculatorul, Editura Tehnică, București, 1985).

Specificațiile de programare și documentația de proiectare sînt ilustrate în modulele de analiză și proiectare structurată, fig. 3.22.

1	12	17	22	27
↓	↓	↓	↓	↓
CONSUMURI	TR.1	TR.2	TR.3	TR.4
	XXXX	XXXX	XXXX	XXXX
	% TR.1 :	xx .xxx		
	% TR.2 :	xx .xxx		
	% TR.3 :	xx .xxx		
	% TR.4 :	xx .xxx		

a)

DATE DE INTRARE			
C1	C2	C3	C4
321	432	766	876

b)

TABELA DE VARIABILE		
Variabile de intrare	Variabile de stare	Variabile de ieșire
C1 : consum trim. I	T : consum total	P1 : procent trim. I
C2 : consum trim. II		P2 : procent trim. II
C3 : consum trim. III		P3 : procent trim. III
C4 : consum trim. IV		P4 : procent trim. IV

c)

SPECIFICAȚII DE PROGRAMARE	
<p>Descrierea programului</p> <p>Programul citește de la terminal consumurile trimestriale ale unui articol, calculează și afișează procente corespunzătoare consumurilor trimestriale din consumul total anual.</p> <p>Intrări</p> <p>Consumurile trimestriale ale unui articol.</p>	<p>Ieșiri</p> <p>Consumurile trimestriale exprimate în procente față de consumul total anual.</p> <p>Lista de funcțiuni ale programului</p> <ol style="list-style-type: none"> 1. Citește date intrare 2. Calculează cantitatea totală anuală 3. Calculează procente corespunzătoare consumurilor trimestriale din consumul total anual 4. Afișează rezultatul 5. Stop

d)

Fig. 3.22 a-h. Modulele de analiză și proiectare structurată: a) formatul datelor de ieșire; b) datele de intrare; c) tabela de variabile; d) specificații de programare;

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

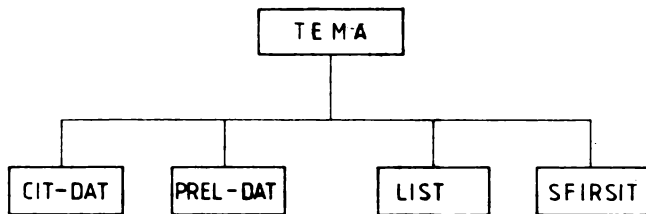
Modul	Funcțiuni
CIT-DAT	1
PREL-DAT	2,3
LIST	4
SFIRȘIT	5

e)

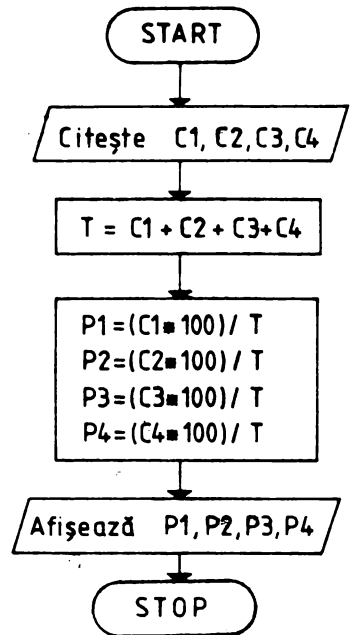
PSEUDOCODUL

TEMA3-5	SEQ
	INPUT C1, C2, C3, C4
PREL-DAT	SEQ
	$T = C1 + C2 + C3 + C4$
	$P1 = (C1 * 100) / T$
	$P2 = (C2 * 100) / T$
	$P3 = (C3 * 100) / T$
	$P4 = (C4 * 100) / T$
PREL-DAT	END
	PRINT P1, P2, P3, P4
TEMA3-5	END

f)



g)



h)

Fig. 3.22. e) alocarea funcțiilor de prelucrare; f) pseudocodul; g) diagrama de structură; h) schema logică.

SOLUȚIA TEMEI 3

□ Răspunsurile sînt:

DA, DA, DA, DA, DA, DA, DA, DA, DA, DA, DA, NU, DA, DA, DA, DA, DA, DA, DA, DA, NU, DA, DA, NU, DA, DA, DA, DA, NU, DA, DA, NU, DA, NU, DA, DA, NU, DA, DA.

□ Cuvintele lipsă sînt:

a) analiză, proiectare, codificare, testare și depanare; **b)** specificațiile de programare; **c)** diagrama de structură, pseudocodul, schema logică; **d)** este independentă de limbaj sau calculator, poate reda o structură a datelor sau a logicii de prelucrare, scoate în evidență structura funcțională a programului, evidențiază necesarul de operații de intrare/ieșire; **e)** structurate; **f)** nu mai necesită informații suplimentare; **g)** primitive; **h)** algoritmul; **i)** scrierea programului (pasul 1), introducerea programului (pasul 2), execuția programului (pasul 3); **j)** dacă programul realizat este în concordanță cu specificațiile de programare; **k)** ?, 5, B, valoarea; **l)** valoarea pe care o introduceți apare pe aceeași linie cu semnul întrebării; **m)** PRAE, HC-85, TIM S, SPECTRUM, M 118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX C, primei variabile, a doua valoare, virgulă; **n)** RUN, [RETURN], tastă 3, virgulă, 8, [RETURN]; **o)** vezi m; **p)** [RETURN]; **r)** greșit, corect, variabilele; **s)** PRAE, AMSTRAD, BASIC-80; HC-85, TIM S, SPECTRUM, COMMODORE, BASIC-PLUS, ABASIC; **t)** BASIC-PLUS, ABASIC; **u)** PRAE, HC-85, TIM S, SPECTRUM, AMSTRAD, COMMODORE, BASIC-80, BASIC-PLUS, ABASIC; **aMIC**; **v)** PRAE, COMMODORE, AMSTRAD, BASIC-80, ABASIC; BASIC aMIC, BASIC-PLUS; **w)** casetă magnetică; dischetă; **z)** COMMODORE, AMSTRAD.

□ Programele BASIC sînt:

a)

```
10 PRINT "RAZA=";
20 INPUT R
25 D=2 * R
30 IY=PI * D ↑ 4/64
40 PRINT "IZ=IY."; IY
50 END
```

```
run
RAZA=? 5
IZ=IY=490.873852
```

ready

```
edit 40
40 INPUT dI
edit 50
50 iz=(PI/64) * (D ↑ 4-dI ↑ 4)
```

run

```
D=? 40
d=? 20
IZ=IY=117809.725
ready
```

b)

```
10 PRINT "D=";
20 input D
30 print "d=";
40 input d
50 iz=(pi/64) * (D ↑ 4-d ↑ 4)
60 PRINT "IZ=IY="; iz
run
D=? 40
d=? 20
IZ=IY=0
```

c)

```
5 min=4 * PI/10 ↑ 7
10 print "a";
20 input a
30 print "I1, I2, I3";
40 input i1, i2, i3
50 b4=(min/(2 * pi * a)) *
  * ((i1 ↑ 2+i3 ↑ 2) ↑ 0.5+i2/2 ↑ 0.5)
55 print
60 print "B4="; b4
```

```
run
a ? · 1
l1, l2, l3 ? 1, 2, 1
B4=5-65685E-06
Ready
```

```
d) 10 print "a1, b1, c1, k1";
    20 input a1, b1, c1, k1
    30 print "a2, b2, c2, k2";
    40 input a2, b2, c2, k2
    50 print "a3, b3, c3, k3";
    60 input a3, b3, c3, k3
    70 d=a1*b2*c3+
    a3*b1*c2+a2*b3*c1-
    a3*b2*c1-a1*b3*c2-a2*b1*c3
    80 d1=b2*c3*k1+b1*c2*k3+
    b3*c1*k2-b1*c1*k3-b3*c2*k1-
    b1*c3*k2
    90 d2=a1*c3*k1+a3*c2*k1+
    a2*c1*k3-a3*c1*k2-a1*c2*k3-
    a2*c3*k1
    100 d3=a1*b2*k3+a3*b1*k2+
    a2*b3*k1-a3*b2*k1-a1*b3*k2-
    a2*b1*k3
    110 x=d1/d; y=d2/d; z=d3/d
    115 print
    120 print "X="; x; "Y="; y; "Z=";
    z
```

```
d1) run
a1, b1, c1, k1 ?
a2, b2, c2, k2 ?
a3, b3, c3, k3 ?
X=2 Y=-1 Z=5
Ready
```

```
d2) run
a1, b1, c1, k1 ?
a2, b2, c2, k2 ?
a3, b3, c3, k3 ?
X=1 Y=-1 Z=2
Ready
```

```
d3) run
a1, b1, c1, k1 ?
a2, b2, c2, k2 ?
a3, b3, c3, k3 ?
X=3 Y=-2 Z=4
Ready
```

```
e) 10 print "F1x, F2x, F3x, F4x, F5x";
    20 input f1x, f2x, f3x, f4x, f5x
    30 print "F1Y, F2Y, F3Y, F4Y, F5Y";
    40 input f1y, f2y, f3y, f4y, f5y
    50 rx=f1x+f2x+f3x+f4x+f5x
    60 ry=f1y+f2y+f3y+f4y+f5y
    70 r=(rx↑2+ry↑2)↑.5
    75 print
    80 print "R="; r; "N"
    90 end
run
F1x, F2x, F3x, F4x, F5x ?
18.1, 7, 0, -20, 0
F1Y, F2Y, F3Y, F4Y, F5Y ?
0,12,11,20, 20, -13.1
R=39.3419636 N
Ready
```

```
f) 5 input x1, x2, y1, y2
    10 c=V2↑2-V1↑2
    20 a=((x1↑2 * V2↑2-
    x2↑2 * V1↑2)/C)↑.5
    30 t=2 * pi * ((x1↑2-x2↑2)/C)↑.5
    40 print "A="; a; "T="; t
run
? 2, 3, 5, 4
Division by zero
Improper argument in 20
Ready
edit 5
5 INPUT X1, X2, V1; V2
run
? 2, 3, 5, 4
A=4.22952585 T=4.68320982
Ready
```

Observație. De notat (vezi f) apariția mesajelor de eroare:

Division by zero
Improper argument in 20

ce se datorează tastării greșite (y1, y2) a numelui celor două variabile de intrare V1 și V2. Cu edit 5 s-a corectat greșeala, rezultând în final valorile afișate, corespunzătoare variabilelor (de ieșire) a și t.

```
g) 5 print "A1, A2, A3";
    10 input a1, a2, a3
    20 print "X1, X2, X3";
    30 input x1, x2, x3
    40 print "Y1, Y2, Y3";
    50 input y1, y2, y3
```

```
run
A1, A2, A3 ? 500, 200, 100
X1, X2, X3 ? 5, 20, 26.60
Y1, Y2, Y3 ? 25, 5, 16.66
X=11.4583333 Y=18.9583334
Ready
```

```

60 a=a1+a2+a3
70 x=(a1 * x1+a2 * x2+a3 * x3)/a
80 y=(a1 * y1+a2 * y2+a3 * y3)/a
85 print : print
90 print "X="; x, "Y="; y
100 end

```

- Programul pentru calculul procentelor corespunzătoare cantităților trimestriale de jucării, fabricate de întreprinderea Ursulețul este:

```

5 print "a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3"
10 input a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3
20 t1=a1+a2+a3; t2=b1+b2+b3; t3=c1+c2+c3
30 t4=d1+d2+d3; t=t1+t2+t3+t4
40 p1=t1 * 100/t; p2=t2 * 100/t; p3=t3 * 100/t
50 p4=t4 * 100/100
60 cls
70 print "JUCĂRII:"; tab (10); "TR. 1"; tab (15); "TR. 2"; tab (20); "
"TR. 3"; tab (25); "TR. 4"
80 print tab (10); t1; tab (15); t2; tab (20); t3; tab (25); t4
85 print; print
90 print tab (2); "% TR. 1:"; p1
100 print tab (2); "% TR. 2:"; p2
110 print tab (2); "% TR. 3:"; p3
120 print tab (2); "% TR. 4:"; p4
130 end

```

Să executăm programul și să analizăm împreună rezultatele (vezi fig. 3.23a).

Eliminarea erorilor de sintaxă

Remarcați pentru început apariția mesajului de eroare:

Syntax error in 20

care ne spune că în linia 20 s-a strecurat o greșeală. Ați găsit-o? Greșeala constă în faptul că s-a utilizat ca separator ";" în loc de ":", ceea ce contravine regulilor de sintaxă prezentate. Se va înlocui așadar simbolul ";" cu ":" în cele două apariții de pe linia 20.

Vom relua execuția programului începând cu linia 20 (vezi fig. 3.23(b)).

Încă o eroare de sintaxă! De astă dată în linia 30. Interesant cât de constanți am putut fi săvârșind aceeași greșeală! Ca și în cazul precedent, se va înlocui simbolul ";" cu ":".

Reluați execuția programului începând cu linia 30 (vezi fig. 3.23(c)).

Chiar să fim urmăriți de ghinion? Nu se poate! De această dată și-au făcut apariția două mesaje:

Division by zero

Syntax error in 40

dintre care primul pare chiar foarte ciudat. Ce s-a întâmplat! Tot corectînd programul am uitat să reluăm execuția programului începînd cu linia 5 și nu cu liniile 20, respectiv 30, cum greșit am procedat. Executînd programul de la linia 30, era normal ca t să fie 0 (numitorul din instrucțiunile de atribuire de la linia 40), deoarece și valorile variabilelor de intrare erau


```
run
a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3
? 10, 20, 30, 5, 6, 7, 15, 20, 25, 30, 5, 8
Syntax error in 20
20 t1 = a1 + a2 + a3; t2 = b1 + b2 + b3; t3 = c1 + c2 + c3
```

a)

```
run 20
Syntax error in 30
30 t4 = d1 + d2 + d3; t = t1 + t2 + t3 + t4
```

b)

```
run 30
Division by zero
Syntax error in 40
40 p1 = t1 * 100 / t; p2 = t2 * 100 / t; p3 = t3 * 100 / t
```

c)

```
run
JUCARII: TR.1 TR.2 TR.3 TR.4
          60  18  60  43
/
% TR.1: 33.1491713
% TR.2: 9.94475138
% TR.3: 33.1491713
% TR.4: 43.1491713
Ready
■
```

d)

Fig. 3.23 a-d

```

run
JUCARII: TR.1 TR.2 TR.3 TR.4
          60   18   60   43

% TR.1: 33.1491713
% TR.2:  9.94475138
% TR.3: 33.1491713
% TR.4: 23.7569061

Ready
■

```

Fig. 3.23 e)

zero (neexecutându-se instrucțiunile 5 și 10). În acest caz, fără să vrem, am făcut o greșală de logică!

În consecință, tastați:

run [RETURN]

nu înainte însă de a corecta aceeași greșală de sintaxă care ne-a urmărit pe tot parcursul execuției programului din linia 40.

În fig. 3.23 (d) se prezintă rezultatul final al execuției programului.

Foarte mulți dintre dvs. ar fi tentați să se declare mulțumiți de răspunsul calculatorului manifestând o grabă sau chiar, în cele mai multe cazuri, o satisfacție, motivind că programul „a mers” iar lista cu rezultate este încântătoare! Desigur, rezultatele sînt tipărite frumos, dar, știți dvs. cum este cu frumusețea!

Așadar, vă rugăm să rămînem în continuare asupra listei cu rezultate, propunîndu-vă să faceți un calcul simplu: adunați cele patru procente! Ați obținut 100? Nu?! Care credeți că este cauza? Am făcut pe undeva vreo greșală de logică? Încercați s-o depistați singuri.

Eliminarea unei erori de logică

Există mai multe tehnici de depistare a unei erori de logică. Într-un fel, ele țin și de practica pe care o aveți dvs. în programarea calculatoarelor. În ceea ce ne privește, pentru problema în cauză vă propunem să parcurgeți următorii pași: a) lăcțurați încă o dată problema; b) revedeți specificațiile de programare și documentația de proiectare; c) revedeți lista programului sursă BASIC, linie cu linie.

Ați găsit greșeala? Nu vă descurajați atît de ușor. Sinteți de-abia la început de drum!

La o primă impresie, se pare că eroarea ar proveni din algoritmul de calcul al valorilor lui p_1 , p_2 , p_3 și p_4 . Verificați în program relațiile de calcul (vezi liniile 40 și 50).

$$40 \quad p_1 = t_1 * 100 / t \quad ; \quad p_2 = t_2 * 100 / t \quad ; \quad p_3 = t_3 * 100 / t$$

$$50 \quad p_4 = t_4 * 100 / 100$$

Mulți dintre dvs. ar putea afirma că nici aici (liniile 40 și 50) nu există greșală, furati poate de faptul că expresiile lui p1, p2 și p3 sînt corecte. Dar p4? Tocmai aici s-a strecurat greșeala. Din neatenție, în loc să tastăm (la numitor) t,

$$50 \text{ p4} = t4 * 100 / 100$$

am introdus 100, ceea ce reprezintă cu totul altceva!

În consecință, corectați linia 50 (tastați **edit 50**) de forma:

$$50 \text{ p4} = t4 * 100 / t$$

tastați:

run [RETURN]

pregătindu-vă de o nouă conversație (vezi fig. 3.23(e)), sperăm și ultima pentru această problemă.

Remarci

- Valorile variabilelor de intrare au fost șterse de pe ecran (vezi instrucțiunea **cls** din linia 60), înainte de afișarea capului de tabel și a rezultatului final;
- Introduceți și executați programul și pe calculatoarele aMIC (nu folosiți liniile multiinstrucțiune), PRAE, HC-85, TIM S, SPECTRUM, COMMODORE, AMSTRAD, M 118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX C, calculatorul dvs., cu recomandarea de a introduce în linia 40 instrucțiunea corespunzătoare, după cum urmează: **INIT** (aMIC), **CLS** (PRAE, HC-85, TIM S SPECTRUM), **PRINT CHR\$(147)**, (COMMODORE), **CHR\$(24)** (M 118, TPD, JUNIOR).

Programul pentru calculul procentelor consumurilor trimestriale ale unui articol este:

```

5 print c1 : print c2
10 input c1, c2, c3, c4
20 l=c1+c2+c3+c4:
   p1=c1 * 100/t: p2=c2 * 100/t
30 p3=c3 * 100/t : p4=c4 * 100/t
40 cls
50 print "CONSUMURI:";
   tab (12); "TR. 1"; tab (17); "TR. 2";
   tab (22); "TR. 3"; tab (27); "TR. 4
60 print tab (12); c1; tab (17); c2;
   tab (22); c3; tab (27); c4
70 print : print
80 print tab (4); "% TR. 1:"; p1
90 print tab (4); "% TR. 2:"; p2
100 print tab (4); "% TR. 3:"; p3
110 print tab (4); "% TR. 4:"; p4
120 end

```

Introduceți și executați programul.

Cum vă explicați rezultatele pe care le-ați obținut?

Să începem cu apariția celor două zerouri. Ele sînt rezultatul execuției instrucțiunii 5:

```
5 print c1 : print c2
```

care, așa cum a fost scrisă, este inutilă (c1, c2 au, pentru început, valoarea zero). Corectați instrucțiunea de forma:

```
5 print "c1, c2, c3, c4"
```

Să mergem mai departe. Deși ați introdus valori pentru variabilele c1, c2, c3 și c4, calculatorul v-a semnalat de patru ori același mesaj:

Division by zero.

```

run
0
0
? 321,432,766,876
Division by zero
Division by zero
Division by zero
Division by zero

```

a)

```

CONSUMURI : TR.1 TR.2 TR.3 TR.4
           321  432  766  876

% TR.1 : 13.4029228
% TR.2 : 18.0375783
% TR.3 : 31.9832985
% TR.4 : 36.5762004
Ready
■

```

b)

Fig. 3.24 a-b.

Credem că ați învățat din greșelile conversației precedente și, ca urmare nu v-a fost greu să observați eroarea din linia 20:

$$20 \quad l = c_1 + c_2 + c_3 + c_4 : p_1 = c_1 * 100/t : p_2 = c_2 * 100/t$$

în loc de $l = \dots$ trebuie introdus:

$$20 \quad t = c_1 + c_2 + c_3 + c_4 : \dots$$

Odată operate aceste două corecții, încercați o nouă execuție a programului. În fig. 3.24 (b) se prezintă rezultatul corect al execuției acestui program. Nu uitați să verificați ... calculatorul!

Observație. Introduceți și executați programul și pe calculatoarele aMIC (nu folosiți liniile multiinstrucțiune), PRAE, HC-85, TIM-S, SPECTRUM, COMMODORE, AMSTRAD, M-118, TPD, JUNIOR, INDEPENDENT, CORAL, FELIX-C, cu recomandarea de a introduce în linia 40 instrucțiunea corespunzătoare fiecărui calculator în parte.

CONVERSAȚIA 4

Proiectarea și realizarea unui program pentru calculul cantității de lichid din nouă rezervoare cilindrice echilaterale a căror rază variază de la 2 la 10 m, cu pasul de un metru. Constante și variabile alfanumerice. Structura de iterație PENTRU. Bucle FOR—NEXT. „Bunicul” lui FOR—NEXT. Editarea valorilor numerice. Instrucțiunea PRINT USING. Modificatorul FOR. APLICAȚII și TESTE pentru cititor



EXEMPLELE 4 – PC, m, M, F

De la problemă la program

Vom relua problema precedentă considerind de această dată mai multe rezervoare cilindrice echilaterale (generatoarea este egală cu diametrul), toate pline cu benzină! Raza acestora variază de la 2 la 10 m cu pasul 1 (2 m, 3 m, . . . ,10 m). Problema pe care ne-o propunem să o rezolvăm acum constă în a calcula și afișa cantitatea de benzină în tone (masa) din fiecare rezervor. De asemenea, în final se dorește să se afișeze cantitatea totală de benzină. Densitatea benzinei ($0,7 \text{ g/cm}^3$) se va introduce în momentul execuției programului. Cum s-ar putea rezolva această problemă?

Cei mai grăbiți dintre dvs. se și gîndesc deja la instrucțiunile BASIC necesare alcătuirii programului. Cîteva **INPUT**-uri pentru citirea datelor de intrare, cîteva instrucțiuni de atribuire necesare calculului cantității de benzină din fiecare rezervor, o instrucțiune care să însumeze cantitatea de benzină pe care o conțin cele 9 rezervoare și cîteva **PRINT**-uri! Banal, vor exclama poate mulți dintre aceștia. Al vor întîmpina poate o „singură” dificultate privind formula densității unui corp. Le-o vom aminti noi:

$$\rho = \frac{m}{V} \quad (1)$$

unde ρ reprezintă densitatea, m masa corpului, iar V volumul cilindrului (aria bazei înmulțită cu generatoarea).

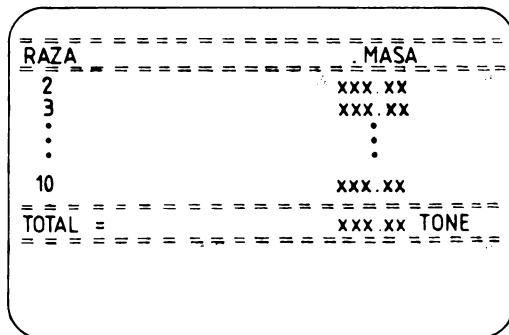
În ceea ce ne privește, nu recomandăm rezolvarea informatică a problemei propuse în acest mod defectuos în care gîndesc foarte mulți debutanți. A gîndi în limbaj informatic (BASIC) încă de la formularea problemei ni se pare o concepție total greșită. Rezolvarea în grabă a problemei este urmată de cele mai multe ori de surprize – evident neplăcute. Spre a le evita vă propunem să străbatem împreună *lungul drum al problemei către program*, prin parcurgerea tuturor fazelor evidențiate în metodica de analiză, proiectare și realizare a programelor.

Analiza problemei

Date de ieșire, intrare și stare

Și în rezolvarea acestei probleme pornim cu datele de ieșire, urmînd să ne întrebăm apoi dacă avem suficiente date de intrare pentru a putea fi obținute ieșirile dorite. Se urmărește ca raportul de ieșire să aibă formatul descris în modulul de analiză structurată, fig. 4.1 (a) în care, rîndul de titlu (RAZA, MASA) este urmat de valorile razelor (2, 3, . . . ,10m) rezervoarelor (cilindrice echilaterale) și de cantitățile de benzină corespunzătoare (calculate prin program) și în final de un rînd de TOTAL în care se afișează cantitatea de benzină din cele 9 rezervoare.

Datele de intrare cuprind numai densitatea benzinei care se va citi (dinamic) în timpul execuției programului. Este momentul să ne întrebăm dacă datele de ieșire (M, S) și de intrare (D) sînt suficiente pentru rezolvarea problemei! Să nu ne grăbim cu răspunsul și să analizăm cu mai mare atenție formatul datelor de ieșire (vezi fig. 4.1(a)). Vă propunem să ne oprim chiar asupra primului element al listei: RAZA – care ia valori într-un domeniu cunoscut [2, 10] m. Așadar, nu ne rămîne decît să-i asociem acestei date un nume de variabilă (R) pe care să o definim ca o *variabilă de stare*.



a)

TABELA DE VARIABLE

Nume program: EXEMPLELE 4 – PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieșire
D: densitatea benzinei	V: volumul rezervorului cilindric echilateral R: raza rezervorului M: masa de benzină	S: masa totală de benzină M: masa de benzină

b)

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 4 – PC, m, M, F

Descriere. Programul calculează și afișează cantitatea de benzină din mai multe rezervoare cilindrice echilaterale (generatoarea este egală cu diametrul) a căror rază variază de la 2 la 10 metri cu pasul de un metru.

Intrări. Se citește dinamic densitatea benzinei (0,7 g/cm³).

Ieșiri. Masa de benzină din fiecare rezervor și masa totală de benzină.

Lista de funcțiuni ale programului

- | | |
|--|---|
| 1. Citește densitate benzină | 7. Afișează raza (R) și masa rezervorului (M) |
| 2. Inițializează cu zero variabila S | 8. Inițializează variabila de control a ciclului |
| 3. Tipărește cap tabel | 9. Incrementează și testează variabila de control |
| 4. Calculează volumul rezervorului | 10. Afișează masa totală de benzină |
| 5. Calculează masa de benzină (M) din fiecare rezervor | 11. Stop |
| 6. Insumează M în S | |

c)

Fig. 4.1. a–d. Modulul de analiză structurată. a) formatul datelor de ieșire; b) tabela de variabile; c) specificații de programare;

TABELA DE VARIABILE

Nume program: EXEMPLELE 4 – PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieșire
S: masa totală de benzină		

d)

d) tabela de variabile extinsă.

În tabela de variabile a EXEMPLELOR 4 – PC, m, M, F s-au definit încă două variabile de stare: V, M a căror semnificație este redată în modulul de analiză structurată, fig. 4.1 (b).

Descrierea programului, precum și lista principalelor funcțiuni sînt prezentate în modulul de analiză structurată, fig. 4.1 (c).

Observație. În cadrul acestei conversații, prelucrările se fac nu pentru un rezervor (articol), ci pentru mai multe (9) rezervoare (articole)

Asigurîndu-ne că nu s-au strecurat greșeli în faza de analiză a problemei, putem aborda în continuare faza de proiectare a programului urmînd să elaborăm în cele ce urmează: diagrama de structură, pseudocodul și schema logică.

□ Proiectarea programului

Proiectarea diagramei de structură

În vederea trasării diagramei de structură vom proceda mai întîi la alocarea funcțiunilor de prelucrare (vezi modulul de analiză structurată fig. 4.1 (c)) unor module, după cum urmează: CAP–TAB, INIT–CIT, CALCUL, EDIT–RIND, TOTAL, SFIRȘIT (vezi modulul de proiectare structurată, fig. 4.2 (a)).

Modulele CAP–TAB, INIT–CIT, TOTAL, SFIRȘIT împreună cu modulul PRELUCRĂRI, pe care l-am introdus în mod artificial, alcătuiesc nivelul 1 al diagramei de structură desenate în fig. 4.2 (b).

Modulele CALCUL și EDIT–RIND, ce aparțin noului modul introdus, alcătuiesc nivelul 2 al diagramei de structură.

De notat că, nu întotdeauna primul nivel al unei diagrame de structură satisface toate cerințele de prelucrare a programului. De aceea, apare ca necesară rafinarea (detalierea) anumitor blocuri (vezi blocul PRELUCRĂRI).

După cum remarcăm, blocul PRELUCRĂRI conține în colțul din dreapta sus un asterisc care are următoarea semnificație: «EXEMPLUL 4 "constă din", mai multe PRELUCRĂRI ce "constau din" CALCUL "urmat de" EDIT–RIND».

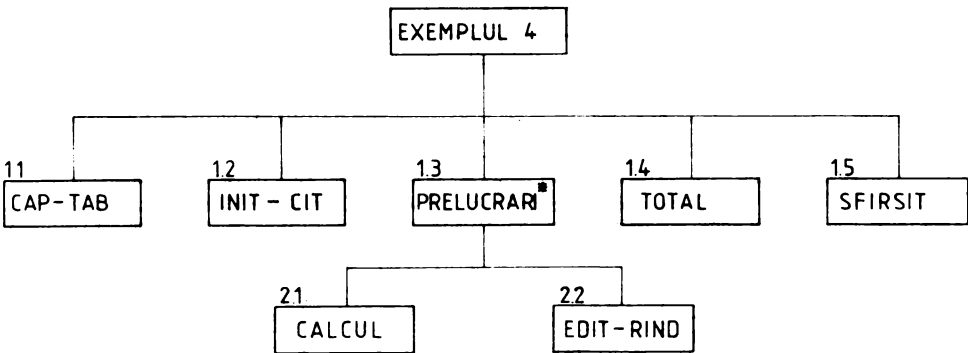
Altfel spus, blocul PRELUCRĂRI reprezintă o structură de repetiție (iterație), în cazul nostru cu număr cunoscut de pași –9.

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Nume program: ELEMPELE 4 – PC, m, M, F

Modul	Funcțiuni
CAP-TAB	3
INIT-CIT	1, 2
CALCUL	4, 5, 6
EDIT-RIND	7
TOTAL	10
SFIRȘIT	11
PRELUCRARI	8, 9

a)



b)

PSEUDOCODUL

Nume program: ELEMPELE 4 – PC, m, M, F

B : citește densitatea benzinei

C : afișează cap-tabel

PENTRU fiecare rezervor a cărui rază variază DE LA 2 LA 10

D"1 : calculează volumul

D"2 : calculează masa de benzină

D"3 : însumează masa de benzină calculată în cantitatea totală de benzină

D"4 : afișează raza și masa de benzină

A

SFIRȘIT

E : afișează masa totală de benzină

c)

Fig. 4.2. Modulul de proiectare structurată: a) alocarea funcțiilor de prelucrare; b) diagrama de structură; c) pseudocodul (varianta 1);

PSEUDOCODUL

Nume program: EXEMPLELE 4 – PC, m, M, F

EXEMPLUL4	SEQ
INIT-CIT	SEQ
	INPUT D
	Inițializare variabile
INIT-CIT	END
PRELUCRĂRI	PENTRU R DE LA 2 LA 10
CALCUL	SEQ
	$V=2 \pi R^3$
	$M=D * V$
	$S=S+M$
CALCUL	END
	PRINT R, M
PRELUCRĂRI	SFIRȘIT
	PRINT S
EXEMPLUL4	END

d)

PSEUDOCODUL

Nume program: EXEMPLELE 4 – PC, m, M, F

EXEMPLUL 4	SEQ
INIT-CIT	SEQ
	INPUT D
	S=0
INIT-CIT	END

(se completează cu blocul PRELUCRĂRI din fig. 4.2 (d))

:

e)

d) pseudocodul (varianta 2); e) pseudocodul (varianta finală);

Test

Verificați dacă modulele „CALCUL” și „EDIT-RIND” se repetă de nouă ori.

De notat că o astfel de reprezentare (de tip JACKSON) se cere a fi completată cu informații (structurate de obicei într-un tabel) suplimentare cu rolul de a specifica în clar toate condițiile ce intervin în cadrul programului.

Remarcă. Numele blocurilor ce alcătuiesc diagrama de structură trebuie să fie cât mai sugestive, scurte, reprezentative.

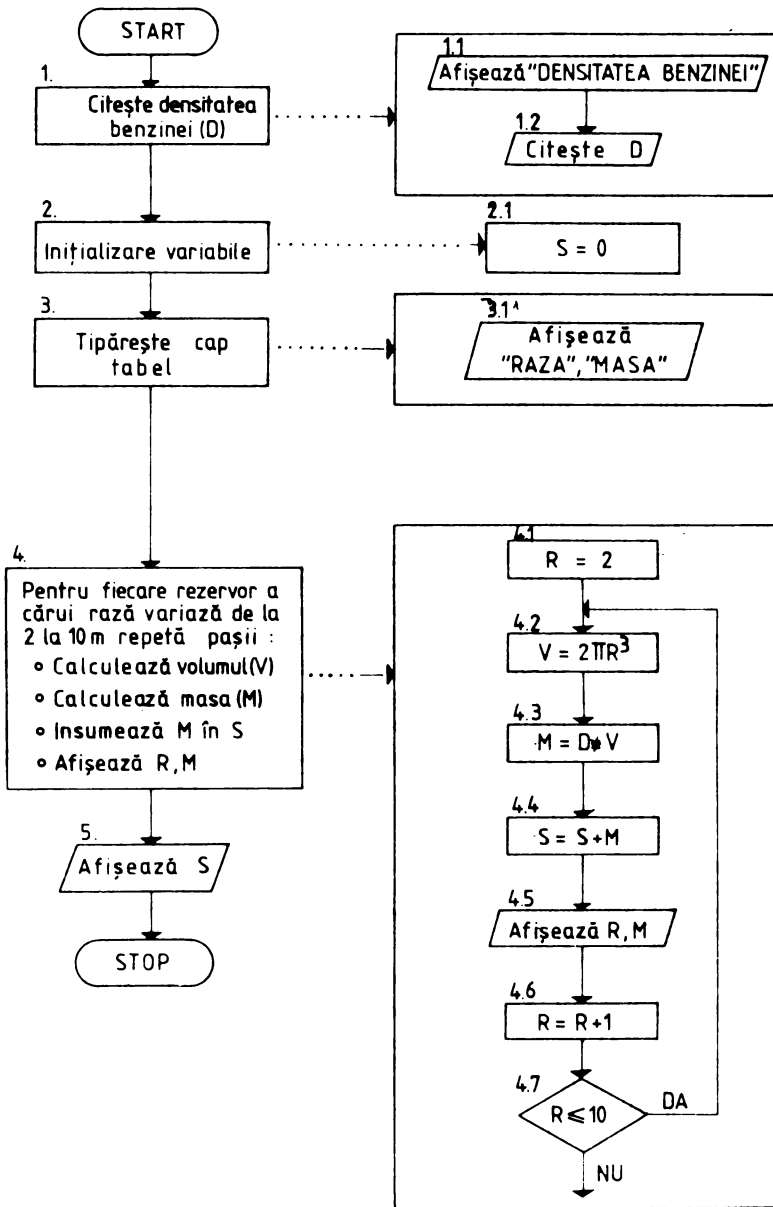


Fig. 4.2. f) schema logică.

Nu puține sînt cazurile cînd, pentru prelucrarea unui articol, folosim un modul comun și altor prelucrări.

Din analiza figurii 4.2 (b) ce conține modulele de prelucrare cu funcțiunile corespunzătoare, se observă că pentru EXEMPLELE 4 – PC, m, M, F nu întîlnim module cu funcțiuni comune.

Odată trasată diagrama de structură, se recomandă o revedere a acesteia pentru o ultimă verificare a completitudinii și corectitudinii ei.

În sfârșit, ultima etapă privind proiectarea diagramei de structură constă în numerotarea blocurilor diagramei (modulele comune vor purta același număr).

Nu este obligatoriu ca blocurile să fie numerotate în secvență. De obicei, prima cifră a modulului reprezintă numărul nivelului de detaliere, cea de-a doua, poziția pe care o ocupă în cadrul fiecărui nivel ș.a.m.d. (vezi fig. 4.2 (b)).

Scrierea pseudocodului

Varianta 1. Sarcina calculatorului poate fi descrisă prin enunțul A:

- A** Citește densitatea benzinei, afișează capul de tabel, calculează, însumează și afișează cantitatea de benzină din rezervoarele cilindrice echilaterale a căror rază variază de la 2 la 10 m cu pasul 1.

Evident, enunțul A nu reprezintă o acțiune primitivă, impunându-se în acest sens o rafinare, o descompunere a acesteia. Există mai multe metode de descompunere a unei acțiuni neprimitive în acțiuni primitive. În cele ce urmează vom aborda *metoda analizei descendente*. A face *analiza descendentă* a unei probleme P descrise printr-un enunț neprimitiv înseamnă a stabili o secvență de enunțuri e_1, e_2, \dots, e_n rezultate în urma descompunerii problemei P, astfel încât execuția acestora să ducă la realizarea obiectivului propus.

Rafinăm, în consecință, acțiunea neprimitivă A:

- B: citește densitatea benzinei
 C: afișează cap-tabel
A D: calculează și însumează masa de benzină din rezervoarele a căror rază variază de la 2 la 10 m cu pasul 1
 E: afișează masa totală de benzină.

Acțiunea D, nefiind primitivă, continuăm analiza descendentă.

D1: pentru rezervorul cu raza de 2 m, calculează volumul, calculează masa de benzină, însumează masa de benzină calculată în cantitatea totală de benzină, afișează raza și masa.

D :

D9: pentru rezervorul cu raza de 10 m calculează volumul, calculează masa de benzină, însumează masa de benzină calculată în cantitatea totală de benzină, afișează raza și masa.

Se observă că acțiunile D1–D9 se repetă pentru valori ale razei cuprinse între 2 și 10 m. Apare deci necesară existența unei structuri algoritmice care să permită descrierea repetiției (iterației) într-o formă comodă. O astfel de structură care definește un grup de acțiuni (enunțuri) sau o combinație de structuri fundamentale algoritmice cu execuție repetată este cunoscută sub denumirea de *structură iterativă*.

În consecință, acțiunile D1÷D9 pot fi restrinse într-o structură de iterație de forma:

Pentru fiecare rezervor a cărui rază variază de la 2 la 10 m, cu pasul 1, repetă acțiunile:

- D'
- calculează volumul
 - calculează masa de benzină
 - însumează masa de benzină în cantitatea totală de benzină
 - afișează raza și masa

Structura de iterație cu număr cunoscut de pași – PENTRU

Descrierea de mai sus, foarte comodă de altfel, reprezintă structura de iterație cu număr cunoscut de pași – **PENTRU**, pe care o vom utiliza în vederea reformulării algoritmului prezentat anterior. Ea poate fi descrisă cu formatul:

PENTRU (expresie)

```

a1
a2
.
.
.
an

```

-acțiuni ce se execută de un număr finit de ori

SFIRȘIT

unde, **PENTRU** și **SFIRȘIT** au și rolul de delimitatori ai secvenței de acțiuni ce se execută.

Aplicăm acest format la scrierea acțiunii D .

PENTRU fiecare rezervor a cărui rază variază **DE LA 2 LA 10**

- D''
- D''1: calculează volumul
 - D''2: calculează masa de benzină
 - D''3: însumează masa de benzină în cantitatea totală de benzină
 - D''4: afișează raza și masa de benzină

SFIRȘIT

Observații:

- Dacă procesorul (calculatorul) înțelege structura **PENTRU** ... **SFIRȘIT**, atunci nu mai este nevoie să continuăm descompunerea acțiunii D.
- Utilizând structura **PENTRU** ... **SFIRȘIT**, sarcina de a varia valoarea razei și de a opri iterația îi revine calculatorului. Când modul de variație nu este precizat, se presupune implicit o creștere cu o unitate între două valori succesive ale variabilei.

Forma finală a primei variante de pseudocod pentru **EXEMPLELE 4 – PC, m, M, F** este prezentată în modulul de proiectare structurată, fig. 4.2(c).

Aplicație. Un cilindru are înălțimea de 10 cm, iar raza R variabilă. Transcrieți în pseudocod algoritmul de calcul al ariei laterale, totale și al volumului cilindrului când R variază de la 1 la 18 cm, cu pasul de 1 cm.

Sarcina calculatorului poate fi descrisă prin acțiunea **A**:

- A**
- pentru fiecare cilindru a cărui rază variază de la 1 la 18 cm, cu pasul 1, repetă acțiunile: a) calculează și tipărește aria laterală; b) calculează și tipărește aria totală; c) calculează și tipărește volumul.

Se observă că putem identifica o structură de iterație de tip **PENTRU**.

PENTRU fiecare cilindru a cărui rază variază **DE LA 1 LA 18**

A
 A1' : calculează aria laterală
 A2' : tipărește aria laterală
 A3' : calculează aria totală
 A4' : tipărește aria totală
 A5' : calculează volumul
 A6' : tipărește volumul

SFIRȘIT

TEST

Transcrieți în pseudocod algoritmul de calcul al ariei și lungimii cercurilor a căror rază variază de la 10 la 20 m, cu pasul 1 m.

Prezentarea acțiunilor primitive de o manieră formalizată

Varianta 2. Prezentarea algoritmului în limbaj natural (vezi varianta 1 a pseudocodului) are, după cum ați putut constata și singuri, unele dezavantaje: lizibilitate redusă, exprimări lungi, greoaie etc. De aceea, se preferă o formalizare a acțiunilor primitive (**INPUT, PRINT** etc.) care să pună în evidență atât variabilele de intrare/ieșire, cât și variabilele de stare.

Varianta prezentării acțiunilor primitive de o manieră formalizată este de preferat celei narative. În consecință, s-o folosim cât este cu putință pe aceasta!

Înainte de a prezenta noua variantă a pseudocodului corespunzătoare **EXEMPLELOR 4 – PC, m, M, F**, va trebui să completăm tabela de variabile cu o nouă variabilă de stare – **S** (vezi modulul de analiză structurată, fig. 4.1(d)).

Această operație este necesară deoarece acțiunea primitivă

D3: însumează masa de benzină calculată în cantitatea totală de benzină
 reclamă definirea variabilei "cantitatea totală de benzină" ca pe o variabilă de stare.

Pseudocodul corespunzător **EXEMPLELOR 4 – PC, m, M, F** (varianta 2) este prezentat în modulul de proiectare structurată, fig. 4.2(d).

Remarcați în descrierea pseudocodului instrucțiunile standard cunoscute: **SEQ, END** dar și noile instrucțiuni standard: **PENTRU, SFIRȘIT** care modifică o structură algoritmică de iterație cu număr cunoscut de pași.

Formatul general al blocului de iterație **PENTRU . . . SFIRȘIT** este:

Format general

Etichetă **PENTRU** **V DE LA V1 LA V2 PAS R**

(corpul buclei)

Etichetă **SFIRȘIT**

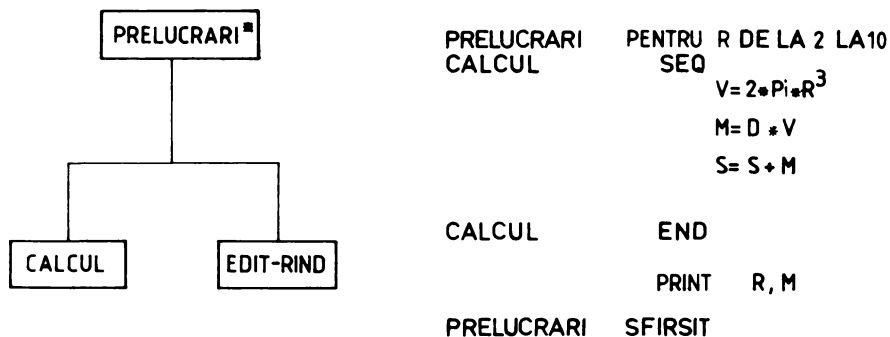
unde, **V** este variabila de control, **V1** valoarea inițială a variabilei, **V2** valoarea finală a structurii de iterație, **R** pasul de incrementare.

REMARCI

- Instrucțiunile **PENTRU** și **SFIRȘIT** au obligatoriu aceeași etichetă.
- Instrucțiunea **PENTRU** reprezintă punctul unic de intrare în bloc.
- Instrucțiunea **SFIRȘIT** reprezintă punctul unic de ieșire din bloc.

Cît privește varianta 2 de pseudocod pentru **EXEMPLELE 4 – PC, m, M, F**, remarcați cum **PRELUCRĂRI** joacă rolul etichetei (aceeași pentru instrucțiunile **PENTRU** și **SFIRȘIT**, iar acțiunile cuprinse între cei doi delimitatori reprezintă acțiuni (instrucțiuni) ce se execută în secvență (vezi blocul de secvență **CALCUL**).

În figura de mai jos se prezintă în paralel imaginea blocului **PRELUCRĂRI** reprezentat în diagrama de structură și în pseudocod.



Remarcă. Nu putem trece neobservată acțiunea primitivă: „Inițializare variabile” ce aparține blocului de secvență **INIT-CIT**. Este foarte greu să ne dăm seama chiar de la începutul alcătuirii pseudocodului ce variabile trebuie inițializate. De aceea, vom preceda prin a semna la într-un prim pas acțiunea de inițializare a variabilelor, urmînd să revenim atunci cînd am obținut deja o primă imagine completă a pseudocodului.

Pentru **EXEMPLELE 4 – PC, m, M, F**, singura cerință de inițializare o reclamă acțiunea (instrucțiunea):

$$S = S + M$$

din blocul **CALCUL** (aparține structurii algoritmice de iterație identificate prin blocul **PRELUCRĂRI**).

Evident, execuția acțiunii “ $S = S + M$ ” nu are sens pentru noi decît dacă S a primit în prealabil o valoare inițială, respectiv zero.

În consecință, vom introduce în locul acțiunii neprimitive „Inițializare variabile” acțiunea primitivă de afectare (atribuire) formalizată:

$$S = 0$$

Varianta 2 a pseudocodului, completă este prezentată în modulul de proiectare structurată, fig. 4.2(e).

Aplicații. Alcătuiți pseudocodul pentru rezolvarea următoarelor probleme:

a) Să se calculeze suma primelor n numere naturale.

- | | |
|---|---|
| <p>A SEQ
 INPUT N
 $S = 0$
 PENTRU I DE LA 1 LA N
 $S = S + I$</p> | <p>A SFIRȘIT
 PRINT S
 B END</p> |
|---|---|

TEST

Modificați algoritmul de mai sus pentru calculul sumei pătratelor primelor n numere naturale.

b) Să se calculeze produsul primelor n numere naturale (calculul lui n!).

```

FACT   SEQ           F           P=P*I
          INPUT N       SFIRȘIT
          P=1           PRINT P
F       PENTRU I DE LA 1 LA N   FACT   END
    
```

c) Să se calculeze $\sum_{k=1}^n K!$

```

SFACT  SEQ           PREL       PENTRU I DE LA 1 LA N
          INPUT N       P=P*I
          SEQ           S=S+P
          S=0           SFIRȘIT
          P=1           PRINT S
          END           SFACT       END
    
```

d) Să se calculeze constanta elastică echivalentă a 3 resorturi de constante elastice identice K legate în: a) serie; b) paralel.

Indicație. Pentru legarea în serie și în paralel se vor utiliza relațiile de calcul:

$$\frac{1}{k} = \sum_{i=1}^n \frac{1}{k_i}, \quad k = \sum_{i=1}^n k_i$$

```

A1     SEQ           B1     SEQ
          INPUT K       S=S+1/K
          S=0           P=P+K
          P=0           B1     END
          PENTRU I DE LA 1 LA 3   B     SFIRȘIT
                                          PRINT P, S
                                          A1    END
    
```

Trasarea schemei logice

Schema logică pentru EXEMPLELE 4 – PC, m, M, F este reprezentată în modulul de proiectare structurată, fig. 4.2(f). Ne vom opri în cele ce urmează asupra modului de reprezentare grafică a structurii de repetiție cu

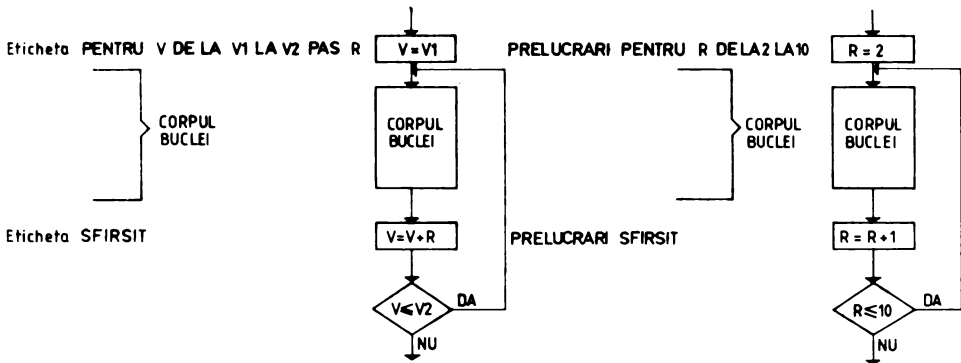
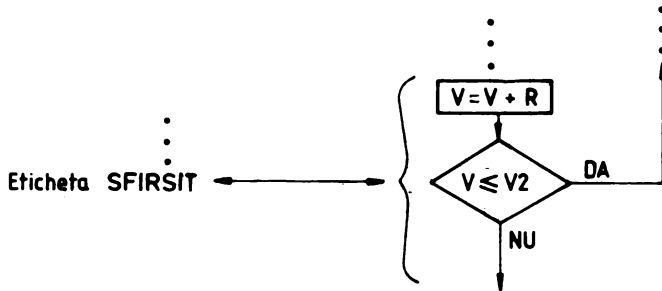


Fig. 4.3.

număr finit de pași **PENTRU** . . . **SFIRȘIT** (vezi blocul 4 din fig. 4.2(f) și rafinarea acestuia în 4.1, . . . , 4.7). Pentru o înțelegere mai ușoară a mecanismului privind trasarea schemei logice vom apela din nou la reprezentarea în pseudocod (vezi fig. 4.3) a structurii **PENTRU**.

După cum constatați, schema logică a structurii de iterație **PENTRU** începe cu un bloc de inițializare a variabilei de control ($V=V1$), continuă cu corpul buclei și se termină cu un bloc de incrementare a variabilei de control V ($V=V+R$), urmat de un bloc de decizie. În cadrul blocului de decizie se testează valoarea variabilei V după cum urmează: dacă valoarea lui V este mai mică sau egală cu $V2$ (valoarea finală a variabilei V) se reia ciclul începând cu primul bloc de după $V=V1$, în caz contrar se iese din ciclu urmînd a se executa blocul imediat următor.

Observație. Delimitatorul **SFIRȘIT** din pseudocodul structurii de iterație **PENTRU** îi corespund în cadrul schemei logice a aceleiași structuri două blocuri: unul de incrementare a variabilei de control și unul de comparare a variabilei de control cu valoarea finală a acesteia.



Aplicație. Să se calculeze C_n^k . Se va alcătui tabela de variabile și se va trasa schema logică.

Indicație. Se va utiliza formula

$$C_n^k = \frac{n!}{k! (n-k)!} \quad (2)$$

unde $n \geq k, n > 0, k > 0$.

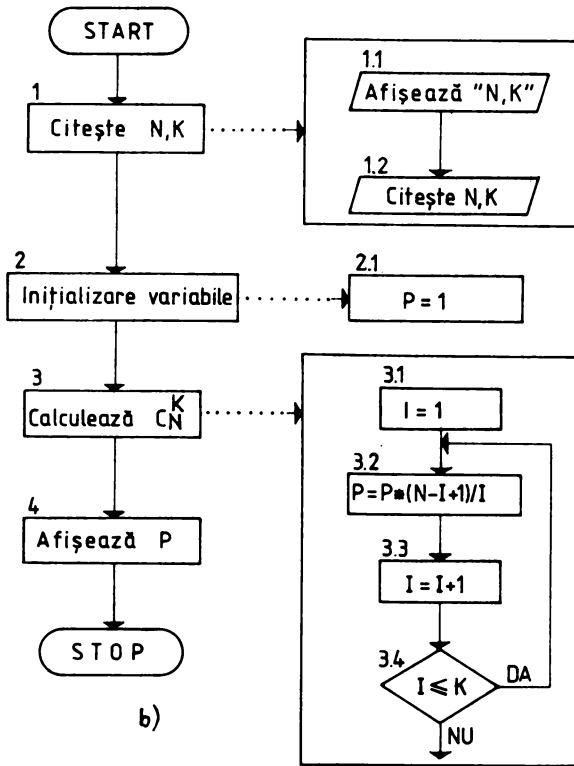
Tabela de variabile și schema logică a aplicației sint prezentate în modulele de analiză și proiectare structurată, fig. 4.4.

TABELA DE VARIABILE

Variabile de intrare	Variabile de stare	Variabile de ieșire
N: numărul de elemente ale mulțimii finite K: numărul submulțimilor avînd fiecare cîte K elemente	I: variabila de control a structurii de iterație PENTRU	P: numărul combinațiilor de n elemente luate cîte k

a)

Fig. 4.4. a-b. a) tabela de variabile;



b)

b)

b) schema logică.

□ Codificarea în limbajul BASIC-aMIC

vol. 2, pag. 205

FOR – NEXT

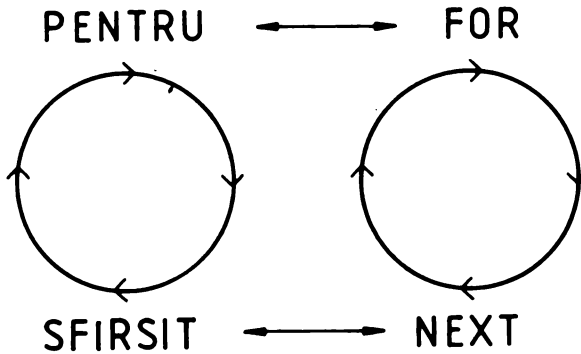
Programul pe care îl analizăm în cele ce urmează conține unul din cele mai "forte" instrumente de conversație BASIC: bucla **FOR** – **NEXT**.

Cele două instrucțiuni considerate ca fiind cele mai "tari" instrucțiuni ale BASIC-ului, din păcate foarte des confundate de începători, sînt echivalentul instrucțiunilor **PENTRU** respectiv **SFIRȘIT** din structura de repetiție cu număr finit de pași întilnită la reprezentarea în pseudocod.

Cu instrucțiunile **FOR** și **NEXT** spunem calculatorului de cîte ori să treacă prin buclă. După aceea, la ieșirea din buclă, calculatorul continuă execuția programului cu instrucțiunea imediat următoare.

Instrucțiunile **FOR** și **NEXT** apar în linii diferite în cadrul programului. **FOR** reprezintă punctul de intrare în buclă și apare întotdeauna prima, iar **NEXT** este ultima instrucțiune din buclă reprezentînd punctul de ieșire din ciclu (buclă). Cît privește instrucțiunea sau instrucțiunile din buclă, cu-

prinse între **FOR** și **NEXT**, acestea sînt executate în mod repetat în ordinea numerelor de linie, de atîtea ori de cît s-a precizat în instrucțiunea **FOR**.



Acest lucru se poate vedea și în program, unde valoarea lui **R** crește de la 2 la 10.

```

340 FOR R=2 TO 10
350   V=2*PI*R↑3
360   M=D*V
370   S=S+M
380   PRINT R, M
390 NEXT R

```

După ce a parcurs de nouă ori bucla (instrucțiunea **FOR** spune calculatorului ca **R** să ia valori de la 2 la 10) calculatorul execută instrucțiunea 420, imediat următoare, după **NEXT** (400 este instrucțiune neexecutabilă) și în final execută instrucțiunile 430 și 440.

Observație. Nu este obligatoriu ca **R** să ia valori începînd cu 1!

TEST

În programul precedent, bucla **FOR-NEXT** ocupă liniile și
R. 340, 350, 360, 370, 380, 390.

Cum lucrează bucla **FOR-NEXT**?

Urmăriți săgețile din figura 4.5. După cum constatați, calculatorul ajunge de fiecare dată la instrucțiunea **NEXT R**, incrementează valoarea lui **R** cu o unitate și compară noua valoare cu 10 (valoarea limită a lui **R**). Cînd valoarea lui **R** este mai mare decît 10, calculatorul continuă execuția programului cu următoarea instrucțiune din program, care este în afara buclei.

Ați înțeles? Să vedem. Prima valoare din buclă a lui **R** este 2 ($R=2$). La a doua parcurgere a buclei

$$R=R+1=2+1=3$$

la a opta parcurgere a buclei $R=?$

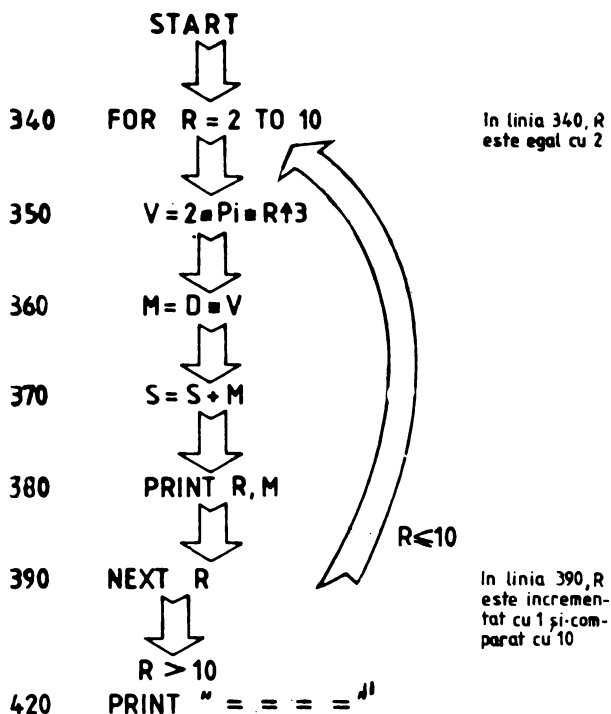


Fig. 4.5. Mecanismul de funcționare a buclei FOR-NEXT.

Dacă ați răspuns 9, felicitări, dacă nu vă invităm să citiți încă o dată explicațiile din fig. 4.5.

Formatele generale ale instrucțiunilor FOR și NEXT sînt:

Format general

<nr. linie> FOR <variabilă> = <exp.1> TO <exp.2> [STEP <exp.3>]

Format general

<nr. linie> NEXT <variabilă>

Remarci:

- <variabilă> reprezintă o variabilă simplă, de control a buclei.
- <variabilă> din instrucțiunea NEXT este aceeași cu <variabilă> din instrucțiunea FOR.
- <exp.1> este valoarea inițială care se atribuie variabilei de control a buclei.
- <exp.2> este valoarea finală a variabilei de control a buclei.
- STEP este opțional.
- <exp.3> este incrementul (pasul) care se adună la valoarea variabilei de control; în absența acestuia se ia automat valoarea unu.
- <exp.1>, <exp.2>, <exp.3> sînt expresii numerice.
- în interiorul unei bucle FOR pot exista maximum trei bucle incluse una în alta cu condiția ca acestea să nu se intersecteze.



- Dacă valoarea limită a variabilei de control este depășită, execuția buclei ia sfârșit, următoarea instrucțiune ce urmează a fi executată fiind cea de după **NEXT**.

În figura 4.6 se prezintă modul de implementare în BASIC a structurii de iterație cu număr finit de pași **PENTRU** (v. blocul **PRELUCRARI** din diagrama de structură).

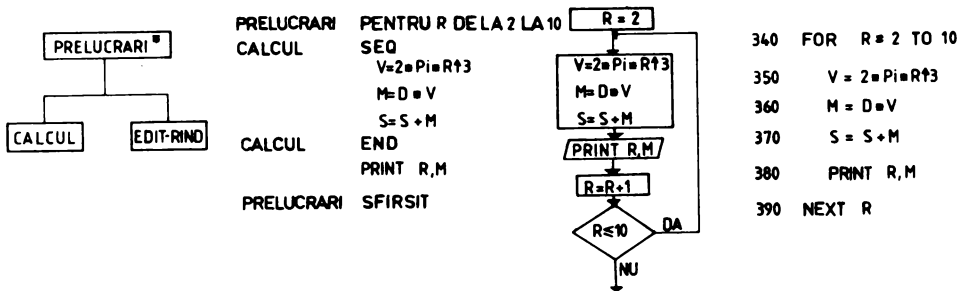


Fig. 4.6

TEST

Scrieți cite o buclă **FOR-NEXT** pentru fiecare din cazurile:

- o buclă trebuie repetată de 999 ori;
- o buclă trebuie repetată de cite ori este necesar pentru ca variabila de control să varieze de la 2 la 82, din 4 în 4.
- o buclă trebuie repetată de cite ori este necesar pentru ca variabila de control să varieze de la 0.01 pină la valoarea dată de expresia $B * B * B - 8$. La fiecare execuție a buclei variabila de control își va mări valoarea cu $B + K$.

Mai multe despre FOR-NEXT

Un alt aspect ce trebuie notat în legătură cu bucla **FOR-NEXT** este acela că atât valoarea inițială cât și valoarea finală a variabilei de control pot fi nu numai numere ci și expresii (v. formatul general al instrucțiunii **FOR**). Aceași observație este valabilă și pentru pasul de incrementare. Iată un exemplu în acest sens:

```

330 S = 0
335 A = 2
336 B = 10
340 FOR R = A TO B
350 V = 2 * PI * R ^ 3
360 M = D * V
370 S = S + M
380 PRINT R, M
390 NEXT R
    
```

Spre a vă convinge de justetea celor afirmate executați programul BASIC-aMIC modificat (v. liniile 335, 336, 340) și confrunțați rezultatele. Evident ele vor fi aceleași.

TESTE

1) Precizați care va fi răspunsul pe care-l va da calculatorul personal aMIC in urma execuției programelor:

a) 10 X = 0
20 Y = 3
30 FOR A = X TO Y
40 PRINT A
50 NEXT A
60 END

b) 10 A=4
20 FOR P=A TO 2*A-1
30 PRINT P;
40 NEXT P
50 END

2) Completați locurile libere din linia 20 astfel încit in urma execuției programului:

10 A = 4
20 FOR B = TO
30 PRINT B;
40 NEXT B
50 END

să se obțină următoarele rezultate

RUN
2 3 4 5 6 7 8 9 10 11 12

3) Se consideră următorul program BASIC-aMIC

```
10 PRINT "NUMAR NOTE";
20 INPUT N
30 S=0
40 FOR I=1 TO N
50 PRINT "INTRODUCEȚI NOTA T";
60 INPUT T
70 S=S+T
80 NEXT I
90 M=S/N
100 PRINT "MEDIA="; M
110 END
```

a) Realizați singuri specificațiile de programare și documentația de proiectare a acestui program.

b) In programul de mai sus bucla **FOR-NEXT** ocupă liniile

c) Care linie din bucla **FOR-NEXT** calculează suma notelor (T)?

d) Dacă N=4, de cite ori va fi afișat pe ecran "INTRODUCEȚI NOTA T"?

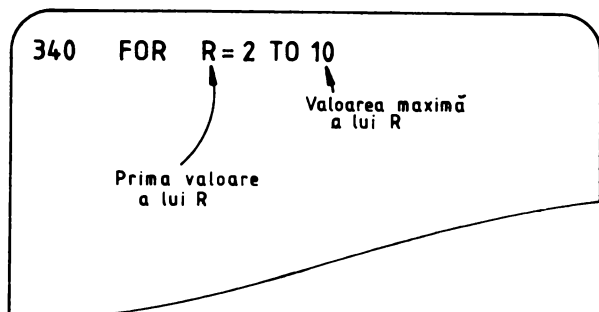
4) Următorul program realizează produsul a N numere! Completați programul de mai jos cu instrucțiunile care lipsesc:

```
10 PRINT "N= ";
20 INPUT N
30 PRINT
40 PRINT
50 ..... ?
60 ..... ?
70 PRINT "A= ";
80 INPUT A
90 ..... ?
100 NEXT I
110 PRINT
120 PRINT "PRODUSUL= "; P
130 END
```

Rezultatul execuției programului este:

```
RUN
N=?4
A=?20
A=?10
A=?2
A=?3
PRODUSUL = 1200
```

S-a observat că în buclele **FOR-NEXT** de până acum, prima valoare care se atribuie variabilei de control al ciclului prin (exp.1) se păstrează până ce calculatorul ajunge la instrucțiunea **NEXT**. Apoi, variabila din instrucțiunea **FOR** își mărește valoarea cu 1 de câte ori se reparcurge bucla până la valoarea maximă precizată prin (exp.2).



De altfel, puteți scrie o instrucțiune **FOR** în care valoarea variabilei de control să crească sau să scadă fie cu valori întregi, fie cu valori reale etc. utilizând opțiunea **STEP**.

În tabelul 4.1 sînt prezentate cîteva exemple în acest sens.

Tabelul 4.1

Instrucțiune	Semnificație
10 FOR I=1 TO 8 STEP 2	Valoarea lui I crește cu 2 ori de cîte ori se parcurge bucla cu condiția ca $I \leq 8$.
10 FOR A=4 TO 7 STEP 1.5	Valoarea lui A crește cu 1.5 ori de cîte ori se parcurge bucla cu condiția ca $A \leq 7$.
10 FOR X=25 TO 5 STEP - 1	Valoarea lui X scade cu 1 ori de cîte ori se parcurge bucla pînă ce X este mai mic decît 5. <i>Observație.</i> X pornește de la 25 și ajunge la 5!

"Bunicul" lui FOR-NEXT

Nu puține sînt cazurile cînd în interiorul unei bucle **FOR** trebuie să facem loc unei alte bucle **FOR** ș.a.m.d. Spunem că avem de-a face cu cicluri în cicluri sau cicluri suprapuse (cicluri imbricate). După cum precizam (v. formatul general al instrucțiunii) condiția fundamentală de definire corectă a ciclurilor imbricate (maximum trei în BASIC-aMIC) este neintersecția acestora. De notat, că fiecare ciclu trebuie să aibă propriile instrucțiuni **FOR**, **NEXT** și propriile variabile de control ale buclelor.

Pentru mărirea lizibilității programelor care conțin cicluri suprapuse se recomandă o scriere în "fierăstrău" a acestora (deplasare spre dreapta a ciclurilor interioare).

Aplicație. Scrieți un program BASIC care să listeze 4 linii de asteriscuri, fiecare linie conținând trei asteriscuri.

```

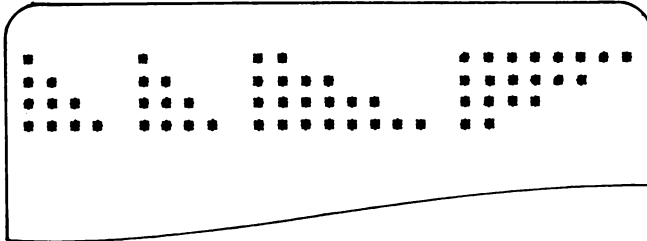
10 FOR I=1 TO 4
20   FOR J=1 TO 3
30     PRINT "*";
40   NEXT J
50   PRINT
60 NEXT I

```

RUN
* * *
* * *
* * *
* * *

TEST

Modificați programul precedent astfel încât să tipărească următoarele rînduri:



Codificarea în limbajul BASIC-PRAE

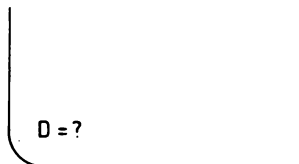
vol. 2, pag. 205

Particularități ale programării

În limbajul BASIC-PRAE, spre deosebire de limbajul BASIC implementat pe calculatorul aMIC se poate vorbi de un „mariage” între instrucțiunile **PRINT** și **INPUT**. Acest aspect este ilustrat în program prin instrucțiunea:

```
240 INPUT "D="; D
```

Lansînd în execuție programul cu comanda **RUN**, pe ultimul rînd al ecranului va apare afișat:



Punînd semnul de întrebare, calculatorul personal PRAE ne semnalează că așteaptă de la noi valoarea lui D sau altfel spus (vezi tabela de variabile) valoarea densității.

După cum precizam, calculatorul este „înarmat” cu foarte multă răbdare și așteaptă (pe timpul nostru (!)) pînă ne hotărîm să tastăm 0.7 – valoarea densității benzinei. Executînd instrucțiunea **INPUT** calculatorul va atribui valoarea 0.7 variabilei D. Execuția programului continuă cu linia 250, apoi cu linia 260 ș.a.m.d.

Formatul general al instrucțiunii **INPUT** (extinse) este:

Format general

{nr. linie} **INPUT** [{constanta șir};] {variabila 1} [, {variabila 2}, ...]

De notat că variabilele se separă prin virgule, iar după constanta șir (opțională) se pune simbolul " ; ".

Aplicație. Se consideră următorul program BASIC-PRAE care calculează volumul unui paralelipiped:

```
10 INPUT "L1, L2, L3"; L1, L2, L3
20 V=L1*L2*L3
30 PRINT "VOLUMUL=";V
40 END
```

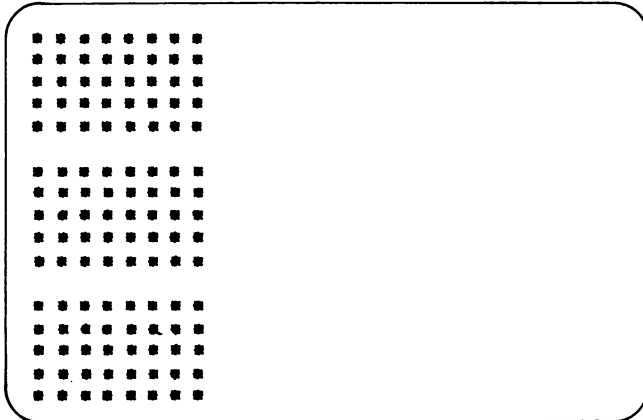
În momentul execuției instrucțiunii **INPUT** din linia 10, **PRAE**-ul afișează mesajul "L1, L2, L3?" iar după ce am introdus prima valoare (lungimea) va afișa încă două semne de întrebare, avertizându-ne în acest fel că încă-i sîntem ... datori. Dacă, din neatenție vom tasta ceva diferit de un număr vom fi imediat "sanționați" cu mesajul "INVALID INPUT", după care se afișează din nou un semn de întrebare.

Numai după ce am răspuns corect instrucțiunii **INPUT** din linia 10, se calculează V iar apoi, pe rîndul următor "VOLUMUL=" urmat de valoarea variabilei V calculate în linia 20.

Ciclurile FOR-NEXT aparținînd limbajului BASIC-PRAE, spre deosebire de cele din BASIC-aMIC, prezintă, în plus, următoarele **particularități**:

- variabila de control a ciclului poate fi modificată în interiorul buclei;
- într-o buclă **FOR-NEXT** pot fi imbricate oricîte cicluri dorim, cu condiția ca acestea să nu se intersecteze.

Aplicație. Scrieți un program BASIC-PRAE care să afișeze:



```
10 FOR I=1 TO 3
20 PRINT
30   FOR J=1 TO 5
40     FOR K=1 TO 8
50       PRINT "*";
60     NEXT K
70   PRINT
80   NEXT J
90 NEXT I
100 END
```

TESTE

1) Se consideră următorul program BASIC:

```
10 FOR I=1 TO 10 STEP 2
20
30 NEXT I
```

```
40 PRINT
50 PRINT "I="; I
60 END
```

Precizați rezultatul execuției acestui program.

2) Completați răspunsul dat de calculator pentru programele de mai jos:

a) 10 X=3
20 FOR Y=X TO 4*4 STEP X
30 PRINT Y;
40 NEXT Y
50 END

e) 10 FOR I=5 TO 7.5 STEP.5
20 PRINT I,
30 NEXT I
40 END

b) 10 FOR I=100 TO 10 STEP-10
20 PRINT I;
30 NEXT I
40 END

f) 10 A=0
20 C=1
30 FOR I=1 TO 4
40 A=A+1
50 C=C*I
60 NEXT I
70 PRINT "A="; A
80 PRINT "C="; C
90 END

c) 10 FOR A=27 TO 18 STEP-3
20 PRINT A;
30 NEXT A
40 END

g) 10 S=0
20 FOR I=1 TO 7 STEP 2
30 S=S+1
40 NEXT I
50 PRINT "S="; S
60 END

3) Se consideră următorul program BASIC:

```
10 S=0
20 PRINT "N="; ;
30 INPUT N
40 FOR I=1 TO N
45 PRINT "A="; ;
50 INPUT A
```

```
60 S=S+A
70 NEXT I
75 PRINT
80 PRINT "SUMA="; S
90 END
```

Ce valoare trebuie să introduceți dinamic pentru variabila N, astfel încât rezultatul execuției programului să fie cel prezentat în figură:

```

RUN
N=:
A=:1
A=:3
A=:4
A=:7
A=:2

SUMA= 17
READY
```

Codificarea în limbajul BASIC HC-85,
TIM S, SPECTRUM

vol. 2, pag. 205

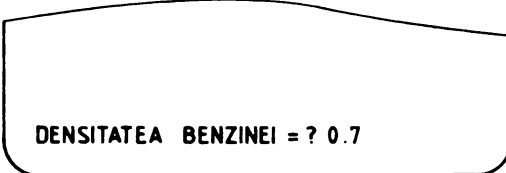
Particularități

Limbajul BASIC implementat pe calculatoarele personale HC-85, TIM S, SPECTRUM combină de asemenea pe **PRINT** și **INPUT** într-o sin-

gură instrucțiune **INPUT**. Această "scurtătură" se poate vedea și în linia 30 a programului:

```
30 INPUT "DENSITATEA BENZINEI = " ; D
```

La executarea instrucțiunii **INPUT** se afișează mai întâi pe terminalul utilizatorului literalul declarat în instrucțiune: "DENSITATEA BENZINEI =", urmat de un semn de întrebare (?).



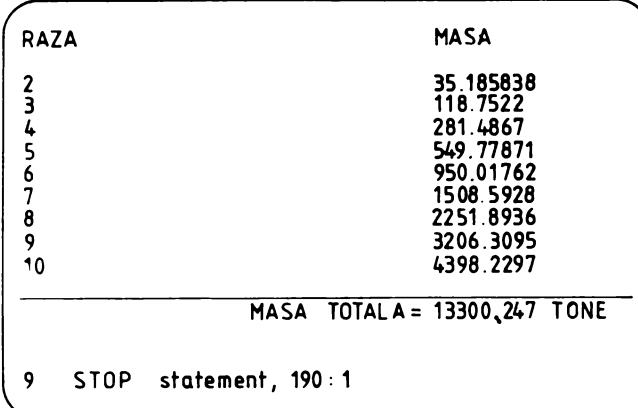
```
DENSITATEA BENZINEI = ? 0.7
```

Dacă literalul a fost omis, se afișează doar "?". În continuare se așteaptă ca utilizatorul să introducă datele solicitate, în ordinea indicată, separate prin virgulă.

Cit privește utilizarea buclei **FOR-NEXT** în limbajul BASIC HC-85, TIM S, SPECTRUM trebuie făcută următoarea **remarcă**:

Variabila de control a buclei trebuie să aibă numele format dintr-o singură literă.

În sfârșit, o ultimă observație privind forma de editare a rezultatelor:



```

RAZA                MASA
2                   35.185838
3                   118.7522
4                   281.4867
5                   549.77871
6                   950.01762
7                   1508.5928
8                   2251.8936
9                   3206.3095
10                  4398.2297
-----
                    MASA TOTALA = 13300,247 TONE
9  STOP  statement, 190 : 1

```

Întrebarea pe care vrem să v-o punem se referă la aspectul listei cu rezultate. Este momentul să fim mai pretențioși cu „ieșirile” încercând pe cât posibil să asigurăm un aspect plăcut rapoartelor generate de calculator.

Nu-i așa că lipsa aliniierilor punctului zecimal „deranjează” puțin un ochi avizat, obișnuit numai cu ordinea?

În BASIC HC-85, TIM S, SPECTRUM aceste „pretenții” pot fi satisfăcute, nu fără o mică dificultate cu instrucțiunea **STR\$** pe cînd în alte „dialecte” există căi mult mai simple (instrucțiunea **PRINT USING**).

Aplicații

Introduceți și executați următoarele programe:

- | | |
|--|---|
| <p>a) 10 FOR i=1 TO 8
20 PRINT TAB 4; i; TAB 10; i ↑ 2
30 NEXT i</p> <p>b) 10 LET m=3
20 LET n=14
30 FOR i=m TO n
40 PRINT TAB 4; i; TAB 6; i/10; TAB 14; i/m
50 NEXT i</p> <p>c) 10 FOR i=100 TO 1 STEP-12.5
20 PRINT TAB 8; i
30 NEXT i</p> <p>d) 10 FOR i=10 TO 1 STEP-0.175
20 PRINT TAB 10; i
30 NEXT i</p> | <p>e) 10 FOR i=1 TO 0
20 PRINT "*"
30 NEXT i</p> <p>f) 10 FOR I=(X=6)*33 TO 31
20 PRINT CHR\$(143)
30 NEXT I</p> <p>g) 5 FOR I=1 TO 5
10 INPUT "N=" ; N
20 PRINT "N=" ; N
30 PRINT N; "N*N"; N ↑ 2
40 NEXT I
50 STOP</p> |
|--|---|

TEST

Precizați dacă rezultatele execuției celor două programe (a, b) sînt identice.

- | | |
|--|---|
| <p>a) 10 FOR I=1 TO 6
20 PRINT "*";
30 NEXT I
40 PRINT "...."
50 END</p> | <p>b) 10 FOR I=1 TO 6
20 PRINT "*";
30 NEXT I
40 PRINT
50 PRINT "...."
60 END</p> |
|--|---|

Aplicație. Fiind date coordonatele mai multor perechi de puncte (acest număr se citește ca parametru) $M(X_1, Y_1)$, $N(X_2, Y_2)$ să se determine distanța MN.

Aplicație numerică: $M(3,4)$; $N(0,0)$ și $M(-3.4,5.75)$, $N(3.125,2)$. Schema logică a programului este prezentată în figura 4.7.

```

10 PRINT
20 PRINT "NUMĂR PERECHI
   PUNCTE"
30 INPUT N
40 FOR I=1 TO N
45 PRINT
50 PRINT "PUNCTUL M"
60 INPUT X1, Y1
70 PRINT "PUNCTUL N"
80 INPUT X2, Y2
90 LET D=((X2-X1) ↑ 2+(Y2-
   -Y1) ↑ 2) ↑ 0.5
100 PRINT "Distanța MN="; D
110 NEXT I
120
130 END

```

RUN

```

NUMĂR PERECHI PUNCTE? 2
PUNCTUL M? 3, 4
PUNCTUL N? 0, 0
Distanța MN=5
RUN
PUNCTUL M?-3.4,5.75
PUNCTUL N? 3.125,2
Distanța MN=7.52583

```

TEST

Precizați care din următoarele două programe este scris corect:

- | | |
|---|---|
| <p>a) 10 FOR A=1 TO 8
20 FOR B=2 TO 7
30 FOR C=2.3 TO 5.1
40 NEXT C
50 FOR D=A TO B
60 NEXT D
70 NEXT B
80 NEXT A</p> | <p>b) 10 FOR A=1 TO 8
20 FOR B=2 TO 7
30 FOR C=2.3 TO 6.1
40 NEXT A
50 NEXT C
60 FOR D=A TO B
70 NEXT D
80 NEXT B</p> |
|---|---|

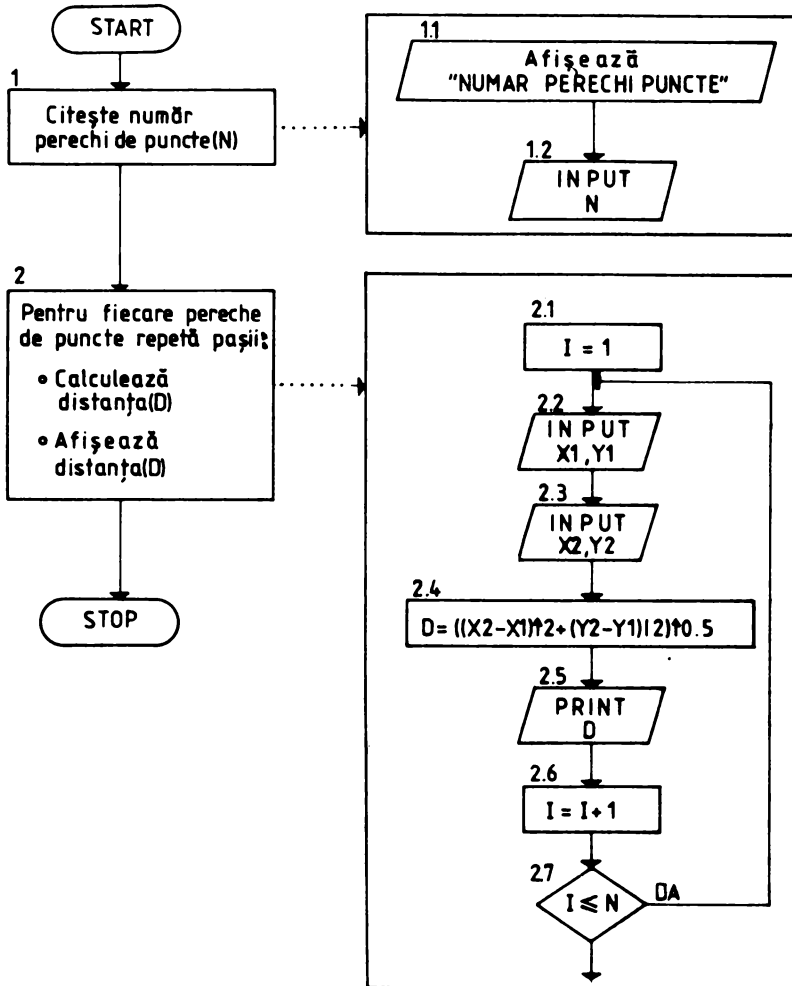


Fig. 4.7.

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 206

Instrucțiunea INPUT

În limbajul BASIC-AMSTRAD instrucțiunea **INPUT** al cărei format general este:

Format general	
(nr. linie)	INPUT [#(nr. canal),] [;] [(șir caractere) (separator)] (listă variabile)

prezintă următoarele **particularități**:

- citește datele de la canalul identificat prin (nr. canal) (dacă nu se precizează, se presupune automat #0);
- punctul și virgula după **INPUT** elimină trecerea la linia următoare după execuția instrucțiunii;
- (separator) poate fi punct și virgulă sau virgulă; ";" plasat după (șir caractere) generează un semn de întrebare, pe cînd "," îl suprimă;
- orice neconcordanță dintre numele variabilelor și tipul datelor este semnalată prin mesajul "? Redo from start" sau alt mesaj de eroare pe care îl puteți stabili chiar dvs.

Urmăriți, vă rugăm, în figura 4.8 modul în care are loc execuția instrucțiunii **INPUT** (linia 10) din cadrul programului BASIC-AMSTRAD.

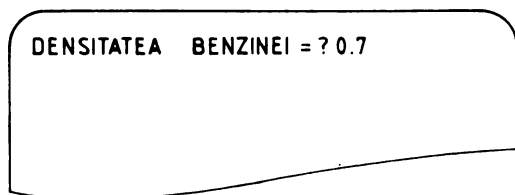


Fig. 4.8.

FOR cu ... NEXT anonim

Semnalăm în cele ce urmează o facilitate interesantă a limbajului BASIC-AMSTRAD privind bucla **FOR-NEXT** și anume: *atribuirea variabilei de control (definite în FOR) lui NEXT* este facultativă căci BASIC-ul implementat pe calculatoarele AMSTRAD determină automat cărei instrucțiuni **FOR** îi este asociat **NEXT**-ul corespunzător.

În cadrul programului este ilustrat acest caz (v. linia 27).

```
25 FOR i=1 TO 40
26   PRINT "-";
27   NEXT
```

Cît privește ... bunicul lui **FOR-NEXT** AMSTRAD vă prezentăm un program pe care l-am mai întilnit deja în cadrul lucrării, dar de această dată scris în limbaj BASIC-AMSTRAD.

```
10 FOR I=1 TO 3
20   FOR J=1 TO 2*I
30     FOR K=1 TO 2*(J+I)
40       PRINT "*";
50   NEXT K, J, I
60 END
```

Observație. Pentru neintersectarea instrucțiunilor **FOR**, am fost nevoiți să specificăm variabilele de control ale celor trei bucle; în ordine inversă (v. linia 50).

Cit privește rezultatele execuției programului (v. figura 4.9) semnalăm și-n acest caz „dezordinea” valorilor (nealiniere la punctul zecimal) listate în cele două coloane și în rîndul final.

DENSITATEA BENZINEI=? 0.7	
raza	masa
2	35.1858377
3	118.752202
4	281.486702
5	549.778714
6	950.017619
7	1508.59279
8	2251.89361
9	3206.30946
10	4398.22971

masa totală = 13300.2467 tone	

Fig. 4.9.

Vom încerca în cele ce urmează să eliminăm acest neajuns, uneori supărător, cu ajutorul instrucțiunii **PRINT USING**.

Test

Simulați execuția acestui program

```

10 PRINT
20 N=1
30 FOR I=0 TO 1
40   FOR J=0 TO 1
50     FOR K=0 TO 1
60       FOR L=0 TO 1
70         FOR M=0 TO 1
80           PRINT N; I; J; K; L; M
90           N=N+1
100        NEXT M
110       NEXT L
120     NEXT K
130   NEXT J
140 NEXT I
150 END

```

Aplicații

- a) Fiind date coordonatele mai multor perechi de puncte (acest număr se citește ca parametru) $M(X_1, Y_1)$, $N(X_2, Y_2)$ și $P(X, Y)$ mijlocul segmentului (MN) să se determine distanța $MP=NP$.

```

10 PRINT
20 PRINT "NUMĂR PERECHI PUNCTE";
30 INPUT N
40 FOR I=1 TO N
50   PRINT "PUNCTUL M";

```

```

60 INPUT X1, Y1
70 PRINT "PUNCTUL N:";
80 INPUT X2, Y2
90 LET X=(X1+X2)/2
100 LET Y=(Y1+Y2)/2
110 LET D=((X1-X)↑2+(Y1-Y)↑2)↑0.5
120 PRINT "DISTANȚA MP="; D
130 NEXT I
140 END

```

- b) Să se calculeze momentul static al suprafeței hașurate din figura 4.10 în raport cu axa OZ. Se vor realiza pseudocodul și schema logică pentru domeniul de valori $[-H/2, H/2]$ ale lui Y.

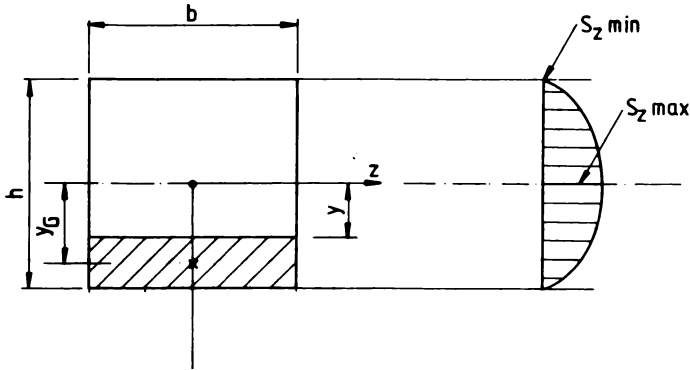


Fig. 4.10.

Pentru determinarea momentului static S_z se utilizează relația de calcul:

$$S_z = Y_G \cdot A \quad (3)$$

unde Y_G și A reprezintă ordonata centrului de greutate respectiv aria zonei hașurate.

Se observă ușor că:

$$Y_G = y + \frac{\frac{h}{2} - y}{2} \quad (4)$$

$$A = b \left(\frac{h}{2} - y \right) \quad (5)$$

rezultînd:

$$S_z = b \left(\frac{h}{2} - y \right) \left(y + \frac{\frac{h}{2} - y}{2} \right) = \frac{b}{2} \left(\frac{h^2}{4} - y^2 \right) \quad (6)$$

Observație: Din relația (6) rezultă că S_z variază parabolic cu y , avînd cea mai mare valoare pentru $y=0$ ($S_{z \max} = \frac{bh^2}{8}$). Pentru valorile $y = \frac{h}{2}$ și $y = -\frac{h}{2}$ valoarea momentului S_z este nulă.

În figura 4.11 (a, b) se prezintă pseudocodul și schema logică ale aplicației.

PSEUDOCODUL	
MOMENT	SEQ
	PRINT "B, H";
	INPUT B, H
A	PENTRU Y DE LA -H/2 LA H/2
	$S=B * (H \uparrow 2/4 - Y \uparrow 2)/2$
	PRINT S
A	SFIRȘIT
MOMENT	END

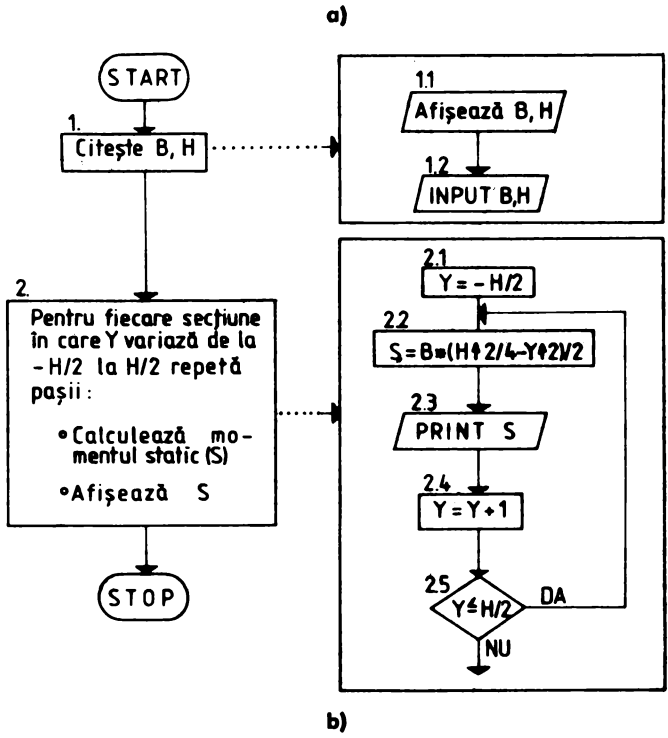


Fig. 4.11. a) pseudocodul; b) schema logică.

```

10 INPUT "B, H"; B, H
20 FOR Y=-H/2 TO H/2
30 S=B*(H*H/4-Y*Y)/2
40 PRINT "Y= "; Y; "S= "; S
50 NEXT Y
60 END
    
```

Editarea valorilor numerice. PRINT USING

În limbajul BASIC, noțiunea de editare trebuie înțeleasă ca pe o suprimare, inserare sau înlocuire de caractere ce aparțin unui anumit câmp. Desigur, pot fi realizate programe BASIC și fără utilizarea clauzelor de

editare, dar cum majoritatea programelor (BASIC) generează în mod frecvent rapoarte tipărite, iar clauzele de editare asigură o mai ușoară manipulare a sarcinilor de suprimare, inserare sau înlocuire a caracterelor, pare lipsit de sens să procedăm altfel. Se cunosc două categorii de caractere de editare: numerice și alfanumerice. Cea mai folosită este prima categorie de caractere.

Instrucțiunea care permite utilizatorului definirea unui format propriu de editare este **PRINT USING** al cărei format general este:

Format general
PRINT [# <nr. canal>] [<listă>] [;]
USING <model format> [<separator> <expresie>]

De notat că <model format> reprezintă orice constantă, variabilă sau expresie numerică, de tip șir caractere, admise de limbaj. Valoarea șirului reprezintă modul în care se va face editarea elementelor din parametrul <listă>. Ca separator poate fi utilizat fie " , " fie " ; ".

Pentru editarea valorilor numerice, formatul de editare furnizează informații referitoare la: numărul de cifre, poziția punctului zecimal, includerea în reprezentare a anumitor simboluri, formatul exponențial.

În tabelul 4.2 sînt prezentate formatele numerice pentru editarea cu instrucțiunea **PRINT USING**, în limbajul BASIC AMSTRAD.

Tabelul 4.2

Indicator de câmp	Funcțiune	Exemplu
0	1	2
#	Fiecare "#" indică amplasarea unei cifre.	10 PRINT USING "####", A
.	Indică poziția punctului zecimal (echivalent cu virgula noastră).	10 PRINT USING "#####.#", A
,	(Rezervă o poziție). Acest simbol neputînd figura decît imediat înaintea punctului zecimal, ne precizează că cifrele situate la stînga punctului zecimal vor fi dispuse în grupe de trei separate între ele printr-o virgulă.	10 PRINT USING "#####.##"; A
##	(Rezervă două poziții). Indică plasarea acestuia imediat înaintea primei cifre sau a punctului zecimal, altfel spus, pe unul din spațiile rezervate cifrelor.	10 PRINT USING "#######.##"; A
**	(Rezervă două poziții). Indică completarea cîmpurilor caracter terminal neocupate, existente la stînga punctului zecimal cu asteriscuri.	10 PRINT USING "**#####.##", A
#	(Rezervă trei poziții). Combină opțiunile ** și ##.	10 PRINT USING "*#####.##", A

0	1	2
\$\$	(Rezervă două poziții). Indică plasarea acestuia înaintea primei cifre sau a punctului zecimal.	10 PRINT USING "\$\$#####.##", A
\$	(Rezervă trei poziții). Combină opțiunile ** și \$.	10 PRINT USING "#####.##", A
+	Indică faptul că se dorește să se figureze semnul numărului. Acest semn va apare înaintea numărului dacă "+" este situat la început de model format și după număr dacă el este plasat la urmă.	10 PRINT USING "#####.##", A
-	Indică faptul că plasarea semnului unui număr la editare se va face după acesta.	10 PRINT USING "#####-.", A
↑↑↑↑	Indică editarea valorii numerice în format exponențial.	10 PRINT USING "#.#####↑↑↑↑", A

Remarci

- Formatul de editare este tratat și interpretat caracter cu caracter;
- Caracterele care nu au semnificații deosebite în formatul de editare sint introduse identic în linia terminal;
- Lungimea maximă pentru (model format) a unui număr este de 20 caractere;
- Dacă formatul de editare este mai mic decât lungimea numărului care se dorește extras, acesta va apare conform formatului precedat de simbolul "%".

Revenind la problema noastră, vă invităm să urmăriți în liniile 80 și 100 ale programului BASIC-AMSTRAD modul în care s-a realizat editarea valorilor numerice pentru toate cele trei cîmpuri: raza, masa și masa totală.

În figura 4.12 se prezintă rezultatele execuției programului (noua versiune).

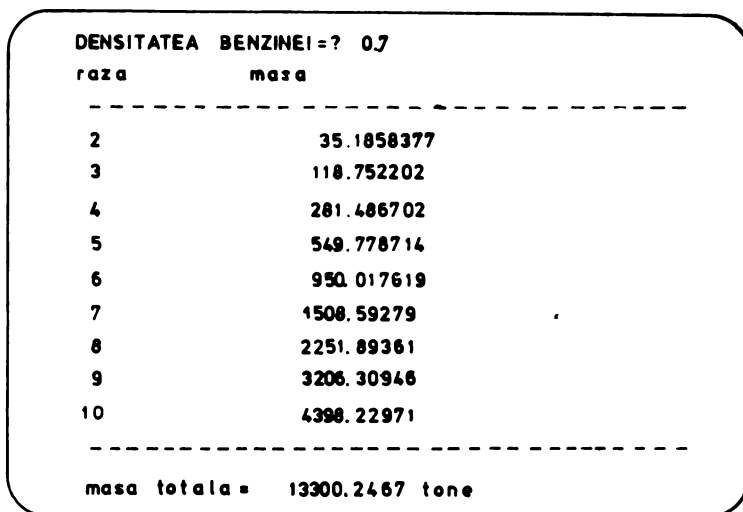


Fig. 4.12.

Încercați fără ajutorul nostru, să înțelegeți bine semnificația formatorilor de editare confruntându-le direct cu rezultatele obținute.

TEST

Precizați rezultatul execuției următoarelor programe:

a) 5 B=3
10 FOR A=-3 TO B STEP 3
20 PRINT A
30 NEXT A
40 END

b) 5 A=4 : B=-5 : C=-2
10 FOR I=A TO B STEP C
20 PRINT I;
30 NEXT I
40 END

c) 10 S=0
20 FOR I=1 TO 100
30 S=S+I
40 I=I+1
50 NEXT I
60 PRINT "S= "; S
70 END

d) 10 P=1 : S=0
20 FOR I=5 TO 23
30 S=S+I
40 P=P*I
50 NEXT I
60 PRINT "P/S"; P/S
70 END

e) 5 INPUT "A="; A
10 FOR I=1 TO A STEP 2
20 PRINT I;
30 NEXT I
40 END

f) 5 INPUT "A, B, C="; A, B, C
10 FOR I=A TO B STEP C
20 PRINT I;
30 NEXT I
40 END

Observație. Se vor introduce dinamic valorile 8, -4, -5 corespunzătoare variabilelor A, B respectiv C.

g) 5 INPUT "C="; C
10 FOR A=-2 TO -11 STEP C
20 PRINT A,

30 NEXT A
40 END

Observație. Se va introduce pentru C valoarea 4.

h) 10 INPUT "N="; N
20 FOR I=1 TO N
30 PRINT "="

40 NEXT I
50 END

Observație. Se va introduce pentru N valoarea 4.

i) 10 FOR I=33 TO 63 STEP 10
20 PRINT X; TAB 10; X ↑ 0.5
30 NEXT I
40 END

60 NEXT I
70 END

j) 10 PRINT "X"; TAB (8); "Y"; TAB
(15); "X+Y"
20 PRINT
30 FOR I=1 TO 5
40 INPUT "X, Y"; X, Y
50 PRINT X; TAB (8); Y; TAB
(15); X+Y
60 NEXT I
70 END

m) 10 FOR I=1 TO 10
20 PRINT I; TAB (5); I ↑ 2;
TAB (12); I ↑ 3
30 NEXT I
40 END

n) 10 FOR I=1 TO 10
20 FOR J=1 TO N
30 PRINT "A";
40 NEXT J
50 PRINT
60 NEXT I

k) 10 FOR N=1 TO 10
20 PRINT N, N ↑ (-2)
30 NEXT N
40 STOP

o) 10 A=1
20 FOR I=1 TO 5
30 A=2*A
40 FOR J=1 TO A
50 PRINT "C";
60 NEXT J
70 PRINT
80 NEXT I
90 STOP

l) 10 FOR I=1 TO 10
20 N=1
30 S=I * I
40 C=I ↑ 3
50 PRINT N; TAB (5); S; TAB
(12); C

```

p) 10 FOR I=1 TO 21 STEP 2
    20 A=I*I
    30 B=A*I
    40 PRINT I;
    50 PRINT "      "; A;
    60 PRINT "      "; B
    70 NEXT I
    80 STOP

r) 10 FOR I=5 TO 7
    20 FOR K=1 TO 10
    25 P=K*I
    30 PRINT K; "X"; I; "="; P
    40 NEXT K
    50 PRINT
    60 NEXT I

s) 10 INPUT "LUNGIMEA="; L
    20 INPUT "LAȚIMEA="; B
    30 FOR I=1 TO L
    40 FOR J=1 TO B
    50 PRINT "X";
    60 NEXT J
    70 PRINT
    80 NEXT I
    90 STOP

```

Observație. Pentru L și B se vor introduce valorile 4 și 9, (vezi s).

```

t) 10 FOR I=1 TO 32
    20 PRINT "-";
    30 NEXT I
    40 STOP

u) 10 FOR I=1 TO 32
    20 PRINT "-";
    30 NEXT I

35 PRINT
40 LET S=14
50 PRINT TAB (S); "(=)"
55 PRINT
60 FOR I=1 TO 32
70 PRINT "-";
80 NEXT I

```

Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag 207

La scrierea programului s-au avut în vedere particularitățile de programare privind instrucțiunile **INPUT** și **FOR-NEXT**. În limbajul BASIC-COMMODORE instrucțiunea **INPUT** (al cărui format general este identic cu cel de la BASIC-AMSTRAD) prezintă următoarele **particularități**:

- citește datele de la canalul identificat prin (nr. canal) (dacă nu se precizează se consideră automat tastatura);
- dacă instrucțiunea conține (șir caractere) acesta este afișat înaintea cererii de date, urmat de "?";
- dacă se răspunde cu mai puține date se afișează pe monitor două semne de întrebare;
- dacă se răspunde cu mai multe date se tipărește mesajul:
? EXTRA IGNORED
- dacă se răspunde cu date de tip necorespunzător se afișează mesajul:
? REDO FROM START

În cadrul programului instrucțiunea **INPUT** ocupă linia 10.

În ceea ce privește utilizarea buclei **FOR-NEXT** menționăm că limbajul BASIC-COMMODORE admite următoarele facilități:

- dacă pe **NEXT** nu este specificat nici un nume de variabilă, el corespunde primului **FOR** care îl precede;
- pot fi imbricate mai multe cicluri **FOR-NEXT** cu condiția ca acestea să nu se intersecteze iar variabilele de control corespunzătoare să fie unice.

În program bucla **FOR-NEXT** ocupă liniile 40–85.

Observație. Limbajul BASIC-COMMODORE nu admite facilități (**PRINT-USING**) privind editarea valorilor numerice și nenumerice.

TEST

Precizați valoarea lui S ("S=") în urma execuției programului:

10 S=0	RUN
20 FOR I=1 TO 4	A, B ? 1, 2
30 INPUT "A, B"; A, B	A, B ? 2, 3
40 S=S+A * B	A, B ? 3, 4
50 NEXT I	A, B ? 4, 5
60 PRINT	S =
70 PRINT "S="; S	
80 END	

**Particularități ale programării
în limbajul BASIC-80**

vol. 2, pag. 207

Instrucțiunea INPUT

În limbajul BASIC-80, instrucțiunea **INPUT**, al cărei format general este:

Format general
<nr. linie> INPUT [;] [(șir caractere);] <listă variabile>

prezintă următoarele **particularități**:

- dacă instrucțiunea conține (șir caractere), acesta este afișat înaintea cererii de date, urmat de "?";
- pentru a suprima caracterul "?" se poate folosi, ",", " după (șir caractere) sau "; " după **INPUT**;
- datele introduse se atribuie variabilelor din (listă variabile) în ordine;
- dacă se răspunde cu mai multe sau mai puține date sau de tip necorespunzător se afișează mesajul:

? Redo from start.

În cadrul programului instrucțiunea **INPUT** ocupă linia 60.

Instrucțiunea FOR

În limbajul BASIC-80, bucla **FOR-NEXT** prezintă următoarele **particularități**:

- ciclurile **FOR-NEXT** pot fi suprapuse (imbricate) dar trebuie să nu se intersecteze și să aibă variabile de control diferite;
- atribuirea variabilei de control lui **NEXT** este facultativă;
- o instrucțiune **NEXT** fără variabilă este asociată cu ultima instrucțiune **FOR**.

În program instrucțiunea **FOR-NEXT** ocupă liniile 130, 140, 150, 160, 170, 180.

Odată cu parcurgerea instrucțiunilor cuprinse în ciclul **FOR-NEXT** al programului ați sesizat ceva deosebit la instrucțiunea din linia 140?

```

130 FOR R=2 TO 10
140 V=2 * PI *R ↑ 3
.
.
.
170 PRINT R, M
180 NEXT R
    
```

Cei mai grăbiți dintre dumneavoastră n-au remarcat poate faptul că valoarea lui PI a fost definită în linia 115 într-un mod cu totul nou.

$$115 \text{ PI} = 4 * \text{ATN} (1) : \text{REM "Numărul PI"}$$

Nu vă alarmați, pentru că nu este nimic complicat.

Amintiți-vă din liceu că

$$\arctg 1 = \frac{\pi}{4} \tag{8}$$

sau:

$$\pi = 4 \cdot \arctg 1 \tag{9}$$

În BASIC (80) relația (9) se codifică exact ca în linia 115 cu mențiunea că **ATN(1)** reprezintă funcția aritmetică arc tangentă de 1.

Menționăm, de asemenea, faptul că în cadrul programului nu ne-am preocupat de editarea cu format (v. linia 170) a valorilor numerice (v. fig. 4.13), ceea ce nu înseamnă că dumneavoastră nu puteți realiza acest lucru singuri, ca o aplicație privind utilizarea instrucțiunii **PRINT USING** în limbajul BASIC-80. De notat că formatele numerice de editare prevăd pe lângă indicatorii de cîmp: "#", "+", "-", "##", "\$\$", "\$\$\$", ".", "^ ^ ^ ^" înțelniți la limbajul BASIC-AMSTRAD (v. tabelul 4.2) și indicatorul "\" cu semnificația că următorul caracter din format se afișează.

R a z a	M a s a
2	3 5 . 1 8 5 8
3	1 1 8 . 7 5 2
4	2 8 1 . 4 8 7
5	5 4 9 . 7 7 9
6	9 5 0 . 0 1 8
7	1 5 0 8 . 5 9
8	2 2 5 1 . 8 9
9	3 2 0 6 . 3 1
1 0	4 3 9 8 . 2 3
M a s a t o t a l ă = 1 3 3 0 0 . 2 t o n e	

Fig. 4.13.

Remarcă. Dacă numărul de cifre specificat este mai mare decât 24 se afişează un mesaj de eroare.

TEST

Simulaţi execuţia acestui program:

```

10  FOR I=1 TO 3
20    FOR J=1 TO 3
30      FOR K=1 TO 3
40        PRINT I; J; K
50      NEXT K
60    NEXT J
70  NEXT I
80  END

```

în maniera:

Valorile variabilelor I, J, K			
Instrucţiune	I	J	K
40 PRINT I; J; K	1	1	1
	1	1	2
	1	1	3
	1	2	1
	1	2	2
	1	2	3
	1	3	1
	1	3	2
	1	3	3
	.	.	.
	.	.	.
40 PRINT I; J; K	3	3	3

Aplicaţii

a) Să se editeze următorul triunghi al numerelor:

```

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9 10

```

```

10  FOR I=1 TO 10
20    FOR J=1 TO I
30      PRINT J;
40    NEXT J
50  PRINT
60  NEXT I
70  END

```

b) Determinaţi al n-lea termen dintr-un şir Fibonacci. Se vor realiza tabela de variabile, schema logică şi programul BASIC.

Un şir Fibonacci este un şir de numere întregi în care fiecare număr este egal cu suma celor două numere precedente. Altfel spus, suma primului şi celui de-al doilea număr este egală cu cel de-al treilea număr, suma celui de-al doilea număr şi celui de-al treilea este egală cu al patrulea număr etc.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34.

$0+1$ $1+1$ $1+2$ $2+3$

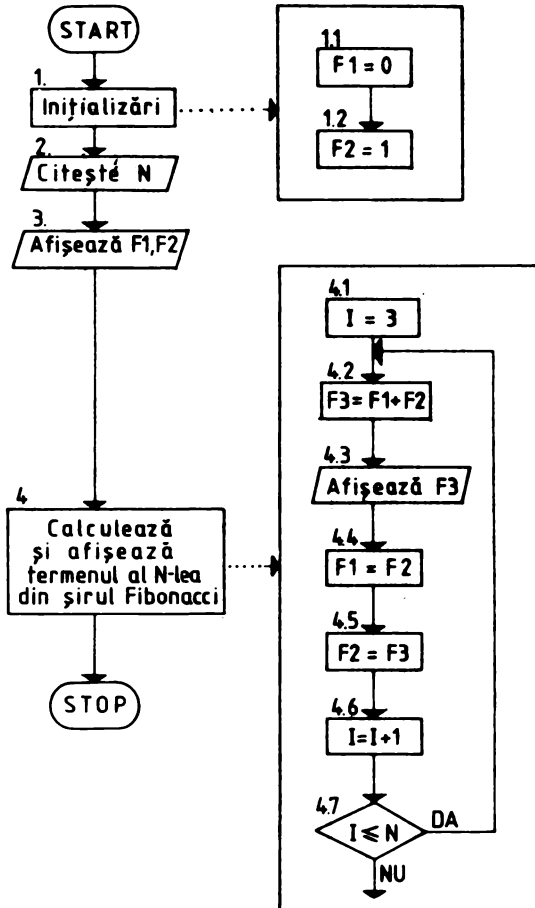
Se va utiliza formula de recurenţă

$$F_N = F_{N-1} + F_{N-2}, \quad N > 2, \quad F_1 = 0, \quad F_2 = 1 \quad (10)$$

În figura 4.14 (a, b) sînt ilustrate tabela de variabile și schema logică ale aplicației.

TABELA DE VARIABILE		
Variabile de intrare	Variabile de stare	Variabile de ieșire
N : rangul termenului căutat	I : variabila de control a structurii de iterație	
	PENTRU	
	F1 : primul număr Fibonacci	F3 : termenul căutat din șirul Fibonacci
	F2 : al doilea număr Fibonacci	

a)



b)

Fig. 4.14. a) tabela de variabile; b) schema logică.

□ Particularități ale programării în limbajul BASIC-PLUS

vol. 2, pag. 208

Programul pe care-l analizăm acum găzduiește pe lângă cunoștințele noastre mai vechi:

REM (v. liniile 100–150, 170, 220–270, 290, 300, 370), cei doi amici nedes-părțiți **FOR** și **NEXT** (v. liniile 200, 204, 240, 290), **INPUT** (v. linia 160), numeroșii **PRINT**, **PRINT USING** (v. linia 280) și invitați de „onoare”, cum sînt: modificatorul **FOR** (v. liniile 310, 360, 370), variabila alfanumerică **A\$** (v. linia 300) ș.a.

Considerăm că nu greșim prea mult dacă vom începe analiza prin a lista particularitățile prietenilor noștri mai de demult: **INPUT** și **FOR-NEXT**; lăsîndu-i la urmă pe cei noi, ca fiind cei mai importanți.

Instrucțiunea INPUT

În limbajul BASIC-PLUS instrucțiunea **INPUT** al cărei format general este:

Format general
INP [UT] [(constantă șir),] (variabilă) {, (variabilă) }

prezintă următoarele particularități:

- Cînd într-o instrucțiune **INPUT** se specifică (constantă șir) aceasta se editează pe linia terminal înaintea simbolului " ? ";
- Dacă numărul datelor introduse este mai mare decît numărul variabilelor specificate în instrucțiunea **INPUT**, datele în plus se neglijează;
- Dacă numărul datelor introduse este mai mic decît numărul variabilelor specificate în instrucțiunea **INPUT**, se va afișa la terminal simbolul " ? " așteptîndu-se introducerea valorilor lipsă;
- Dacă tipul datei introduse de la terminal nu corespunde cu tipul variabilelor specificate în instrucțiunea **INPUT** se va genera caracterul "R", după care utilizatorului îi revine sarcina de a reintroduce datele considerate eronate.

În figura 4.15 este ilustrat modul în care are loc execuția instrucțiunii **INPUT** (v. linia 160 din cadrul programului).

```

:RUN
-DENSITATEA BENZINEI =? 0.7

```

Fig. 4.15.

Bucła FOR-NEXT. Particularități

În limbajul BASIC-PLUS instrucțiunile **FOR...TO** și **NEXT** prezintă următoarele **particularități**:

- variabila de control a ciclului poate fi orice variabilă numerică simplă de tip întreg sau real;
- variabila de control poate fi modificată în interiorul ciclului;
- este posibil transferul controlului în exteriorul buclei înainte de terminarea execuției buclei prin intermediul instrucțiunilor **GO TO, ON ... GOTO, IF**;
- controlul poate fi transferat în interiorul unui ciclu, dar acesta se va manifesta diferit, funcție de realizarea sau nu a inițializării ciclului;
- **FOR-NEXT** nu poate să apară pe o linie imediată sau ca parte componentă a unei instrucțiuni **IF**;
- ciclurile **FOR-NEXT** pot fi imbricate (suprapuse); numărul ciclurilor suprapuse depinde de memoria disponibilă a utilizatorului;
- restricțiile ciclurilor suprapuse sînt: a) fiecare buclă trebuie să aibă **FOR-NEXT**-ul și variabilele de control proprii; b) buclele nu trebuie să se intersecteze.

În program buclele **FOR-NEXT** cuprind liniile 200, 202, 204 pentru tipărirea capului de tabel și liniile 230, 240, 250, 260, 270, 280, 290 pentru calculul masei de benzină din rezervoare.

Cît privește prima buclă **FOR**

```
200 FOR I=1 TO 29
202 PRINT TAB (I); "=";
204 NEXT I
```

remarcați utilizarea lui **TAB (I)** în instrucțiunea **PRINT** (v. linia 202) necesar poziționării și tipării caracterului "=" în fiecare din cele 29 de coloane ale liniei terminal.

În cea de-a doua buclă, în linia 280 am utilizat instrucțiunea **PRINT USING** în scopul editării cu format numeric a valorilor variabilelor **R** și **M**.

Este momentul să acordăm cîteva clipe bune acestei instrucțiuni atît de necesare în editarea valorilor numerice dar și în editarea formatelor nenumerice.

PRINT USING

În limbajul BASIC-PLUS, instrucțiunea **PRINT USING** al cărei format general este:

Format general
PR[NT] USING <șir> [,<listă>]

prezintă următoarele **particularități**:

- <șir> reprezintă orice constantă, variabilă sau expresie numerică, de tip șir de caractere, admisă de limbaj;
- <listă> reprezintă o listă de elemente (admise de limbaj) separate între ele prin caracterele " , " sau " ; " ;
- <listă> se poate termina prin caracterul " , " ; " sau spațiu;

- elementele din (listă) sînt parcurse în timpul execuției programului, paralel cu caracterele din (șir);
- separatorii dintre elementele listei nu au efect asupra formatului de editare;
- dacă după ultimul element al listei urmează caracterul ", " sau " ; ", următoarea instrucțiune **PRINT USING** sau **PRINT** va tipări pe aceeași linie terminal;
- formatul de editare este tratat și interpretat caracter cu caracter;
- caracterele care nu au semnificații deosebite în formatul de editare sînt introduse identic în linia terminal.

În tabelul 4.3 sînt prezentate formatele numerice pentru editarea cu instrucțiunea **PRINT USING** în limbajul BASIC-PLUS a valorilor numerice.

Tabelul 4.3

Indicator de cîmp	Funcțiune	Exemplu
#	Precizează că un cîmp caracter din linia terminal va fi ocupat de o cifră zecimală sau spațiu. Indică poziția punctului zecimal.	30 PRINT USING " # # # # # # ", C 5 PRINT USING " # # # . # # ", C
,	Indică inserarea simbolului " , " după fiecare grup de trei cifre, pornind spre stînga punctului zecimal, dacă acesta există.	5 PRINT USING " # # # # . # # # # ", A
* *	Indică completarea cîmpurilor caracter terminal neocupate, existente la stînga punctului zecimal, cu " * ".	5 PRINT USING " * * # # . # # ", B
\$ \$	Indică plasarea simbolului " \$ " înaintea numărului editat și eventuala completare cu spațiu în stînga punctului zecimal.	5 PRINT USING " \$ \$ # . # # # "; A
-	Precizează faptul că plasarea semnului unui număr la editare se va face după acesta.	5 PRINT USING " # # # . # - ", A
^^^	Indică editarea valorii numerice în format exponențial, cele patru simboluri rezervă spațiu pentru: litera E, semnul exponentului, exponent.	5 PRINT USING " # # # . # ^ ^ ^ ^ ", A

În linia 280 a programului am utilizat instrucțiunea **PRINT USING** în scopul editării cu format numeric a valorilor variabilelor R și M.

280 **PRINT USING** "# # # # # # # # #", R ; M

Remarcați modul în care au fost întrebuiți cei doi indicatori de cîmp "# " și ". ". Ați înțeles de ce pentru R=10 se afișează 0 sub 9 și 1 imediat la stînga? Deoarece numărul caracterelor "# " este mai mare decît numărul de cifre ale valorilor (2, 3, 4, 5, 6, 7, 8, 9) de editat, cîmpurile caracter neocupate se completează cu spațiu (v. figura 4.16).

1	# # b b	b b . b b b b b . b b b	sablon pentru formatul de editare
b 2		b b b b b b 3 5 . 1 9	
b 3		b b b b b b 1 1 8 . 7 5	
b 4		b b b b b b 2 8 1 . 4 9	
b 5		b b b b b b 5 4 9 . 7 8	
b 6		b b b b b b 9 5 0 . 0 2	
b 7		b b b b b b 1 5 0 8 . 5 9	
b 8		b b b b b b 2 2 5 1 . 8 9	
b 9		b b b b b b 3 2 0 6 . 3 1	
1 0		b b b b b b 4 3 9 8 . 2 3	
⋮	⋮	⋮	

Fig. 4.16.

Cit privește spațiile (blancurile) din formatul de editare acestea sint introduse identic în linia terminal.

Aplicație. Deasupra centrului unei mese rotunde de 2 m diametru la înălțimea h este atârnată o sursă de lumină de 100 cd. Să se determine iluminarea la marginea mesei cînd h variază în intervalul $0,5 \leq h \leq 0,9$ m (v. fig. 4.17). Se vor alcătui tabela de varia-bile și schema logică.

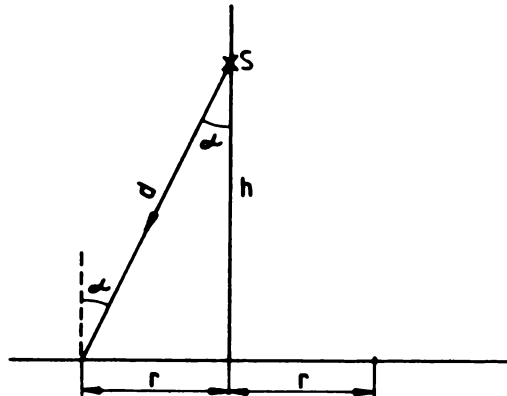


Fig. 4.17.

Iluminarea la marginea mesei este dată de relația

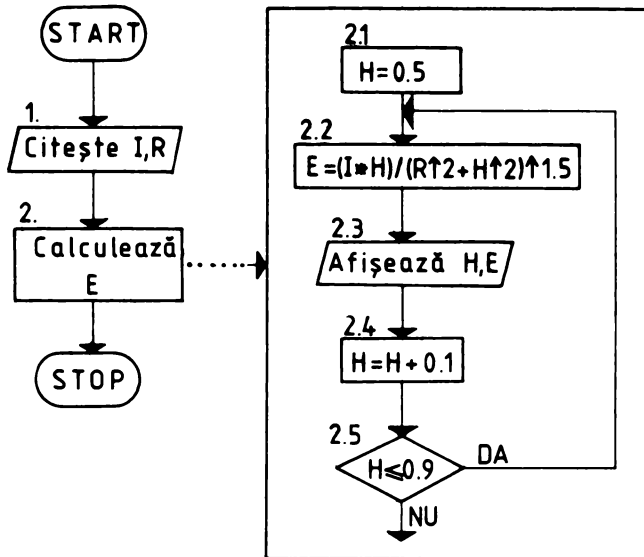
$$E = \frac{I}{d^2} \cos \alpha = \frac{I}{r^2 + h^2} \cdot \frac{h}{\sqrt{r^2 + h^2}} = \frac{Ih}{(r^2 + h^2)^{3/2}} \tag{11}$$

unde I este intensitatea luminoasă a sursei, E iluminarea, h înălțimea la care este situată sursa de lumină, $r = D/2$ (D este diametrul mesei).

Tabela de variabile și schema logică sint prezentate în figura 4.18 (a, b).

TABELA DE VARIABILE		
Variabile de intrare	Variabile de stare	Variabile de ieșire
R : Raza mesei I : intensitatea luminoasă a sursei	H : înălțimea	E : iluminarea la marginea mesei

a)



b)

Fig. 4.18. a-b. a) tabela de variabile; b) schema logică.

TEST

Precizați rezultatele execuției următoarelor programe BASIC-PLUS:

a) 10 A=123 : B=1235 : C=123.9
 20 PRINT USING '###.###', A
 30 PRINT USING '#####', B
 40 PRINT USING '#####', C
 50 END

b) 10 A=304.23 : B=25.77 : C=
 25.779 : D=111.99
 20 PRINT USING '###.###', A
 30 PRINT USING '#####', B
 40 PRINT USING '#####', C
 50 PRINT USING '##.###', D
 60 END

c) 10 A=8.99 : B=89.99 : C=789.99
 20 PRINT USING '**###.###', A
 30 PRINT USING '**#####', B
 40 PRINT USING '**###.###', C
 50 END

d) 10 A=5.842 : B=17.38 : C=17.385
 20 PRINT USING '\$\$.###', A
 30 PRINT USING '\$\$.###', B
 40 PRINT USING '\$\$.###', C
 50 END

e) 10 A=-2.3 : B=3.5
 20 PRINT USING '#.#.#', A
 30 PRINT USING '#.#.#', B
 40 END

f) 10 A=835 : B=12345.67
 20 PRINT USING '#.#.#.#', A
 30 PRINT USING '\$\$##',
 ##.#', B
 40 END

g) 10 A=2 : B=22 : C=222 :
 D=222222 : E=22.22 : F=22.222
 20 PRINT USING '###.# ^^^^', A
 30 PRINT USING '###.# ^^^^', B
 40 PRINT USING '###.# ^^^^', C
 50 PRINT USING '###.# ^^^^', D
 60 PRINT USING '###.# ^^^', E
 70 PRINT USING '###.# ^^^^', F
 80 END

Constante și variabile alfanumerice

A sosit și rindul „invitațiilor”! Să facem cunoștință cu variabila alfanumerică A\$ (v. linia 300).

300 A\$="="! INIT. VARIABILA ALFA

Puțină teorie! Limbajul BASIC permite manipularea nu numai a informațiilor numerice ci și a celor nenumerice, recunoscute sub forma șirurilor de caractere (o succesiune de litere, cifre sau caractere speciale nu mai multe de 255).

Constanta șir de caractere este formată dintr-un șir de caractere inclus între simbolurile " " sau " "" ". Cât privește *variabilele simple nenumerice* acestea sînt folosite pentru definirea, identificarea sau referirea informației nenumerice de tip șir de caractere. Tipul nenumeric al unei variabile șir de caractere se indică prin simbolul "\$" adăugat numelui acesteia. Așadar, A\$ – noua noastră cunoștință din linia 300 este o variabilă simplă nenumerică (variabila șir de caractere) căreia i s-a atribuit constanta șir de caractere "="".

Observație. Într-un program BASIC-PLUS pot exista variabile simple numerice și variabile simple nenumerice cu același nume. Altfel spus, dacă într-un program BASIC-PLUS folosim variabilele A (de tip real), A% (de tip întreg), A\$ (de tip șir de caractere) sigur nu facem nici o greșeală.

TEST

Simulați execuția următorului program:

```

10 A$="BASIC"
20 B$="INVATAȚĂM"
30 C$=" . . . "
40 PRINT
50 PRINT B$, A$, C$

60 PRINT
70 PRINT B$; A$; C$
80 PRINT
90 PRINT B$; " "; A$; " "; C$
100 STOP
```

Completare la instrucțiunea LET

Instrucțiunea de atribuire **LET** afectează (atribuie) o valoare numerică sau un șir de caractere unei variabile simple numerice sau nenumerice (alfanumerice) sau unei variabile indexate.

Formatul general al instrucțiunii este

Format general

[LET] < variabilă > { , < variabilă > } = < expresie >

De notat că variabila din stînga semnelui egal și expresia din dreapta trebuie să fie de același tip (numeric sau nenumeric). Și încă un amănunt de loc de neglijat: instrucțiunea **LET** permite în limbajul BASIC-PLUS o atribuire multiplă.

Remarcă. Instrucțiunea de atribuire din linia 300 poate fi folosită în toate versiunile BASIC din această lucrare.

Patru metode de generare a unui rînd de subliniere

Pentru generarea unui rînd cu 29 de simboluri "=" folosit în sublinierile din lista cu rezultate s-au folosit patru metode (v. figura 4.19) – pentru fiecare rînd de subliniere cite o metodă. Astfel, pentru primul rînd necesar trasării liniei de început a capului de tabel s-a utilizat instrucțiunea

```
180 PRINT "===== "
```

Metoda este incomodă din motive evidente.

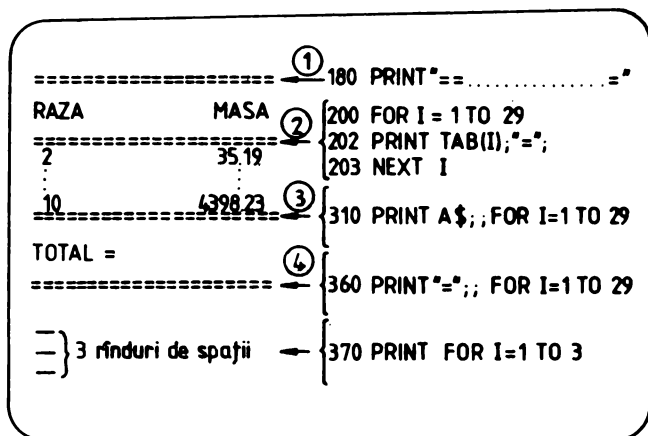


Fig. 4.19.

Cea de-a doua linie cu "=" a capului de tabel a fost trasată mult mai elegant apelînd la bucla **FOR-NEXT** care iterează instrucțiunea **PRINT TAB** din linia 202 de 29 de ori.

```
200 FOR I=1 TO 29
202 PRINT TAB (I); "=";
203 NEXT I
```

Observație. Remarcați prezența separatorului ";" la sfîrșitul instrucțiunii **PRINT** necesar tipăririi pe același rînd a caracterului "=".

Ultimele două linii ale listei au fost generate cu două instrucțiuni (v. liniile 310 și 360) care la prima vedere, de ce să nu recunoaștem, par a fi puțin ciudate! Este vorba de **modificatorul FOR**. De ce modificatorul **FOR**? vă veți întreba pe bună dreptate.

Modificatorul FOR

Desigur că există un răspuns și la această întrebare. Utilizarea modificatorilor de instrucțiuni permite o mai bună flexibilitate în exprimarea unei condiții. Limbajul BASIC-PLUS cunoaște cinci modificatori: **IF, UNLESS, WHILE, UNTIL, FOR.**

Modificatorul **FOR** (cunoștința noastră!) creează o buclă de o instrucțiune avînd o variabilă de control indicată.

Formatul general al modificatorului **FOR** este:

Format general	
< nr. linie >	< instrucțiune > FOR < variabilă > = < exp. 1 > TO < exp. 2 > [STEP < exp. 3 >]

Remarci

- Parametrii instrucțiunii sint aceeași cu cei de la instrucțiunea **FOR.**
- Modificatorul **FOR** se aplică numai instrucțiunii care îl precede.
- O instrucțiune poate fi însoțită de unul sau mai mulți modificatori, acțiunea acestora fiind tratată de la dreapta la stînga.
- Pe o linie program multiplă modificatorul acționează numai asupra instrucțiunii care îl precede.

Revenind la instrucțiunile din liniile 310 și 360

```
310 PRINT A$ ; ; FOR I=1 TO 29
```

```
.
```

```
360 PRINT "=" ; ; FOR I=1 TO 29
```

remarcați cum modificatorul **FOR** este folosit pentru însoțirea instrucțiunii **PRINT** cu singura deosebire că în linia 310 în locul caracterului "=" s-a folosit variabila șir de caractere **A\$** a cărei valoare a fost stabilită în linia precedentă 300 (inițializare variabilă alfanumerică – v. comentariul).

În linia 370 a aceluiași program

```
370 PRINT FOR I=1 TO 3! SARE 3 RINDURI
```

modificatorul **FOR** a fost utilizat pentru tipărirea a trei rinduri albe (v. mesajul **STOP AT LINE 380**).

TEST

Simulați execuția programului cu creionul și hirtia. Veți întâmpina mici dificultăți în special la editarea cu format a valorilor numerice.

Confrunțați în final rezultatul dumneavoastră cu cel dat de calculator. Nu vă grăbiți!

**Particularități ale programării
în limbajul ABASIC**

vol. 2, pag. 209

La scrierea programului s-au avut în vedere particularitățile de programare privind instrucțiunile **INPUT, FOR-NEXT** și **PRINT USING.**

În limbajul ABASIC, instrucțiunea **INPUT** prezintă următoarele particularități:

- dacă nu se specifică (listă variabile), ci numai (șir caractere) se face numai afișarea acestuia (șir caractere);
- numărul de valori separate prin „,“ trebuie să fie cel puțin egal cu numărul de variabile din listă (valorile suplimentare se ignoră).

În cadrul programului instrucțiunea **INPUT** ocupă linia 160. Instrucțiunea **FOR-NEXT** prezintă particularitățile:

- dacă pe **NEXT** nu este specificat nici un nume de variabilă, el corespunde primului **FOR** care îl precede;
- se pot imbrica mai multe cicluri **FOR-NEXT** cu terminarea pe un **NEXT** comun, sau pe **NEXT**-uri diferite;
- ciclurile imbricate nu trebuie să se intersecteze iar variabilele de control corespunzătoare trebuie să fie unice;
- **FOR-NEXT** acceptă opțiunea **WHILE**, condiția testându-se după testul variabilei de control.

În program bucla **FOR-NEXT** ocupă liniile 240–290.

Cît privește utilizarea instrucțiunii **PRINT USING** (v. linia 280) menționăm că limbajul ABASIC admite specificații de format atît pentru numere cît și pentru șiruri.

Simbolurile folosite pentru editarea valorilor numerice sînt:

“#”, “-”, “\$\$”, “^^^^”.

Aplicație. Scrieți un program BASIC care să tipărească următoarele combinații: 00, 01, 10, 11,

```

10 FOR I=0 TO 1           40 NEXT J
20 FOR J=0 TO 1           50 NEXT I
30 PRINT I; J             60 END

```

TEST

Precizați rezultatul execuției următorului program:

```

40 PRINT "50 FOR I=1 TO 5"
50 FOR I=1 TO 5
60 PRINT I;
70 NEXT I
80 PRINT
90 PRINT "100 FOR I=1 TO
19 STEP 2"
100 FOR I=1 TO 19 STEP 2
110 PRINT I;
120 NEXT I
130 PRINT
140 PRINT "150 FOR I=345 TO
282 STEP-9"
150 FOR I=345 TO 282 STEP-9
160 PRINT I;

170 NEXT I
180 PRINT
190 PRINT "200 FOR I=91.5
TO 3 STEP - 15.7"
200 FOR I=91.5 TO 3 STEP - 15.7
210 PRINT I;
220 NEXT I
230 PRINT
240 PRINT "270 A=5 : B=45 : C=6"
250 PRINT "280 FOR I=A TO B
STEP C"
270 A=5 : B=45 : C=6
280 FOR I=A TO B STEP C
290 PRINT I;
300 NEXT I

```

TEMA 4

□ Răspundeți prin DA sau NU la următoarele întrebări:

- Blocul **PENTRU ... SFIRȘIT** codifică o structură algoritmică de iterație cu număr cunoscut de pași.
- Instrucțiunea **PENTRU** reprezintă punctul unic de intrare în blocul de iterație.
- Instrucțiunea **SFIRȘIT** reprezintă punctul unic de ieșire din blocul de iterație.
- Instrucțiunile **FOR** și **NEXT** apar în linii diferite în cadrul programului.
- Într-o buclă **FOR-NEXT** este obligatoriu ca variabila de control să ia valori începând cu 1.
- Într-o instrucțiune **FOR** parametrul **STEP** este obligatoriu.
- În absența opțiunii **STEP** pasul de incrementare este 1.
- Variabila din instrucțiunea **NEXT** este diferită de variabila din instrucțiunea **FOR**.
- În BASIC-aMIC variabila din instrucțiunea **NEXT** poate să lipsească.
- În interiorul unei bucle **FOR-NEXT** din limbajul BASIC-aMIC se pot imbrica patru bucle.
- În BASIC-aMIC ciclurile suprapuse se pot intersecta.
- Dacă valoarea limită a variabilei de control este depășită, execuția buclei **FOR-NEXT** ia sfârșit, următoarea instrucțiune ce urmează a fi executată fiind cea de după **NEXT**.
- Valoarea variabilei de control a unei bucle **FOR-NEXT** poate să dească cu valori fracționare.
- În limbajul BASIC-aMIC instrucțiunile **PRINT** și **INPUT** nu pot fi combinate într-una singură - **INPUT**.

Secvențele de instrucțiuni BASIC de mai jos sînt scrise corect:

```

- 10 FOR I=1 TO 20
      I=I+1
90 NEXT I

```

```

- 10 FOR A=0 TO 40 STEP 4
      FOR B=0 TO A
150   NEXT B
200 NEXT A

```

```

- 10 FOR I=1 TO M
      FOR K=1 TO N
60   NEXT I
90 NEXT K

```

```

- 10 FOR A=0 TO 1 STEP 0.002
      FOR A=0 TO 8 STEP 0.5
180 NEXT A

```

```

- 10 FOR B=I TO (C+D) STEP L
90 NEXT L

```

```

- 10 FOR A=B TO 4 * (C-D)
      STEP L
- 10 FOR I=0 TO - 80 STEP - 6
- 10 FOR I=J TO 80 STEP J
- 10 FOR A$=1 TO 100 STEP 2
- 30 FOR I=L TO A STEP I
- 10 FOR A=1 TO B STEP L
90 NEXT L

```

```

- 10 FOR I=1 TO A           100     FOR R=1 TO 50
    50   FOR J=1 TO 100     180     NEXT R
    80   FOR K=1 TO M       300     NEXT J
    90   NEXT K
                                           900 NEXT I

```

– Instrucțiunea

```
10 INPUT "A, B, C"; A, B, C
```

este acceptată de limbajul BASIC-PRAE.

– În BASIC-PRAE variabila de control a buclei **FOR-NEXT** nu poate fi modificată.

– În BASIC-PRAE pot fi imbricate oricâte cicluri **FOR-NEXT** dorim cu condiția ca acestea să nu se intersecteze.

Instrucțiunea

```
20 INPUT "A, B" ; A, B
```

nu este acceptată de calculatorul HC-85.

– În BASIC-SPECTRUM variabila de control a buclei **FOR-NEXT** trebuie să aibă numele format dintr-o singură literă.

– În BASIC-AMSTRAD orice neconcordanță dintre numele variabilelor și tipul datelor este semnalată prin mesajul

? Redo from start

– În BASIC-AMSTRAD, **NEXT**-ul poate fi ... anonim (fără variabilă).

– În BASIC-AMSTRAD se pot realiza programe utilizând clauzele de editare.

– Instrucțiunea care permite utilizatorului definirea unui format propriu de editare este **PRINT USING**.

– Indicatorul de câmp "#" este folosit pentru a reprezenta poziția fiecărei cifre a numărului.

– Instrucțiunea BASIC-AMSTRAD

```
10 PRINT USING "##.##" 4.3, 5.789
```

afișează: 4.30 5.79

– Instrucțiunea BASIC-AMSTRAD

```
10 PRINT USING "***$.##.##" ; 8.25
```

afișează: ***\$8.25

– instrucțiunea BASIC-AMSTRAD

```
10 PRINT USING "#### ↑↑↑"; 77777
```

afișează .7779E06

– Instrucțiunea BASIC-COMMODORE

```
10 PRINT USING "####", 3856
```

afișează 3856.

– În BASIC-80 caracterul "?" generat de instrucțiunea **INPUT** poate fi șters cu ",", " sau ";".

– În BASIC-80 atribuirea variabilei de control lui **NEXT** este obligatorie.

– Instrucțiunea BASIC-80

```
10 PI=4 * ATN (1)
```

calculează numărul π .

– Indicatorul de cîmp "\ " este specific formatelor numerice de editare din BASIC-80.

– În BASIC-PLUS variabila de control a ciclului **FOR-NEXT** poate fi orice variabilă numerică simplă de tip întreg sau real.

– Variabila de control a unei bucle **FOR-NEXT** din BASIC-PLUS poate fi modificată în interiorul ciclului.

– Secvența de instrucțiuni BASIC

```
10 FOR I=1 TO 14
20 PRINT TAB (I); " * ";
30 NEXT I
```

tipărește un rînd alcătuit din 12 asteriscuri.

– Secvența de instrucțiuni BASIC-PLUS

```
20 PRINT " * " ; ; FOR I=1 TO 14
```

tipărește un rînd alcătuit din 13 asteriscuri.

– Secvența de instrucțiuni BASIC-PLUS

```
10 B$=" * "
20 PRINT B$ ; ; FOR I=1 TO 14
```

tipărește un rînd cu 14 asteriscuri.

– Secvența de instrucțiuni BASIC-PLUS

```
10 A=67801.78
20 PRINT USING '$$##.##.##', A
```

tipărește valoarea \$67, 801.8.

– În limbajul ABASIC se pot imbrica mai multe cicluri **FOR-NEXT** cu terminarea pe un **NEXT** comun sau pe **NEXT**-uri diferite.

– Limbajul ABASIC admite specificații de format pentru valori numerice și nenumerică.

Inlocuiți cuvintele care lipsesc din următoarele propoziții:

- a) Instrucțiunea **FOR** reprezintă punctul de în ciclu iar instrucțiunea **NEXT** reprezintă punctul de din ciclu.
- b) Într-o instrucțiune **FOR** valoarea variabilei de control poate să crească fie cu valori fie cu valori
- c) Rolul opțiunii **STEP** este
- d) În BASIC-AMIC numărul maxim de cicluri imbricate este pe cînd în BASIC-PRAE numărul maxim de cicluri imbricate este
- e) În limbajul BASIC-PLUS instrucțiunile **PRINT** și **INPUT**
.....fi cuplate într-una singură pe cînd în limbajul BASIC-AMICfi cuplate într-o instrucțiune **INPUT**.
.....pot/nu pot
.....pot/nu pot
- f) În BASIC variabila de control a buclei **FOR-NEXT** poate fi modificată în interiorul instrucțiunii.
- g) În BASIC ciclurile suprapuse se pot intersecta.
- h) În BASIC-AMSTRAD variabila de control a buclei **FOR-NEXT** trebuie să aibă numele format din litere pe cînd în BASIC TIM S numele variabilei de control este format
- i) Instrucțiunea **PRINT USING** poate fi utilizată în BASIC dar nu poate fi utilizată în BASIC
- j) În BASIC atribuirea variabilei de control nu este obligatorie.
- k) În BASIC-80 simbolurile folosite pentru editarea variabilelor numerice sînt
- l) În BASIC-AMSTRAD simbolurile folosite pentru editarea valorilor numerice sînt iar în BASIC-PLUS sînt
- m) Pentru editarea unui rînd cu 7 vapoaze "**<=>**" se folosește secvența de instrucțiuni BASIC sau
sau sau
sau sau

Să se scrie cîte un program BASIC pentru fiecare din problemele de mai jos:

- a) Aplicațiile de la paginile 190, 195 (v. tabela de variabile și schemele logice realizate).
- b) Să se calculeze lungimile de undă ale spectrului emis de atomul de hidrogen, folosind formula lui Balmer.

c) Să se editeze în BASIC HC-85, TIMS, SPECTRUM textul de mai jos:

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0
I	N	V	A	T	A	M			B	A	S	I	C					.	.	.
I	N	V	A	T	A	M	B	A	S	I	C	.	.	.						
I	N	V	A	T	A	M		B	A	S	I	C		.	.	.				

d) Să se calculeze suma parțială a seriei

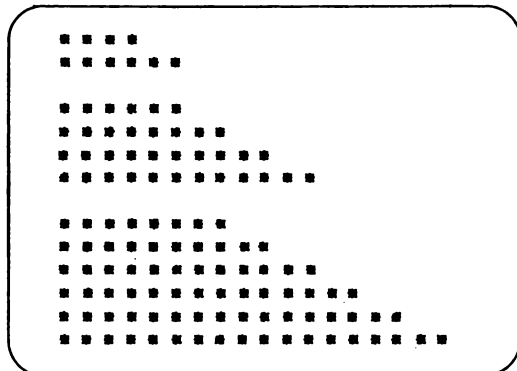
$$S=1-\frac{1}{4}+\frac{1}{9}+\dots+(-1)^{N+1}/N^2, \text{ pentru } N \text{ dat.}$$

e) Să se tipărească următorul triunghi al numerelor:

1																				
1	1																			
1	2	1																		
1	3	3	1																	
1	4	6	4	1																
1	5	10	10	5	1															
1	6	15	20	15	6	1														
1	7	21	35	35	21	7	1													
1	8	28	56	70	56	28	8	1												
1	9	36	84	126	126	84	36	9	1											
1	10	45	120	210	252	210	120	45	10	1										

f) Să se calculeze rezistența echivalentă a N rezistențe legate în:
a) serie; b) paralel.

g) Să se editeze următoarele blocuri de asteriscuri:



Schema logică este prezentată în figura 4.20.

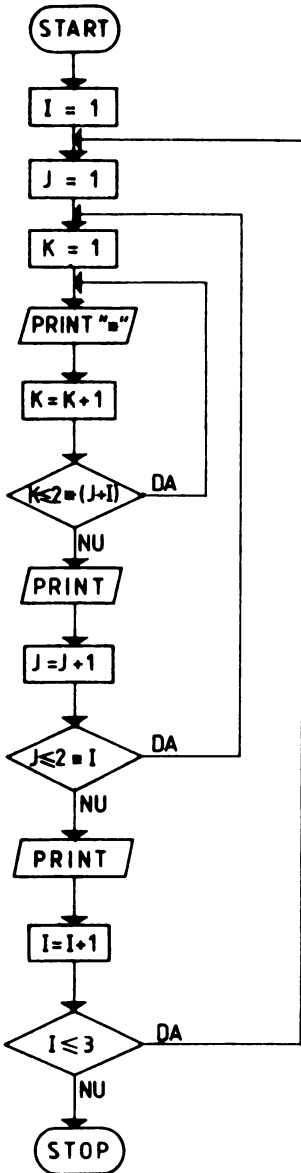


Fig. 4.20.

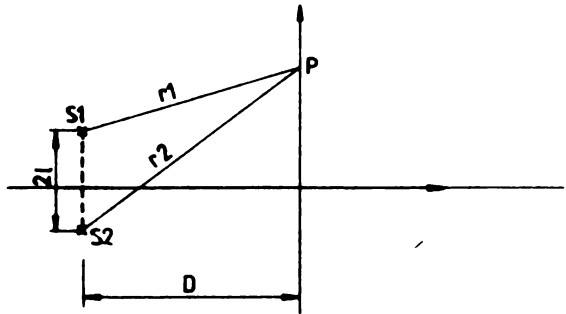


Fig. 4.21.

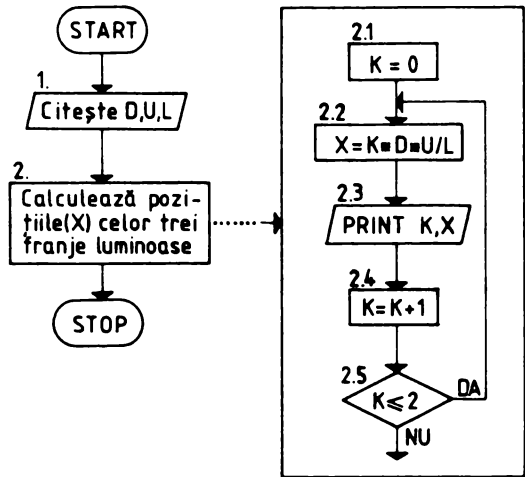


Fig. 4.22.

- h) În experiența lui Young se lucrează cu o radiație monocromatică cu $\lambda = 6 \cdot 10^{-7}$ m. Distanța dintre fante (v. fig. 4.21) este 1 mm, iar distanța de la fante la ecran 3 m. Să se găsească pozițiile primelor trei franje.
 Schema logică este prezentată în figura 4.22.

- i) Să se calculeze abaterea standard a rezultatelor obținute de un student la cursul de programarea calculatoarelor, știind că el a susținut 8 teste. Se vor realiza tabela de variabile și schema logică.

Indicație. Abaterea standard se calculează cu formula:

$$\epsilon = \sqrt{\frac{\Sigma x^2 - (\Sigma x)^2/n}{n-1}}, \text{ unde:}$$

n reprezintă numărul de note;

Σx este suma notelor;

Σx^2 este suma pătratelor notelor.

- Se reia problema din tema precedentă complicind-o sub următorul aspect. Programul citește de la terminal cantitățile lunare de jucării pentru 500 de articole fabricate de întreprinderea „URSULEȚUL”, calculează cantitățile totale trimestriale și procentele corespunzătoare din cantitatea totală anuală.
- Se reia problema din tema precedentă complicind-o după cum urmează. Programul citește de la terminal consumurile trimestriale pentru 1000 de articole, calculează și afișează procentele corespunzătoare consumurilor trimestriale din consumul total anual.

SOLUȚIA TEMEI 4

- Nu răspundem.
- Nu răspundem.
- Programele BASIC sint:

- b) Formula lui Balmer pentru determinarea lungimilor de undă a patru linii din vizibil ale atomului de hidrogen este:

$$\lambda = 3645,6 \frac{n^2}{n^2-4} \cdot 10^{-10} \text{ m}$$

unde: λ este lungimea de undă;

$n=3, 4, 5, 6$.

Schema logică a programului este ilustrată în figura 4.23.

```
10 FOR N=3 TO 6
20   U=3645.6*(N↑2/(N↑2-4))/10↑10
30   PRINT "N="; N; "U="; U
40 NEXT N
50 END
```

```
d) 10 S=0 : S1=1
20 INPUT "N="; N
30 FOR I=1 TO N
40   S=S+S1/I * I
50   S1=-S1
60 NEXT I
70 PRINT "S="; S
80 END
```

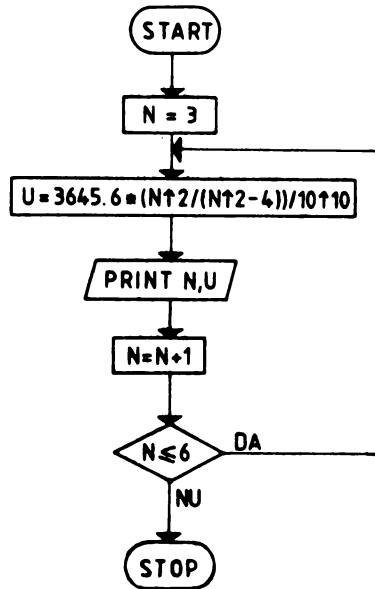


Fig. 4.23.

i) `10 B=0 : C=0` `60 NEXT I`
`20 FOR I=1 TO 8` `70 E=((C-C/8)/7) ↑ · 5`
`30 INPUT "NOTA="; N` `80 PRINT "E="; E`
`40 B=B+N` `90 END`
`50 C=C+N * N`

- Se vor adăuga la programul din tema precedentă (v. conversația 3, pag. 150) următoarele instrucțiuni:

```
1 FOR I=1 TO 500
```

(se copiază instrucțiunile din liniile 5–120).

```
125 NEXT I
130 END
```

- Se vor adăuga la programul din tema precedentă (v. conversația 3, pag. 153) următoarele instrucțiuni:

```
1 FOR I=1 TO 1000
```

(se copiază instrucțiunile din liniile 5–115).

```
115 NEXT I
120 END
```

Proiectarea și realizarea unui program BASIC pentru calculul cantității de lichid, dintr-un număr variabil de rezervoare ale căror raze pot lua numai valori pozitive. Variabile indexate numerice. Vectori de date. Instrucțiunea DIM. Introducerea statică a datelor. Instrucțiunile READ și DATA. Structura de iterație CÎT TIMP. Bucla WHILE-WEND. Structura de selecție. Instrucțiunile IF-THEN și IF-THEN-ELSE. Subprograme. Instrucțiunile GOSUB și RETURN. JOCURI, APLICAȚII și TESTE pentru cititor



□ De la problemă la program

Problema ce se pune seamănă cu precedentele, dar este mai complicată. Se citesc de la tastatură un număr oarecare (maximum 30) de valori ale razelor unor rezervoare cilindrice echilaterale. Pentru fiecare valoare a razei se calculează și afișează cantitatea de benzină din rezervor. Programul verifică, totodată, ca valorile razelor rezervoarelor să fie numere pozitive. În caz de eroare se afișează un mesaj corespunzător. De asemenea, programul calculează și afișează masa totală de benzină din rezervoare. În felul acesta, cu același program se pot rezolva probleme ce diferă prin numărul de rezervoare.

□ Analiza problemei

Problemele care se pun în această etapă privesc în special alcătuirea tabelii de variabile mai precis identificarea variabilelor de intrare ale programului.

Numărul rezervoarelor, necunoscut în problemă trebuie furnizat ca o variabilă de intrare (N) în momentul execuției programului, înaintea citirii datelor propriu-zise (razele rezervoarelor). Cît privește numărul variabil de raze, de asemenea necunoscut în momentul scrierii programului vom folosi ca variabilă de intrare un **vector** adică un tablou ce conține un șir de variabile, avînd o dimensiune maximă cunoscută (30 în acest caz).

Variabila structurată de tip vector

Un vector poate fi imaginat ca o clădire cu mai multe apartamente. Dacă, de exemplu, numele clădirii este ZAMBILA și ea dispune de 4 apartamente, atunci acele apartamente se vor identifica (într-o exprimare vectorială) prin ZAMBILA (1), ZAMBILA (2), ZAMBILA (3) ZAMBILA (4).

În informatică, se desemnează prin abuz de limbaj sub numele de vector, mulțimea $\{V(1), V(2), \dots, V(n)\}$, unde $V(1), V(2), \dots, V(n)$ sînt elementele vectorului. Așadar, un vector este o mulțime de elemente identificate prin poziția pe care o ocupă. Prelucrările ce se efectuează asupra elementelor unui vector sînt funcție de valoarea unui indice de poziție ($1 \leq i \leq n$). Acestuia i se fixează o valoare inițială care, în general, corespunde primei poziții, iar prelucrarea se efectuează cît timp valoarea acestui indice nu depășește o valoare finală.

Pentru EXEMPLELE 5 vom defini variabila structurată de tip vector R (variabila de intrare, reprezentînd raza rezervorului) căreia îi asociem o variabilă de stare I semnificînd indicele de poziție (indexul) al vectorului R cu valori între 1 și N.

Specificațiile de programare sînt prezentate în modulul de analiză structurată, figura 5.1.

FORMATUL DATELOR DE IEȘIRE

Nume program: EXEMPLELE 5 – PC, m, M, F

RAZA

XX
XX
:
:

MASA

XXXX · XX
XXXX · XX
:
:

MASA TOTALĂ = XXXXX · XX TONE

a)

TABELA DE VARIABILE

Nume program: EXEMPLELE 5 – PC, m, M, F

Variabile de intrare

Variabile de stare

Variabile de ieșire

R : vectorul
razelor
rezervorului

I : indicele
de poziție
(indexul)

b)

TABELA DE VARIABILE

Nume program: EXEMPLELE 5–PC, m, M, F

Variabile de intrare

Variabile de stare

Variabile de ieșire

N: număr de rezervoare

V: volumul rezervorului
S: masa totală de benzină
M: masa de benzină din
rezervor

M: masa de benzină din
rezervor
S: masa totală de benzină

c)

Fig. 5.1. a-d. Modulul de analiză structurată: a) formatul datelor de ieșire; b) tabela de variabile; c) tabela de variabile (completare);

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 5 – PC, m, M, F

Descrierea programului

Programul calculează și afișează cantitatea de benzină (S) dintr-un număr oarecare (N) de rezervoare cilindrice echilaterale (acest număr este furnizat ca parametru). Programul citește valorile razelor rezervoarelor într-un vector (R) printr-o procedură de introducere dinamică a datelor (razele cu valori negative nu se iau în considerare). Programul citește densitatea benzinei (D) printr-o procedură de introducere statică a datelor.

Intrări

Se introduc de la tastatură numărul rezervoarelor cilindrice echilaterale și valorile razelor acestora.

Ieșiri

Masa de benzină (M) din fiecare rezervor și masa totală (S) de benzină.

Lista de funcțiuni ale programului

1. Citește număr rezervoare (N)
2. Inițializează variabila S
3. Afișează un rând cu simbolul "="
4. Afișează "raza, masa"
5. Pentru fiecare rezervor:
 - 5.1. Citește valoarea razei unui rezervor

- 5.2. Validează datele introduse după cum urmează: pentru fiecare articol se verifică dacă valoarea razei citite este negativă. Programul afișează mesajul "Raza negativă".

6. Pentru fiecare rezervor:

- 6.1. Calculează volumul rezervorului
 - 6.2. Calculează masa de benzină din rezervor
 - 6.3. Insumează M în S
 - 6.4. Afișează R, M
 7. Afișează masa totală de benzină
 8. Stop
-

d)

d) specificații de programare.

Proiectarea programului

Diagrama de structură

Pentru trasarea diagramei de structură s-a procedat la identificarea, mai întâi, a principalelor funcțiuni de prelucrare apoi la alocarea acestor funcțiuni modulelor de program (v. fig. 5.2 (a)), după aceea trecându-se la reprezentarea diagramei de structură (v. fig. 5.2 (b)).

Vom dețuca din această diagramă blocul de iterație PREL-VALID ale cărui funcțiuni sînt:

- citește raza;
- afișează raza;
- validează datele introduse și afișează mesaj în caz de eroare (raza negativă).

Observație. Lista completă de funcțiuni este prezentată în modulul de analiză structurată, fig. 5.1 (d).

De notat că blocul PREL-VALID (nivelul 1) conține și el un bloc de iterație-VALID (nivelul 2) care se execută **cît timp** raza rezervorului este negativă (v. lista de funcțiuni ale modulului).

Spre deosebire de blocul PREL-VALID, blocul VALID reprezintă o structură de iterație condiționată anterior ce se execută cît timp condiția ("Raza este negativă") este adevărată.

Remarci

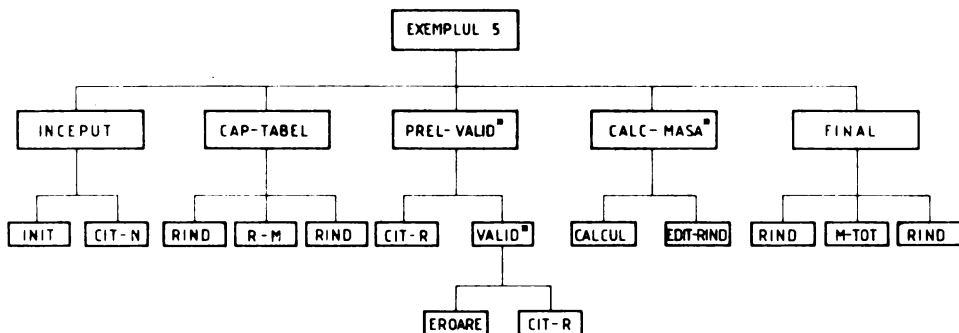
- Diagrama de structură utilizează două module comune: RIND (v. blocurile CAP-TABEL, FINAL) și CIT-R (v. blocurile PREL-VALID, VALID).
- Lista de funcțiuni a modului CALCUL implică prelucrări de vectori.
- Blocul PREL-VALID reprezintă o structură de iterație cu număr cunoscut de pași (N).
- Blocul VALID reprezintă o structură de iterație condiționată anterior.

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 5 – PC, m, M, F

Modul	Funcțiuni
INIT	2
CIT-N	1
RIND	3
R-M	4
PREL-VALID	5
VALID	5.2, 5.1
EROARE	5.2
CALCUL	6.1, 6.2
M-TOT	6.3
FINAL	3, 7, 8
CIT-R	5.1
EDIT-RIND	6.4

a)



b)

Fig. 5.2. a-d. Modulul de proiectare structurată: a) alocarea funcțiunilor de prelucrare; b) diagrama de structură;

PSEUDOCODUL

Nume program: EXEMPLELE 5 – PC, m, M, F

```

EXEMPLUL5      SEQ
INCEPUT        SEQ
                S=0
                INPUT N

INCEPUT        END
CAP-TABEL      SEQ
                DO RIND
                PRINT "RAZA", "MASA"
                DO RIND

CAP-TABEL      END
PREL-VALID     PENTRU I DE LA 1 LA N
                DO CIT-R
                PRINT R(I)

VALID          CIT TIMP (R(I)<0)

EROARE         SEQ
                PRINT "RAZA NEGATIVĂ"
                DO CIT-R
                PRINT R(I)

EROARE        END
VALID         SFIRȘIT

PREL-VALID    SFIRȘIT
CALC-MASA     PENTRU I DE LA 1 LA N
CALCUL        SEQ
                M=D * 2 * PI * R (I) † 3
                S=S+M
                PRINT R (I), M

CALCUL        END
CALC-MASA     SFIRȘIT
FINAL         SEQ
                DO RIND
                PRINT "MASA TOTALĂ="; S; "TONE"
                DO RIND

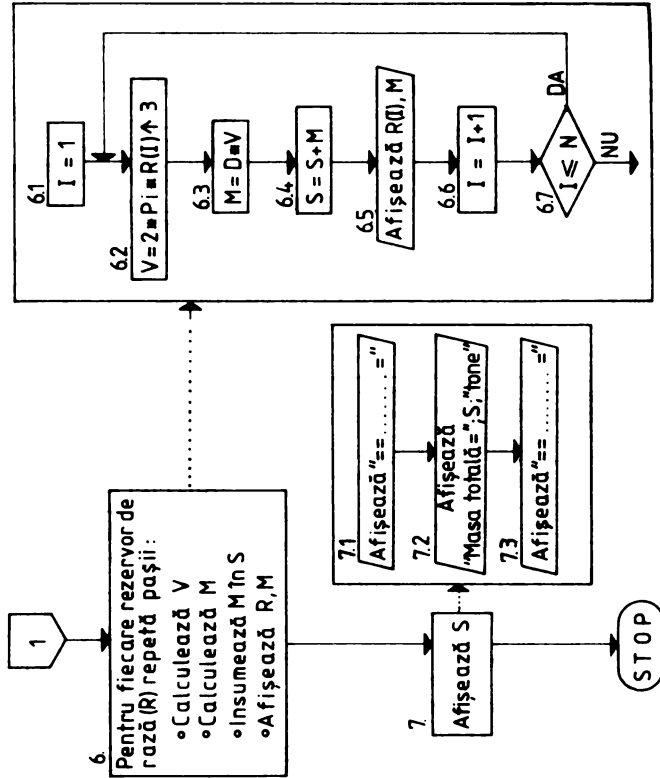
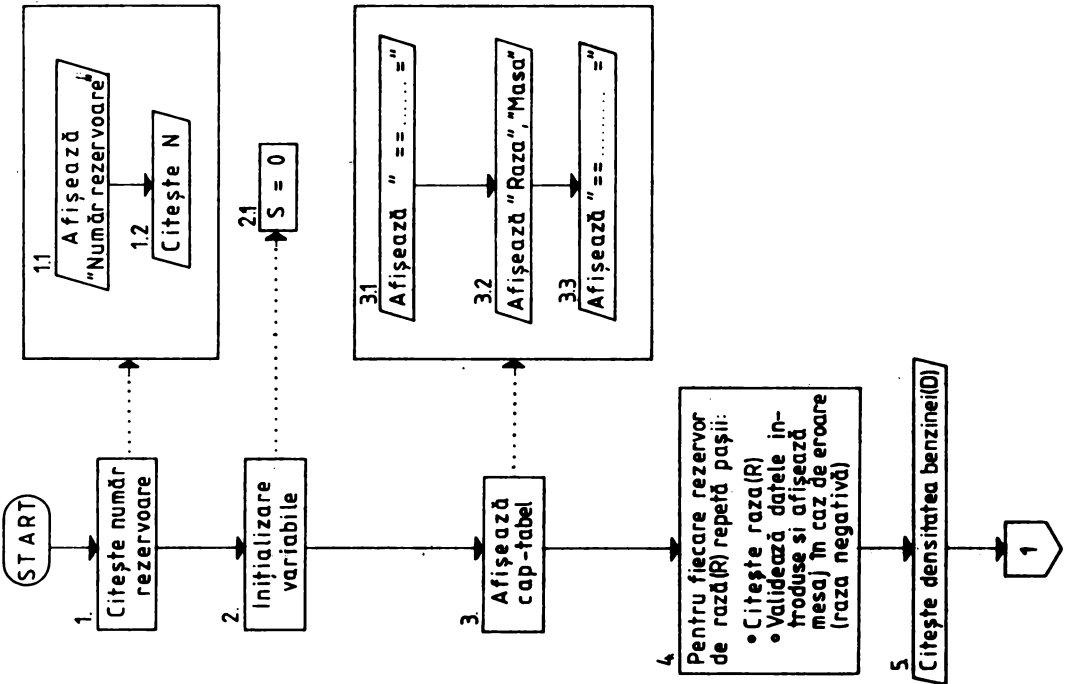
FINAL         END
EXEMPLUL5     END
CIT-R         SEQ
                INPUT R (I)

CIT-R         END

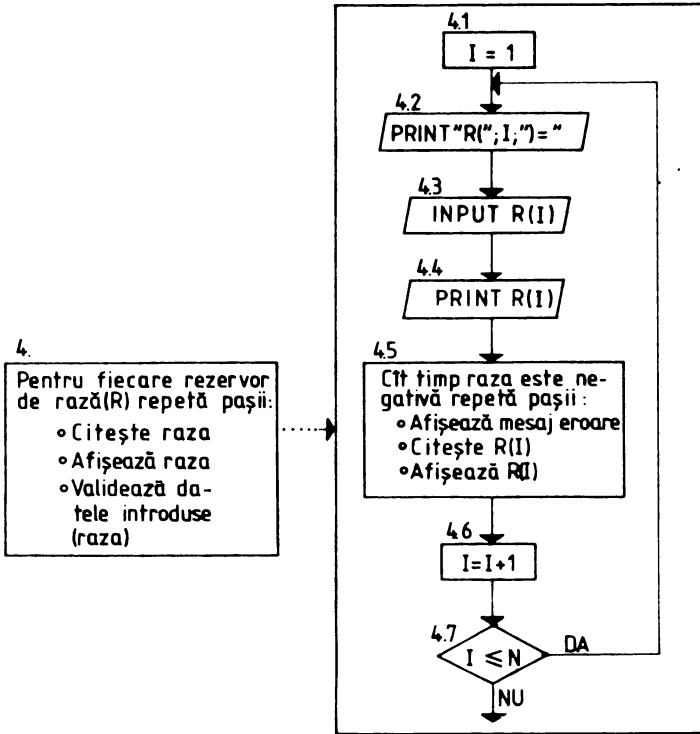
```

c)

c) pseudocodul;

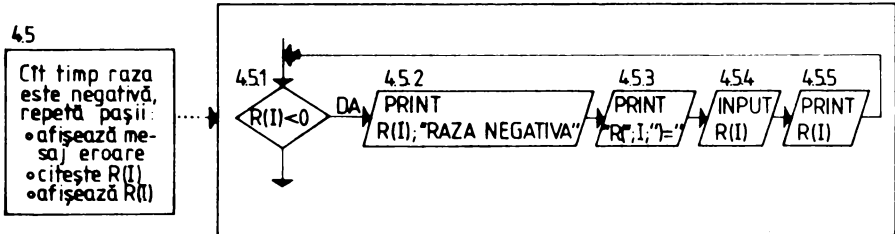


d) schema logică.



4. Pentru fiecare rezervor de rază(R) repetă pașii:

- Citește raza
- Afixează raza
- Validează datele introduse (raza)



d) schema logică.

Pseudocodul

De foarte multe ori, în rezolvarea problemelor cu calculatorul, o acțiune sau un grup de acțiuni necesită o execuție repetată. Pentru ca un calculator să poată executa astfel de operații este necesară o structură de iterație care să permită descrierea unei repetiții (iterații) într-o formă comodă. Există mai multe moduri de reprezentare a iterației. Cu una dintre ele ați făcut și dumneavoastră cunoștință (v. structura **PENTRU-SFIRȘIT**). Se pot întâlni însă numeroase cazuri de iterație diferite de cel precedent când o repetiție are loc atâta timp cât (sau pînă cînd) o anumită condiție este îndeplinită.

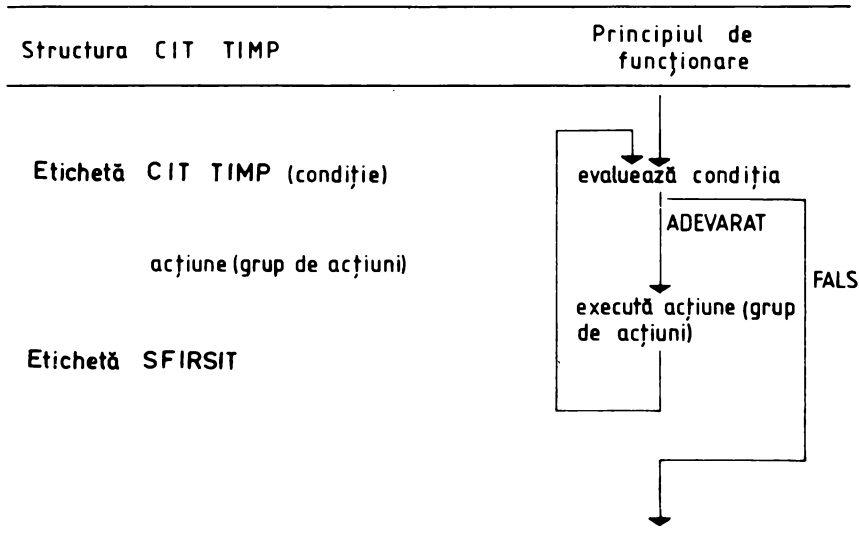
Structura de iterație CIT TIMP

Modulul VALID din EXEMPLUL 5 reprezintă o structură de iterație în care se repetă acțiunile:

- Afișează "Raza negativă"
- Citește o nouă rază
- Afișează raza

atâta timp cât $R(I) < 0$.

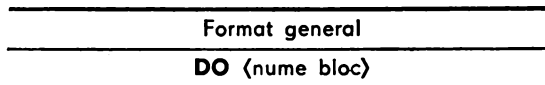
Spunem în acest caz că am identificat o structură de iterație condiționată anterior – **CIT TIMP** sau "bucula de tip **DOWHILE**" al cărei format general este:



Remarci

- Instrucțiunile **CIT TIMP** și **SFIRȘIT** au obligatoriu aceeași etichetă.
- Instrucțiunea **CIT TIMP** reprezintă punctul unic de intrare în bloc.
- Instrucțiunea **SFIRȘIT** reprezintă punctul unic de ieșire din bloc.
- Blocurile cuprinse între **CIT TIMP** și **SFIRȘIT** sunt executate atâta timp cât condiția de ieșire din bloc este adevărată.

În figura 5.2 (c) se prezintă scrierea în limbaj pseudocod a programului. De notat că pentru scrierea modulelor comune de prelucrare (**RIND** și **CIT-R**) s-a utilizat blocul de prelucrare **DO (EXECUTA)** al cărui format general este:



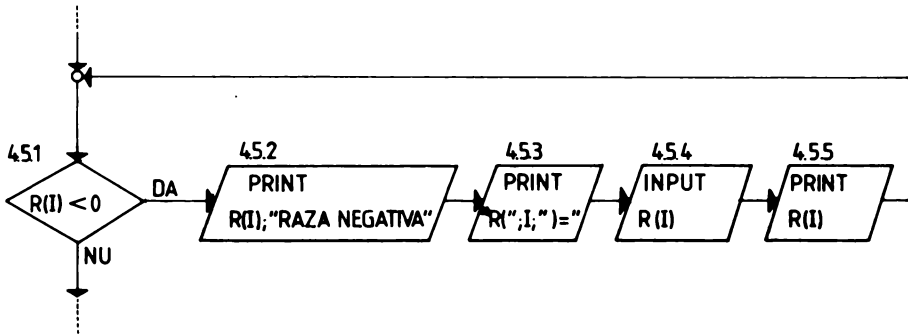
Remarcați, de asemenea, în secvența algoritmică intitulată **CALCUL**, modul în care s-a determinat masa de benzină a unui rezervor:

$$M = D * 2 * \pi * R(I) \uparrow 3$$

în funcție de densitatea benzinei și de raza $R(I)$, pentru I avînd valori de la 1 (primul rezervor) pînă la N (al N -lea rezervor).

Schema logică

În modulul de proiectare structurată, fig. 5.2 (d) este ilustrată schema logică a programului. Vom extrage din această schemă numai blocurile care reprezintă structura de iterație condiționată anterior – **CIT Timp**.



După cum se constată, structura de iterație începe cu un bloc de decizie care controlează corpul buclei. Condiția de repetiție ($R(I) < 0$) este evaluată înaintea execuției corpului buclei (structura de iterație condiționată anterior). Atîta timp cit condiția este adevărată se repetă grupul de acțiuni care constituie corpul ciclului. De îndată ce condiția devine falsă, execuția buclei se termină.

Observație. Dacă la prima evaluare condiția nu este adevărată, atunci corpul buclei nu va fi executat.

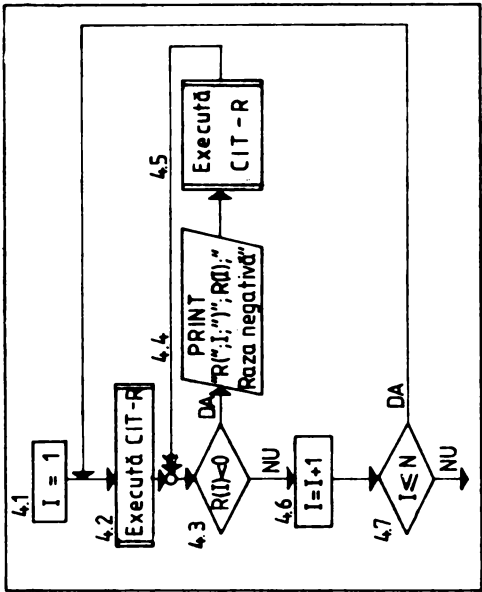
În figurile 5.3 și 5.4 "prin utilizarea simbolului $\boxed{\text{---}}$ " sînt ilustrate modulele independente de program RIND și CIT-R. Remarcați că cele două module au fost detaliate separat, avînd menționat în locul lui START și STOP numele propriu-zis (RIND, CIT-R) respectiv RETURN (întoarcere la blocul imediat următor blocului apelant).

□ Codificarea în limbajul BASIC-aMIC

vol. 2, pag. 210

Variabile indexate numerice

Noțiunea următoare pe care o vom introduce vă cere multă atenție. Pînă acum am folosit numai variabile simple. O variabilă simplă este formată dintr-o literă (de la A la Z) urmată opțional de o cifră (de la 0 la 9). Acum vă vom prezenta un nou tip de variabilă numit variabila indexată.



4. Pentru fiecare rezervor de rază (R) repetă pașii:

- Citește raza (R)
- Afișează raza (R)
- Validează datele introduse și afișează mesaj de eroare (rază negativă)

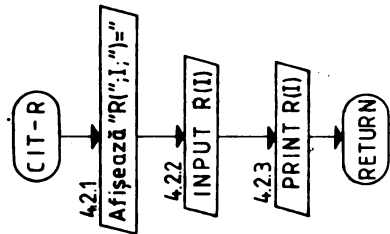


Fig. 5.3.

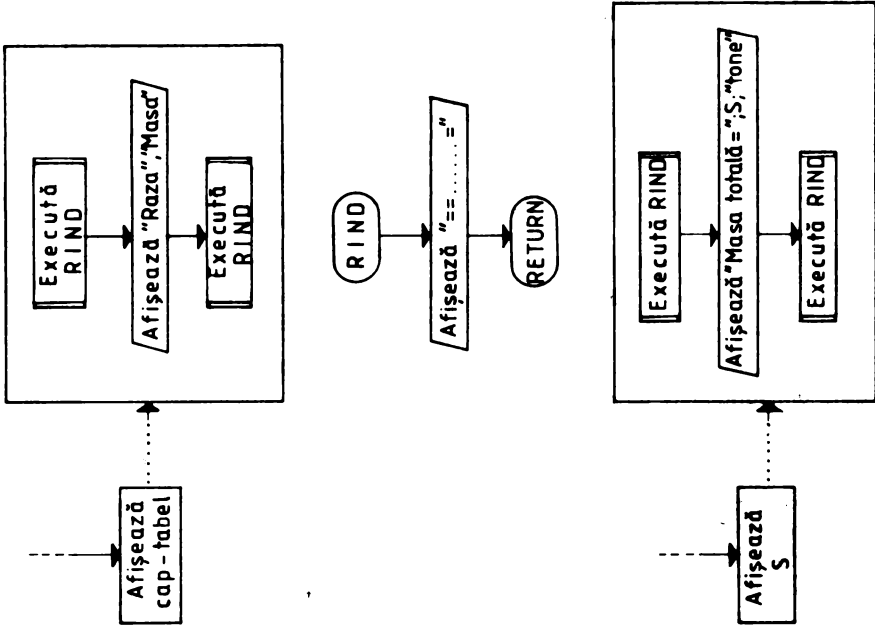
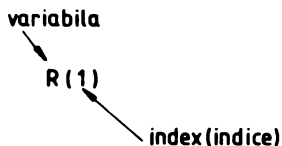


Fig. 5.4.

În BASIC-aMIC numele unei variabile indexate se compune dintr-o singură literă, $R(1)$ este o variabilă indexată pe cînd $R1$ nu este o variabilă indexată.



NOTA : Se citește "R de 1"

TEST

Care din variabilele de mai jos sînt indexate?

A (2); A; A2; B (30); M.

R. A (2) și B (30)

De notat că A, A2 și A(2) reprezintă trei variabile diferite. Dumneavoastră le puteți confunda, dar calculatorul le va recunoaște ca variabile distincte.

R(1)	
R(2)	8
R(3)	
R(4)	
R(5)	3
R(6)	
R(7)	11
R(8)	
R(9)	

Observație. Într-un program BASIC-aMIC pot exista variabile simple numerice și variabile indexate cu același nume.

Ca și variabilele simple folosite pînă acum, variabila indexată ocupă, de asemenea, o locație în memoria calculatorului. Un set de variabile indexate cu o singură dimensiune reprezintă un vector.

Să presupunem acum că dumneavoastră sintetizați... calculatorul și $R(5)=3$.

Cu alte cuvinte, luați, vă rugăm creionul și puneți numărul 5 în cutia alăturată lui $R(5)$. Realizați același lucru și cu $R(2)=8$. Care va fi valoarea lui $R(7)$ dacă

$$R(7)=R(2)+R(5)$$

Verificați-vă răspunsul cu rezultatul din figură.

Ce este așa de minunat și de misterios în legătură cu aceste variabile? Iată explicația: variabilelor indexate li se pot atașa anumite valori. De exemplu, variabila $R(1)$ care are drept index tot o variabilă.

Dacă $l=1$ atunci $R(l)$ este $R(1)$;

Dacă $l=2$ atunci $R(l)$ este $R(2)$;

Dacă $l=4$ atunci $R(l)$ este $R(4)$

Indicele poate fi, de asemenea, o expresie, de exemplu $R(l+8)$.

Dacă $l=5$ atunci $R(l+8)$ este $R(13)$.

Remarcă. Dacă, în urma evaluării expresiei indice, nu se obține o valoare întregă, se va reține partea întreagă a valorii rezultate.

TESTE

a) Precizați valorile lui $A(1)$, $A(2)$, $A(3)$ în urma execuției următorului program:

```
5 DIM A(3)
10 FOR I=1 TO 3
20   A (I) =2 * I-1
30 NEXT I
```

b) Pentru următoarea buclă **FOR-NEXT** precizați valorile lui $I(1)$, $I(2)$, $I(3)$, $I(4)$.

```
5 DIM I(4)
10 FOR I=1 TO 4
20   I(I)=I * I
30 NEXT I
```

Introducerea dinamică a vectorilor de date

Cum pot fi folosite cel mai ușor variabilele indexate în limbajul BASIC?

Un prim mod de a le utiliza este cel al creării unui vector de date cu ajutorul instrucțiunii **INPUT** plasate într-o buclă **FOR-NEXT**. Pentru a ilustra acest caz, vă invităm să urmăriți instrucțiunile 70, 80, 90, 100 și 170 (numai pe acestea!) din program:

```
70 FOR I=1 TO N
80   PRINT "R (" ; I ; ") = " ;
90   INPUT R (I)
100  PRINT R (I)
    .
    .
    .
170 NEXT I
```

De notat că în timpul execuției acestei secvențe de program se vor citi N valori (corespunzătoare razelor celor N rezervoare cilindrice echilaterale) ce se vor atribui variabilelor $R(1)$, $R(2)$, ..., $R(N)$.

Totul pare normal, dar cu toate acestea ceva este în neregulă. Nu putem executa secvența de instrucțiuni BASIC (70–140) fără a-i preciza calculatorului valoarea lui N (numărul de rezervoare) dar și spațiul de memorie pe care trebuie să-l rezerve pentru datele ce aparțin vectorului $R(I)$. Vom realiza acest lucru adăugând programului secvența de instrucțiuni:

```
20 PRINT "NUMĂR REZERVOARE";
30 INPUT N
40 PRINT N
60 DIM R (30)
```

DIM: instrucțiune de dimensionare a tablourilor

DIM face o rezervare de memorie pentru date, dar nu determină nici un fel de operații (aritmetice sau de alt fel). Instrucțiunea **DIM** este foarte des utilizată în limbajul BASIC și totuși foarte mulți o evită întrucât nu înțeleg exact care este rolul său.

În cazul nostru în loc să se folosească N variabile pentru valorile razelor rezervoarelor cilindrice echilaterale s-a folosit un vector cu N elemente recunoscut sub numele R : prima valoare se află în primul element, a doua în cel de-al doilea element ș.a.m.d.

Într-o instrucțiune **DIM** se specifică între paranteze numărul de elemente din tablou. În aceeași instrucțiune se pot declara mai multe tablouri, separate prin virgulă. De exemplu, instrucțiunea

10 **DIM** A(30), A1(20), A2(50)

descrie trei vectori: A cu 30 de elemente, $A1$ cu 20 elemente, și $A2$ cu 50 elemente.

Formatul general al instrucțiunii este:

Format general
$\langle \text{nr. linie} \rangle$ DIM $\langle \text{variabilă} \rangle$ ($\langle \text{dim1} \rangle$ [, $\langle \text{dim2} \rangle$]) {, $\langle \text{variabilă} \rangle$ ($\langle \text{dim1} \rangle$ [, $\langle \text{dim2} \rangle$]) }

în care:

$\langle \text{variabilă} \rangle$ indică numele tabloului. El poate fi orice nume de variabilă indexată admisă de limbaj;

$\langle \text{dim1} \rangle$, $\langle \text{dim2} \rangle$ reprezintă dimensiunile tabloului. Ele pot fi constante, variabile numerice simple (de tip întreg) sau expresii aritmetice.

Remarci

- Dimensiunea unui tablou are valori cuprinse în intervalul [1, 254].
- Tablourile numerice pot avea una sau două dimensiuni.
- Indicele unei variabile indexate trebuie să aibă valori cuprinse în intervalul [1, $\langle \text{dimensiune declarată în DIM} \rangle$].
- La execuția instrucțiunii **DIM** pentru tablourile numerice, se alocă câte 4 octeți pentru fiecare element al unei variabile tablou.
- În cazul utilizării unor tablouri mai există situații cind la execuția programului se afișează mesajul de eroare MEMORY FULL indicind faptul că nu mai există memorie suficientă pentru alocarea tablourilor.

Introducerea statică a vectorilor de date

Pînă acum am introdus datele (în memoria calculatorului) în mod dinamic (v. instrucțiunea **INPUT**). Mai există o posibilitate pe care ne-o oferă limbajul BASIC – introducerea datelor în mod static. Pentru aceasta se înlocuiește **INPUT** cu instrucțiunile **READ** și **DATA**. **READ** citește datele conținute de o linie **DATA**.

Formatul instrucțiunilor este:

Format general
$\langle \text{nr. linie} \rangle$ READ (lista variabile)
Format general
$\langle \text{nr. linie} \rangle$ DATA (listă constante)

Ce se "pune" în instrucțiunea **DATA**? Tot ceea ce doriți: cifre, litere sau litere și cifre. Pentru citirea și utilizarea acestor date (listă constante) de către calculator, ele vor fi afectate unei liste de variabile din instrucțiunea **READ**.

În figura 5.5 se prezintă mecanismul de introducere statică a datelor.

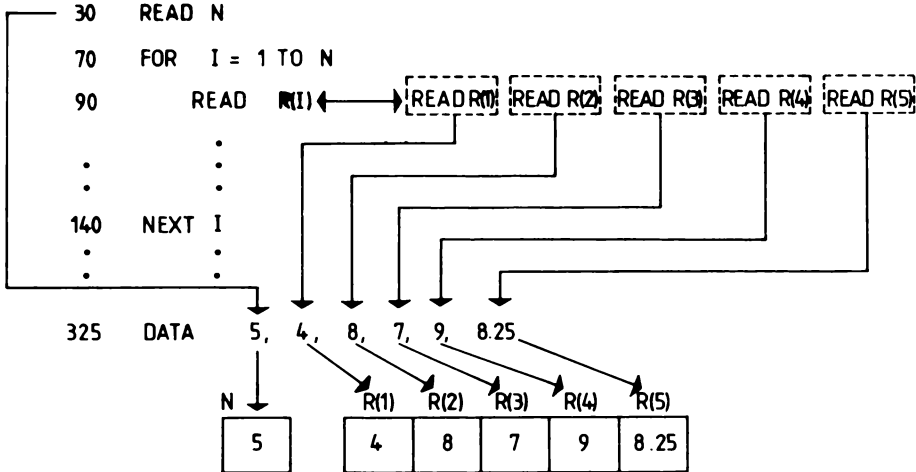


Fig. 5.5. Modul de utilizare a instrucțiunilor **READ** și **DATA**.

Remarci

- Instrucțiunea **READ** are același efect cu **INPUT**, cu deosebirea că datele nu sînt introduse de la tastatură, ci sînt citite dintr-un bloc de date, definit cu instrucțiunea **DATA**.
- O instrucțiune **DATA** nu este asociată unei instrucțiuni **READ**, ci toate instrucțiunile **DATA** sînt tratate ca și cum ar forma un bloc de date.
- Instrucțiunea **DATA** poate apare oriunde în cadrul programului.
- Pentru ținerea evidenței constantelor citite în instrucțiunile **DATA**, interpretorul **BASIC-aMIC** folosește un indicator.

În cadrul programului **BASIC-aMIC** instrucțiunile **READ-DATA** au fost întrebuițate pentru citirea valorii densității benzinei în mod static.

```

190 READ D
:
:
:
350 DATA .7
    
```

În urma execuției instrucțiunii din linia 190 lui **D** i se atribuie valoarea 0.7.

TEST

Precizați rezultatul execuției următorului program:

```

3 DIM A(3)
5 READ N
10 FOR I=1 TO N
20 READ A(I)
30 PRINT A(I)
40 NEXT I
50 DATA 3, 7, 8, 9
60 END
    
```

Aplicație. Se consideră sistemul de puncte materiale din figura 5.6, avind masele $m_1=m_5=m_{12}=0,004$ kg; $m_3=m_8=m_{10}=0,006$ kg; $m_2=m_6=m_{11}=0,008$ kg; $m_4=m_7=m_9=0,002$ kg. Distanțele dintre punctele materiale sînt exprimate în metri. Să se determine coordonatele centrului maselor.

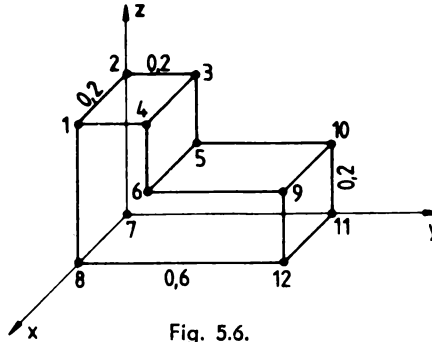


Fig. 5.6.

Pentru a determina coordonatele centrului maselor utilizăm relațiile:

$$X_c = \frac{\sum_{i=1}^{12} x_i m_i}{\sum_{i=1}^{12} m_i} \quad (1);$$

$$Y_c = \frac{\sum_{i=1}^{12} y_i m_i}{\sum_{i=1}^{12} m_i} \quad (2);$$

$$Z_c = \frac{\sum_{i=1}^{12} z_i m_i}{\sum_{i=1}^{12} m_i} \quad (3).$$

și completăm tabelul de mai jos:

Nr. crt.	m_i	x_i	y_i	z_i
1	0,004	0,2	0	0,4
2	0,008	0	0	0,4
3	0,006	0	0,2	0,4
4	0,002	0,2	0,2	0,4
5	0,004	0	0,2	0,2
6	0,008	0,2	0,2	0,2
7	0,002	0	0	0
8	0,006	0,2	0	0
9	0,002	0,2	0,6	0,2
10	0,006	0	0,6	0,2
11	0,008	0	0,6	0
12	0,004	0,2	0,6	0

```
10 REM CENTRUL MASELOR
20 DIM M(12), X(12), Y(12), Z(12)
30 FOR I=1 TO 12
```

```
40 READ M(I)
50 NEXT I
```

```

60 DATA 0.004, 0.008, 0.006, 0.002
   0.004, 0.008, 0.002, 0.006,
   0.002, 0.006, 0.008, 0.004
70 FOR I=1 TO 12
80 READ X(I)
90 NEXT I
100 DATA 0.2, 0, 0, 0.2, 0, 0.2
110 DATA 0, 0.2, 0.2, 0, 0, 0.2
120 FOR I=1 TO 12
130 READ Y(I)
140 NEXT I
150 FOR I=1 TO 12
160 READ Z(I)
170 NEXT I
180 DATA 0.4, 0.4, 0.4, 0.4, 0.2, 0.2,
   0, 0
190 DATA 0.2, 0.2, 0, 0
200 S=0
210 FOR I=1 TO 12
220 S=S+M (I)
145 DATA 0, 0, 0.2, 0.2, 0.2, 0.2, 0, 0,
   0.6, 0.6, 0.6, 0.6
230 NEXT I
240 P1=0
250 FOR I=1 TO 12
260 P1=P1+X (I) * M (I)
270 NEXT I
280 P2=0
290 FOR I=1 TO 12
300 P2=P2+Y(I)*M(I)
310 NEXT I
320 P3=0
330 FOR I=1 TO 12
340 P3=P3+Z(I)*M(I)
350 NEXT I
360 X1=P1/S
370 Y1=P2/S
380 Z1=P3/S
390 PRINT "XC="; X1; "M"
400 PRINT "YC="; Y1; "M"
410 PRINT "ZC="; Z1; "M"
420 END

```

Tehnici de implementare a structurii de iterație CIT TIMP • Metodologie

Odată identificate structurile fundamentale de program (secvența, selecția, iterația) se pune problema implementării lor, în cazul nostru în limbajul BASIC. Iată posibilitățile principale:

a) utilizarea unor interpretoare (compilatoare) speciale de BASIC structurat; b) utilizarea precompilatoarelor, în care pornindu-se de la un limbaj auxiliar de descriere a structurilor (pseudocod) se generează instrucțiuni BASIC standard; c) simularea structurii în BASIC standard cu respectarea unei anumite metodologii.

În acest capitol vom trata implementarea structurii fundamentale de program **CIT TIMP**, pe calculatorul aMIC, utilizând limbajul "BASIC structurat" care poate fi prelucrat de un interpretor corespunzător: BASIC-AMSTRAD, BASIC-80, ABASIC, GWBASIC.

Pe calculatoarele personale aMIC, PRAE, HC-85, COMMODORE cit și pe minicalculatoarele INDEPENDENT, CORAL nu există un astfel de interpretor, implementarea structurii descrise putându-se efectua printr-un preprocesor sau prin "traducere cu mâna".

Codificarea în BASIC structurat a buclei de tip **CIT TIMP** este:

```

WHILE (condiție)
    <corpul buclei>
WEND

```

Remarci

- **WHILE** indică punctul de intrare în ciclu.
- **WEND** indică punctul de ieșire din ciclu.
- <condiție> definește condiția care controlează corpul buclei.
- pot exista oricâte nivele de cicluri **WHILE-WEND**.
- fiecare **WEND** închide cel mai apropiat **WHILE**.

Simularea iterației

BASIC Structurat → traducere → **BASIC standard**

WHILE (condiție) e1 **IF** (complementul condiției) **THEN** e2

(corpul buclei)

(corpul buclei)

WEND

GOTO e1
e2 (instrucțiunea următoare)

Remarci

- Semnificația instrucțiunii **IF-THEN** din linia e1 este următoarea: **DACA (IF)** condiția (complementul condiției) este adevărată **ATUNCI (THEN)** calculatorul merge ("sare") la linia e2 după care continuă execuția programului. Prima parte a instrucțiunii (**IF**) pune condiția (complementul condiției) iar cealaltă parte a instrucțiunii (**THEN**) spune calculatorului la ce linie e2 să se ducă (ce linie să urmeze) dacă condiția (complementul condiției) este adevărată. De exemplu, instrucțiunea

```
110 IF R(I) > 0 THEN 170
120
```

spune calculatorului să meargă la linia 170 dacă valoarea lui R (I) este mai mare ca zero.

Dacă valoarea lui R (I) este mai mică sau egală cu zero, calculatorul nu mai merge la linia 170. El pur și simplu trece la linia următoare (120), continuând în acest fel execuția programului.

- Instrucțiunea **GOTO** spune calculatorului să facă un "salt" înainte sau înapoi, în cadrul programului la linia a cărei etichetă e1 este indicată după **GOTO**.

De exemplu, instrucțiunea **GOTO** din linia 160

```
110 IF R(I) > 0 THEN 170
160 GOTO 110
```

spune calculatorului să se ducă de la linia 160 la linia 110 (salt necondiționat) și apoi să continue cu instrucțiunea din linia 110, 120 etc.

Metodologie pentru simularea iterației CIT TIMP

Putem formula o metodologie de traducere (din **BASIC** structurat în **BASIC-aMIC Standard**) respectând următorii pași:

1. Alegem două numere de linie e1 și e2 nefolosite în program pentru instrucțiunile **GOTO** și **IF**.

2. Folosind complementul condiției \bar{C} scriem instrucțiunea **BASIC**
e1 **IF** \bar{C} **THEN** e2

3. Listăm instrucțiunile ce urmează a fi incluse în corpul buclei

4. Scriem instrucțiunile

GOTO e1

e2 (instrucțiunea următoare),
urmând imediat corpul buclei.

Înapoi la program

Urmăriți pentru început, în tabelul 5.1 simularea iterației **CIT TIMP** în limbajul **BASIC-aMIC**. Remarcați corespondența dintre **WHILE-IF THEN** și **WEND-GOTO**.

Tabelul 5.1

BASIC structurat	BASIC-aMIC (Standard)
110 WHILE R(I)<0	110 IF R(I)>0 THEN 170
120 PRINT "R("; I; ")="; R(I); "RAZA NEGATIVA"	120 PRINT "R("; I; ")="; R(I); "RAZA NEGATIVA"
130 PRINT "R("; I; ")=";	130 PRINT "R("; I; ")=";
140 INPUT R(I)	140 INPUT R(I)
150 PRINT R(I)	150 PRINT R(I)
160 WEND	160 GO TO 110
170	170

În figura 5.7 puteți urmări, de asemenea, întregul mecanism de funcționare al instrucțiunilor **FOR**, **IF-THEN** și **GOTO** aplicat programului BASIC-aMIC.

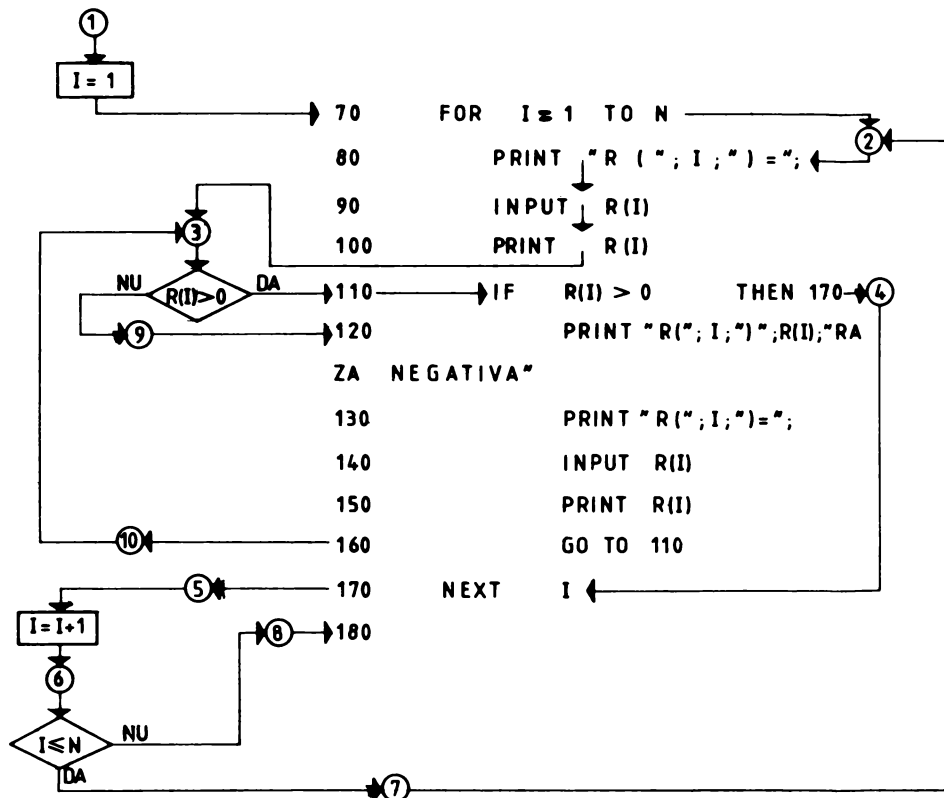


Fig. 5.7. Mecanismul de funcționare a instrucțiunilor **FOR**, **IF-THEN** și **GOTO**.

- 1) Se inițializează variabila de control (I) a buclei **FOR**;
- 2) Se execută instrucțiunile din liniile 80, 90, 100;
- 3) Se testează condiția $R(I) > 0$;
- 4) Dacă condiția este adevărată se execută saltul la linia 170;
- 5) Se incrementează valoarea lui I;
- 6) Se testează condiția $I \leq N$;
- 7) Dacă condiția este adevărată se reia ciclul de la pasul 2;
- 8) Dacă condiția $I \leq N$ nu este adevărată, se execută instrucțiunea din linia 180;
- 9) Dacă condiția $R(I)$ nu este adevărată, se execută instrucțiunea imediat următoare după **IF** (linia 120) urmată de instrucțiunile din liniile 130, 140 și 150;
- 10) Se execută saltul necondiționat la instrucțiunea din linia 110.

Structuri alternative. Instrucțiunea IF-THEN

În mod normal, pașii unui algoritm sînt realizați în ordinea specificării lor. Totuși succesiunea acestora este determinată de structura datelor de intrare. În funcție de valorile acestora se iau anumite decizii. Rezultatul evaluării determină ce pași ai algoritmului urmează să se execute. Începînd cu Conversația 5, ne întîlnim cu astfel de situații pe tot parcursul lucrării. În schema logică din figura 5.7 este pus în evidență blocul de decizie cu o alternativă (selecția simplă).

Se citește: **DACA** R(I) este mai mare ca zero **ATUNCI** mergi la linia 170.

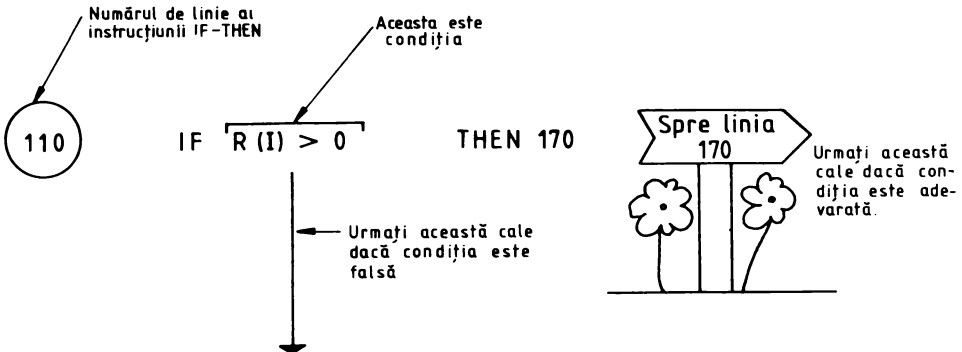
În BASIC-aMIC, o astfel de situație se scrie:

110 IF R(I) > 0 THEN 170

Instrucțiunea **IF-THEN** este o instrucțiune de salt condiționat. Formatul general al instrucțiunii este:

Format general
(nr. linie) IF (relație) THEN (nr. linie. 1)

IF-THEN indică două continuări posibile, în funcție de rezultatul evaluării condiției (relație) (o expresie relațională). Dacă rezultatul evaluării expresiei relaționale este de tipul "condiție îndeplinită" atunci se execută instrucțiunea cu numărul de linie (nr. linie 1), iar dacă este de tipul (condiție neîndeplinită) execuția va continua cu instrucțiunea care urmează după **IF**.



Condiția ce apare între cuvintele cheie **IF** și **THEN** este de forma unei relații între două expresii numerice sau șiruri. Relațiile se definesc cu ajutorul următorilor operatori relaționali:

Simbol matematic	Simbol BASIC-aMIC	Semnificație	Tipuri operanzi
$=$	<code>=</code>	egal	Constante și variabile numerice sau șir
$<$	<code><</code>	mai mic	
$<=$	<code><=</code>	mai mic sau egal	
$>$	<code>></code>	mai mare	
$>=$	<code>>=</code>	mai mare sau egal	
\neq	<code><></code>	diferit	

Aplicație. Identificați în BASIC-aMIC următoarele situații:

- a) Dacă valoarea lui A este mai mică decât cinci atunci mergi la linia 80.
- b) Dacă valoarea lui B este mai mare sau egală cu cea a lui C la pătrat atunci mergi la linia 30.
- c) Dacă de 8 ori valoarea lui X nu este egală cu 7 atunci mergi la linia 800.

-
- a) 10 IF A < 5 THEN 80
 - b) 20 IF B >= C ↑ 2 THEN 30
 - c) 30 IF 8 * X < > 7 THEN 800

TEST

Se consideră următorul program BASIC:

```
10 FOR I=1 TO 9
20 READ A
30 IF A < 0 THEN 50
40 PRINT "A="; A
```

- a) Pentru care din valorile conținute de blocul de date DATA condiția $A < 0$ este adevărată?
- b) Când $A < 0$ la ce linie se va face saltul?
- c) Dacă condiția $A < 0$ este adevărată, valoarea lui A va mai fi afișată?
- d) Pentru care din valorile conținute de blocul de date DATA condiția $A < 0$ este falsă?
- e) Când condiția $A < 0$ este falsă la ce linie se va face saltul?

```
50 NEXT I
60 DATA 4, 8, 0, -3, 6, -2, 7, 9, -4
70 END
```

- f) Dacă condiția $A < 0$ este falsă, va mai fi afișată valoarea lui A?
- g) Modificați instrucțiunea IF-THEN astfel încât programul să afișeze numai numerele diferite de zero.
- h) Modificați instrucțiunea IF-THEN astfel încât programul să afișeze numai numerele mai mari sau egale cu 2.
- i) Modificați instrucțiunea IF-THEN astfel încât programul să afișeze numai numerele: 7; 8; 9.

Instrucțiunea GOTO

În cadrul programului am utilizat instrucțiunea GOTO pentru construcția unei bucle de tip CIT TIMP.

```
160 GO TO 110
```

Instrucțiunea GOTO definește un salt necondiționat la orice instrucțiune din cadrul unui program.

Formatul instrucțiunii este:

Format general

⟨nr. linie⟩ GOTO ⟨nr. linie-n⟩

unde ⟨nr. linie-n⟩ reprezintă numărul de linie al instrucțiunii la care se face saltul (transferul).

Observație. Limbajul BASIC-aMIC, nu este un limbaj structurat. Deși dăunătoare, instrucțiunea GO TO servește simulării structurilor algoritmice fundamentale.

TEST

Se consideră programul BASIC-aMIC

```
10 A=0
20 PRINT "A="; A
30 A=A+5
40 IF A > 25 THEN 60
```

```
50 GOTO 20
60 PRINT "A > 25"
70 END
```

a) Precizați liniile care definesc o buclă în cadrul programului.

b) De câte ori instrucțiunea **IF-THEN** a verificat valoarea lui A înainte de a ști dacă condiția pusă a fost adevărată?

R. a) 20, 30, 40, 50; b) de șase ori.

Aplicație. Se consideră următorul program BASIC:

```
10 F=1                                40 F=F+1
20 IF F>8 THEN 99                    50 GO TO 20
30 PRINT "F="; F                      99 END
```

Modificați programul utilizând o structură de iterație cu număr cunoscut de pași

```
10 FOR F=1 TO 8                        30 NEXT F
20 PRINT "F="; F                       40 END
```

□ Codificarea în limbajul BASIC-PRAE

2 noi 110

Structura de selecție

În procesul de alcătuire a algoritmilor intră în sarcina noastră să exprimăm condițiile care urmează să fie evaluate de calculator în timpul execuției programului. Vom relua în cadrul acestui capitol problema structurilor de selecție.

Structura de selecție definește posibilitatea de selectare a unor structuri fundamentale algoritmice (de program), fiecare dintre ele avînd un singur punct de intrare și un singur punct de ieșire. Structura de selecție poate fi cu una (selecția simplă) sau cu două alternative (selecția).

În literatura de specialitate sint precizate mai multe forme de reprezentare a structurilor de selecție.

Pentru selecția simplă (structura de decizie cu o singură alternativă) în cadrul acestei lucrări vom folosi reprezentarea

ETICHETĂ **DACA** <condiție>

 <G_{A_D} – Grup de acțiuni pe ramura de DA>

ETICHETĂ **SFIRȘIT**

Pentru selecție (structura de decizie cu două alternative) vom utiliza reprezentarea:

ETICHETĂ **DACA** <condiție>

 <G_{A_D} – grup de acțiuni pe ramura de DA>

ETICHETĂ **IN CAZ CONTRAR**

 <G_{A_N} – grup de acțiuni pe ramura de NU>

ETICHETĂ **SFIRȘIT**

Observații

- Instrucțiunile **DACA**, **IN CAZ CONTRAR** și **SFIRȘIT** au obligatoriu aceeași etichetă.
- Instrucțiunea **DACA** reprezintă punctul unic de intrare în bloc.
- Instrucțiunea **SFIRȘIT** reprezintă punctul unic de ieșire din bloc.
- Dacă condiția este adevărată se execută acțiunile cuprinse între **DACA** și **IN CAZ CONTRAR** după care **IN CAZ CONTRAR** acționează ca o instrucțiune **GOTO SFIRȘIT**.
- Dacă condiția este falsă se execută acțiunile cuprinse între **IN CAZ CONTRAR** și **SFIRȘIT**.

În figura 5.8 sînt prezentate sintetic formatele structurii alternative cu ajutorul schemelor logice, diagramei de structură și pseudocodului.

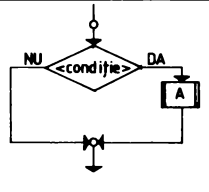
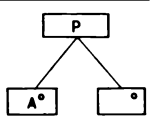
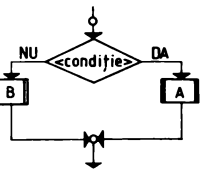
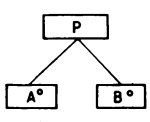
Tip instrucțiune	Schema logică	Diagrama de structură	Pseudocod
DACA-ATUNCI		 Observație: Se citește: P constă din sau A. Cerculețul semnifică "sau"(selecția)	Etichetă DACA condiție A Etichetă SFIRȘIT
DACA-ATUNCI- IN CAZ CONTRAR		 Observație: Se citește: P constă din sau A sau B.	Etichetă DACA condiție A Etichetă IN CAZ CONTRAR B Etichetă SFIRȘIT

Fig. 5.8. Reprezentarea structurilor alternative utilizind schemele logice, diagrama de structură și pseudocodul.

IF-THEN-ELSE

În BASIC-PRAE structurile de selecție (selecția și selecția simplă) se codifică cu instrucțiunea **IF-THEN-ELSE** (v. restricțiile instrucțiunii).

Formatul general este:

Format general
(nr. linie) IF (relație) THEN (nr. linie 1/instrucțiune 1) [ELSE (nr. linie 2/instrucțiune 2)]

în care:

⟨relație⟩ este de forma unei relații între două expresii (numerice sau șiruri) sau a unei reuniuni de relații (prin operatorii logici **AND**, **NOT**, **OR**);

⟨nr. linie 1⟩ poate fi orice număr de linie admis;

⟨instrucțiune 1⟩ poate fi orice instrucțiune executabilă exceptind instrucțiunile: **FOR**, **NEXT**, **FNEND**, **END**, **DIM**, **DEF**, **IF**, **REM**, **DATA**.

Dacă rezultatul evaluării expresiei relaționale este de tipul „condiție îndeplinită”, atunci se execută (nr. linie 1/instrucțiune 1), iar dacă este de tipul „condiție neîndeplinită”, atunci se execută (nr. linie 2/instrucțiune 2). În cazul când opțiunea **ELSE** lipsește din instrucțiune atunci nu are loc nici un salt sau nu se execută (instrucțiune 2).

De notat că prin (nr. linie 1/instrucțiune 1) și (nr. linie 2/instrucțiune 2) se înțelege fie saltul la o instrucțiune identificată prin numărul liniei de program, fie execuția unei instrucțiuni (în afara celor exceptate), în funcție de modul în care este scrisă instrucțiunea. În funcție de evaluarea expresiei relaționale se disting cazurile:

1. Relație adevărată

a) Dacă instrucțiunea este de forma:

IF (relație) **THEN** (nr. linie 1)

atunci se execută instrucțiunea care are numărul de linie (nr. linie 1). De exemplu, instrucțiunea

20 IF A=8 THEN 80

are ca efect saltul la instrucțiunea din linia 80 și execuția acesteia.

b) Dacă instrucțiunea este de forma:

IF (relație) **THEN** (instrucțiune 1)

atunci se execută instrucțiunea executabilă (instrucțiune 1) iar programul continuă cu următoarea instrucțiune executabilă (v. instrucțiunile exceptate).

De exemplu:

20 IF A > B THEN Y=5

are ca efect, atribuirea lui 5 variabilei Y (relație adevărată!).

Observație. În ambele situații opțiunea **ELSE** este ignorată.

2. Relație falsă

a) dacă opțiunea **ELSE** nu este specificată, atunci execuția programului va continua cu următoarea instrucțiune executabilă;

b) dacă opțiunea **ELSE** este specificată, atunci continuarea execuției programului va avea loc în conformitate cu cazurile a și b din situația precedentă.

De exemplu:

10 IF A=3 THEN 20 ELSE 180

are ca efect saltul la instrucțiunea din linia 180 și execuția acesteia (relație falsă!).

Observație. În ambele situații opțiunea **THEN** este ignorată.

Aplicație. Codificați în BASIC-PRAE următoarele situații:

- a) Dacă valoarea lui A este mai mică decât 20 atunci mergi la linia 30.
- b) Dacă valoarea lui C este mai mică sau egală cu cea a lui B ridicată la puterea a treia atunci afișează „EROARE”.
- c) Dacă de cinci ori valoarea lui A plus de trei ori valoarea lui B nu este egală cu 20 atunci afișează „EROARE”, în caz contrar mergi la linia 200.

d) Dacă valoarea lui A este 2 atunci mergi la linia 800, în caz contrar mergi la linia 20.

-
- a) 10 IF A<20 THEN 30
 b) 20 IF C<=B↑3 THEN PRINT "EROARE"
 c) 30 IF(5*A+3*B)<>20 THEN PRINT "EROARE" ELSE 200
 d) 40 IF A=2 THEN 300 ELSE 20

TEST

1) Se consideră următorul program:

```
10 FOR I=1 TO 9
20   READ A
30   IF A<0 THEN 50 ELSE PRINT "A="; A
40
50 NEXT I
60 DATA 1, 2, 0, -2, 5, -3, 8, 9, -8
70 END
```

- a) Când A este mai mare sau egal cu zero, la ce linie se va face saltul?
 b) Pentru care din valorile conținute de blocul de date DATA se afișează mesajul "A="?
 c) Dacă condiția A<0 este adevărată va mai fi afișată valoarea lui A?
 d) Modificați instrucțiunea IF-THEN-ELSE astfel încât programul să afișeze numai numerele diferite de zero?

2) Se consideră următorul program:

```
10 DIM A (20)
20 READ N
30 S=0
40 FOR I=1 TO N
50   READ A(I)
60   IF A(I)=0 THEN 80 ELSE
      S=S+A(I)
70 NEXT I
80 PRINT "S="; S
90 DATA 8, -6, 7, 0, 8, -3, -5, -2, -9
100 END
```

- a) Care sînt funcțiunile programului?
 b) De cîte ori se execută bucla FOR-NEXT?
 c) Care este valoarea finală a variabilei S?

Tehnici de implementare a structurilor de selecție. Metodologie

Codificarea selecției. Codificarea în BASIC structurat a selecției cu două ramuri este de forma IF-THEN-ELSE.

IF <condiție> THEN		DACA <condiție> ATUNCI
<G _I D>		<G _I D>
ELSE	în traducere:	IN CAZ CONTRAR
<G _I N>		<G _I N>
ENDIF		SFIRȘIT DACA

Codificarea structurii cu o ramură în BASIC structurat este de forma "IF-THEN".

IF <condiție> THEN	DACA <condiție> ATUNCI
<G _{I_D} >	<G _{I_D} >
ENDIF	SFIRȘIT DACA

Simularea selecției

BASIC structurat → traducere → BASIC standard

IF <condiție> THEN	IF <complementul condiției> GOTO e1
<G _{I_D} >	<G _{I_D} >
ELSE	GOTO e2
	e1
<G _{I_N} >	<G _{I_N} >
ENDIF	e2 <instrucțiunea următoare>

Observație. Pentru traducere se folosește complementul condiției.

Simularea selecției simple

BASIC structurat → traducere → BASIC standard

IF <condiție> THEN	IF (complementul condiției) GOTO e1
<G _{I_D} >	<G _{I_D} >
ENDIF	e1 <instrucțiunea următoare>

Metodologie de traducere

1. Alegem un număr de etichetă e1 nefolosit în program spre a fi utilizat în instrucțiunea **GO TO**.
2. Folosind complementul condiției, scriem instrucțiunea

IF <complementul condiției> **GOTO** e1

3. Scriem instrucțiunile BASIC standard pentru grupul de instrucțiuni G_{I_D}.
4. Pentru **selecția simplă**: completați procesul de traducere plasând imediat după ultima instrucțiune din G_{I_D}.

e1 <instrucțiunea următoare>

Pentru **selecție** alegem un număr de etichetă nefolosit e2 și adăugăm instrucțiunea **GOTO** e2 după ultima instrucțiune în G_{I_D}. Atașăm apoi numărul de linie e1 la prima instrucțiune în G_{I_N} și listăm instrucțiunile în G_{I_N}.

5. Pentru **selecție** adăugăm:

e2 <instrucțiunea următoare>

imediat după ultima instrucțiune în G_{I_N}.

Aplicație. Codificați în BASIC-PRAE următoarele situații:

- a) Dacă valoarea lui A este mai mică decât 20 atunci: B, C, D și E au valoarea 5.
- b) Dacă valoarea lui A este mai mică decât 20 atunci: adună 5 la B, C, D, E iar în caz contrar adună 8.3 la X, Y, Z.

```

a) 10 IF A >= 20 THEN 30
    20 B=5 : C=5 : D=5 : E=5
    30

b) 10 IF A >= 20 THEN 40
    20 B=B+5 : C=C+5 : D=D+5 : E=E+5
    30 GO TO 50
    40 X=X+8.3 : Y=Y+8.3 : Z=Z+8.3
    50
    
```

Operatorii logici AND, OR

Într-o instrucțiune **IF-THEN-ELSE** se pot defini **condiții compuse** utilizând operatorii logici binari **AND**, **OR**. Se permite, de asemenea, utilizarea operatorului unar de complementare **NOT**, aplicat unei relații.

Reguli

- Dacă două relații sînt reunite prin operatorul logic **AND** (SI logic – corespunde simbolului matematic "X") perechea relațiilor este adevărată numai dacă ambele relații sînt adevărate (v. figura 5.9).

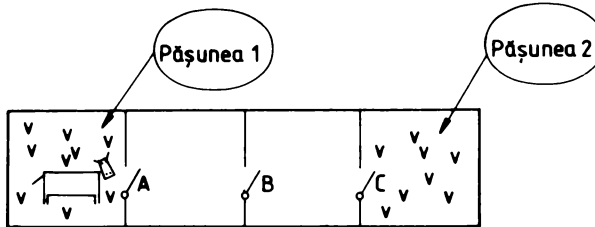


Fig. 5.9.

În figura 5.9, dacă poarta A și B și C sînt deschise, vaca Viorica poate să se ducă de pe pășunea 1 spre pășunea 2. Dacă una din porți este închisă Viorica nu poate trece.

- Dacă două relații sînt reunite prin operatorul logic **OR** (SAU logic – corespunde simbolului matematic "+"), perechea relațiilor este adevărată dacă cel puțin una dintre relații este adevărată. (v. figura 5.10).

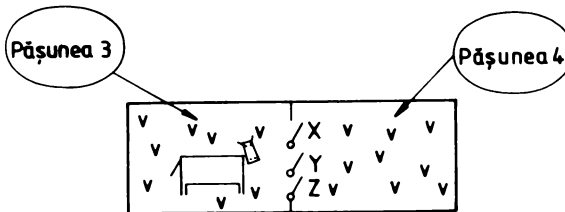


Fig. 5.10.

În figura 5.10, dacă poarta X sau Y sau Z sînt deschise, Viorica poate trece din pășunea 3 la pășunea 4.

În ciuda dificultăților întîlnite cu ocazia prezentării operatorilor logici, folosirea lor spre deosebire de notații este foarte simplă.

Aplicație. Următorul program ne ajută să înțelegem mai bine logica operatorului **AND**.

```

10 REM DA=1; NU=0
20 INPUT "POARTA A ESTE DESCHISĂ (1/0)"; A
30 INPUT "POARTA B ESTE DESCHISĂ (1/0)"; A
40 INPUT "POARTA C ESTE DESCHISĂ (1/0)"; A
50 PRINT
60 IF (A=1) AND (B=1) AND (C=1) THEN PRINT "VIORICA POATE PĂȘTE PE PĂȘUNEA 2"
ELSE PRINT "VIORICA RĂMINE PE PĂȘUNEA 1"
70 END

```

Liniile 20, 30 și 40 fixează poziția porții de intrare: deschis (1) închis (0).

Linia 60 reprezintă cheia programului. Dacă toate porțile sînt deschise „VIORICA POATE PĂȘTE PE PĂȘUNEA 2”. Dacă oricare din porți este închisă „VIORICA RĂMINE PE PĂȘUNEA 1”.

TEST

Folosind programul de mai sus ca model scrieți un altul care să sugereze logica operatorului **OR** (de preferat ca personajul principal să rămînă tot VIORICA).

```

60 IF (A=1) OR (B=1) OR (C=1) THEN PRINT "POARTA ESTE DESCHISĂ. VIORICA POATE TRECE" ELSE PRINT "VIORICA RĂMINE PE PĂȘUNEA 1".

```

Aplicații

a) Iați un program simplu care utilizează de asemenea, condițiile compuse într-o instrucțiune **IF-THEN-ELSE**. Un student promovează examenul dacă a obținut la scris cel puțin 5 sau dacă la 2/3 din semestru a obținut o notă mai mare decît 7 la laborator și cel puțin 8 la testele din timpul anului.

```

10 INPUT "NOTA DE LA SCRIS"; S
20 INPUT "NOTA DE LA LABORATOR"; L
30 INPUT "NOTA DE LA TESTE"; T
40 IF (S>=5) OR ((L>7) AND (T>=8)) THEN PRINT "PROMOVAT"
ELSE PRINT "CĂZUT"
50 END

```

b) Următorul program studiază dacă două numere (citite dinamic) sînt ambele pozitive, negative sau de semne contrare.

```

10 INPUT "PRIMUL NUMĂR ESTE"; A
20 INPUT "AL DOILEA NUMĂR ESTE"; B
30 IF (A=0) AND (B=0) THEN PRINT "AMBELE POZITIVE"
40 IF (A<0) AND (B<0) PRINT "AMBELE NEGATIVE"
ELSE PRINT "SEMNE DIFERITE"
50 END

```

c) Scrieți un program pe care să-l executați înainte de a vă culca. Se vor programa întrebări de tipul: a) "Ați curățit încălțămîntea... soțului?"; b) "Ați pus ceasul să sune?"; c) "Ați stins lumina pe hol?"; d) "Au adormit cei patru copii?"; e) "Ați oprit gazele?" ș.a.m.d. Rulînd acest program puteți scăpa de eventuale cașmaruri!

Înapoi la program

Ne-am luat cu vorba și n-am spus nimic în legătură cu variabilele indexate din cadrul programului.

```

20 INPUT "NUMAR REZERVOARE";N
40 PRINT N
50 REM CITIRE (DINAMICA) SI V
ALIDARE DATE
60 DIM R(N)
70 FOR I=1 TO N
80   PRINT "R(";I;")=";
90   INPUT R(I)
100  PRINT R(I)
110  IF R(I)>0 THEN 170
120  PRINT "R(";I;")=";R(I);
"RAZA NEGATIVA"
130  PRINT "R(";I;")=";
140  INPUT R(I)
150  PRINT R(I)
160  GO TO 110
170 NEXT I
180 REM CITIRE (STATICA) DENS
ITATE BENZINA "
190 READ D
200 REM CALCULUL MASEI DE BEN
ZINA DIN REZERVOARE
210 PRINT
220 REM TIPARIRE CAP TABEL
230 PRINT "RAZA"," MASA"
232 FOR I=1 TO 28
234   PRINT "=";
236 NEXT I
240 S=0
250 FOR I=1 TO N
260   M=D*2*PI*R(I)^3
270   S=S+M
280   PRINT R(I),M
290 NEXT I
300 FOR I=1 TO 28
310   PRINT "--";
320 NEXT I
330 REM TIPARIRE REZULTAT FIN
AL
340 PRINT "MASA TOTALA=";S;"T
ONE"
350 DATA .7
360 END

```

lată, pe scurt, **particularitățile:**

- În BASIC-PRAE se pot trata variabile indexate cu unul, doi sau pînă la 255 indici.
- Numele unei variabile indexate este alcătuit dintr-un nume admis de variabila simplă urmat de una sau două expresii de indici în paranteze.
- Într-un program BASIC-PRAE este posibilă definiția prin același nume a unei variabile simple și a unei variabile indexate.
- Într-un program BASIC-PRAE declararea tablourilor se poate realiza în două moduri:
 - declarare explicită prin instrucțiunile **DIM** (vezi linia 60) și **COM**;
 - declararea implicită prin referirea la un element al tabloului (v. liniile 90, 100, 110, 260, 280).

Observație. La declararea implicită numărul de dimensiuni al tabloului este egal cu numărul indicilor variabilei indexate, fiecare dimensiune fiind egală cu 10.

- Într-un program BASIC-PRAE există posibilitatea declarării unui tablou cu dimensiuni variabile (v. linia 60 dar și linia 20 în care se citește valoarea lui N).

Aplicație. Introduceți și executați următoarele programe:

```

a) 10 REM MAX (X1, X2, ..., XN)
    20 READ N
    30 DIM X(N)
    40 FOR I=1 TO N
    50   READ X(I)
    60 NEXT I

```

```

70 DATA 10
80 DATA 10, 5, -8, 0.9,
  -28, 6, 7, 8, 9, -4
90 M=X(1)
100 FOR I=2 TO N

110 IF M>=X(I) THEN 130
120 M=X(I)
130 NEXT I
140 PRINT "MAX="; M
150 END

```

Remarcă. Încercați să realizați singuri, fără ajutorul nostru, tabela de variabile și schema logică ale acestui program.

```

b)
5 FOR I=1 TO 5
10 INPUT "INTRODUCEȚI UN
  NUMĂR"; A
20 PRINT
30 IF A>1988 THEN PRINT "A>1988":
  GOTO 70
40 IF A=1988 THEN PRINT
  "A=1988": GOTO 70
50 PRINT "A<1988"
60 PRINT
70 CLS
80 NEXT I

30 PRINT "EXEMPLU : 10, 20, 38"
40 INPUT H, M, S
50 CLS
60 C=0
70 PRINT H; " : "; M; " : "; S
80 C=C+1
90 IF C<100 THEN 80
100 S=S+1
110 IF S<60 THEN 50
120 S=0
130 M=M+1
140 IF M<60 THEN 50
150 M=0
160 H=H+1
170 IF H<=12 THEN 50
180 H=1
190 GO TO 50
200 END

c)
10 CLS
20 PRINT "INTRODUCEȚI ORA
  EXACTĂ: ORA, MINUTUL,
  SECUNDA"

```

□ Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 211

Variabile indexate numerice. Reguli

În limbajul BASIC HC-85, TIM S, SPECTRUM variabilele indexate prezintă următoarele **particularități**:

- numele unei variabile indexate este alcătuit dintr-o singură literă;
- tablourile numerice sînt memorate pe un număr de $4+2*$ (numărul dimensiunilor) $+ 5*$ (număr de elemente) octeți;
- o variabilă numerică simplă și o variabilă tablou (indexată) pot avea același nume deoarece sînt distincte ca tip.

Aplicație. Introduceți și executați următorul program:

```

10 DIM A(4)
20 FOR I=1 TO 4
40 READ A(I)
50 NEXT I
60 FOR I=1 TO 4

70 PRINT TAB 6; "A ("; I; ")="; A(I)
80 NEXT I
90 DATA 6, 9, 7, 1
100 END

```


Înapoi la program

Programul utilizează o singură variabilă indexată de tip numeric R(I).

```

90 INPUT R(I)
100 PRINT R(I)
110 IF R(I) > 0 THEN GOTO 140
120 PRINT "R ("; I; ")="; R(I); "R
    AZA NEGATIVĂ"
:
:
:
230 LET M=D*2*PI*R(I)^3
:
:
:
250 PRINT R(I), M

```

Remarcați modul în care s-a folosit această variabilă în cele șase linii de program de mai sus.

DIM

În limbajul BASIC HC-85, TIM S, SPECTRUM instrucțiunea **DIM** prezintă următoarele **particularități**:

- inițializează elementele tabloului cu zero sau blank (variabile tablou/șir);
- șterge orice tablou care are același nume cu variabila definită prin instrucțiunea curentă;
- cu o instrucțiune **DIM** poate fi definită numai o singură variabilă indexată (de tip tablou);
- există posibilitatea declarării unui tablou cu dimensiuni variabile.

Aplicație. Să se scrie un program BASIC care calculează valoarea polinomului:

$$5x^2 + 4x + 10$$

pentru $x=2$.

Indicație. Se va utiliza schema lui Horner:

```

10 REM * SCHEMA LUI HORNER *
20 READ N, X
30 DIM A(N)
40 FOR I=1 TO N
50   READ A(I)
60 NEXT I
70 P=A(N)
80 FOR I=N-1 TO 1 STEP -1
90   LET P=P*X+A(I)
100 NEXT I
110 PRINT "VALOAREA POLINOMULUI="; P
120 DATA 3, 2, 10, 4, 5
130 END

```

Remarcă. Coeficienții 10, 4 și 5 se află în vectorul A după cum urmează: 10 în A(1), 4 în A(2) și 5 în A(3). N reprezintă numărul de termeni iar X gradul polinomului. P este valoarea polinomului, ce se inițializează cu 5.

Înapoi la program

Programul utilizează instrucțiunea **DIM** pentru rezervarea spațiului de memorie corespunzător variabilei indexate R(I).

```

30 INPUT N
40 PRINT N
:
:
:
60 DIM R(N)
:
:
:

```

Remarcați în linia 60 a programului, modul de declarare a tabloului R – cu dimensiuni variabile (N), valoarea lui N fiind citită anterior, în linia 30.

IF-THEN. Reguli

Instrucțiunea BASIC care realizează luarea deciziilor este de forma **IF-THEN**. Formatul general al instrucțiunii este:

Format general
(nr. linie) IF (relație) THEN (acțiune)

în care:

(relație) este de forma unei relații între două expresii (numerice sau șiruri) sau a unei reuniuni de relații prin operatorii logici: **AND**, **NOT**, **OR**;
(acțiune) poate fi o instrucțiune **GOTO**, o altă instrucțiune sau o secvență de instrucțiuni.

Aplicație. Introduceți și executați următoarele programe:

- | | |
|---|--|
| <p>a) 10 INPUT "Introduceți un număr între 1 și 3"; a
20 IF a=1 THEN PRINT "unu"
30 IF a=2 THEN PRINT "doi"
40 IF a=3 THEN PRINT "trei"
50 GO TO 10</p> | <p>d) 10 INPUT a
20 INPUT b
30 IF a=1 AND b=2 THEN PRINT "adevărat"</p> |
| <p>b) 10 INPUT x
20 INPUT y
30 IF x=1 AND y=1 THEN PRINT "adevărat"
40 GO TO 10</p> | <p>e) 10 INPUT m
20 IF m THEN PRINT m</p> |
| <p>c) 10 INPUT a
20 INPUT b
30 IF a=1 THEN IF b=2 PRINT "adevărat"</p> | <p>f) 10 INPUT m
20 IF NOT m THEN PRINT m</p> |
| | <p>g) 10 LET a=0
20 PRINT "BASIC"
30 LET a=a+1
40 IF a<25 THEN GO TO 20</p> |
| | <p>h) 10 LET a=0
20 LET a=a+1
30 IF a<25 THEN PRINT "BASIC"
40 GO TO 20</p> |

TEST

Să se codifice în BASIC HC-85, TIM S, SPECTRUM următoarea situație: dacă $x=1$ atunci $y=1$ altfel, dacă $x=5$ mergi la linia 100.

Varianta 1

```
10 IF X=1 THEN LET Y=1
20 IF X<>1 THEN IF X=5 THEN GO TO 100
```

Varianta 2

```
10 IF X=1 THEN LET Y=1
20 IF X<>1 AND X=5 THEN GOTO 100
```

Expresii condiționale

În BASIC HC-85, TIM S, SPECTRUM operatorii logici (**AND**, **OR**, **NOT**) pot fi considerați și folosiți ca funcții:

OPERATOR	Expresie condițională	Rezultat
AND	a AND b	a dacă b < > 0 0 dacă b = 0
OR	a OR b	1 dacă b < > 0 a dacă b = 0
NOT	NOT a	0 dacă a < > 0 1 dacă a = 0

Aplicații

a) Se consideră programul

```

10 LET a=0
20 LET b=a
30 LET a=a+1
40 IF a>31 THEN LET a=0
50 PRINT "b=" ; b
60 GO TO 20

```

Putem economisi o linie de program modificând linia 30.

```
30 LET a=a+1 AND a<31
```

Modul în care lucrează linia 30 este puțin mai complex. Dacă valoarea lui a este inferioară lui 31 atunci adaugă 1 la valoarea lui a dar dacă valoarea lui a nu este inferioară lui 31 atunci inițializează-l pe a cu zero. Ștergeți acum linia 40 (pentru a nu avea redundanțe) și renumerați din 10 în 10 instrucțiunile programului. Vom obține:

```

10 LET a=0
20 LET b=a
30 LET a=a+1 AND a<31
40 PRINT "b=" ; b
50 GO TO 20

```

b) Se consideră programul

```

10 LET x=15
20 IF b=5 AND x>1 THEN
  LET x=x-2
30 IF b=8 AND x<30 THEN
  LET x=x+2
40 PRINT "x=" ; x
50 GO TO 20

```

Puteți obține același rezultat (x=) cu: sau cu:

```

10 LET x=15
20 LET x=x-(b=5 AND x>1) *
  2+(b=8 AND x<30) * 2
40 PRINT "x=" ; x
50 GO TO 20
10 LET x=15
20 LET x=x-(2 AND b=5 AND
  x>1)+(2 AND b=8 AND x<30)
40 PRINT "x=" ; x
50 GO TO 20

```

Dacă doriți, puteți citi ultima linie 20 și cu voce tare: atribuie lui x valoarea: x-(2, dacă b=5 și valoarea lui x este superioară lui 1, în caz contrar 0)+(2 dacă b=8 și valoarea lui x este inferioară lui 30, în caz contrar 0).

Remarcă. Valorile ADEVĂRAT și FALS sint reprezentate respectiv prin 1 și zero.

```

10 INPUT a
20 INPUT b
30 LET x=(a=b)
40 PRINT "x=" ; x
50 GO TO 10

```

Parantezele din linia 30 nu sint absolut necesare dar ele permit o înțelegere mai ușoară a sensului expresiei. Expresia dintre paranteze nu ia decît valoarea 1 sau 0 atunci cînd a este sau nu este egal cu b.

TEST

Precizați valoarea lui x .

```
10 INPUT a
20 INPUT b
30 LET x=(a=b) * 3
40 PRINT "x="; x
50 GOTO 10
```

De această dată veți obține o valoare $x=0$ pentru $a \neq b$ și o valoare $x=3$ atunci când $a=b$.

Înapoi la program

Programul utilizează o singură dată instrucțiunea **IF**, în linia 110. Remarcați între **IF** și **THEN** condiția $R(I) > 0$

```
110 IF R(I) > 0 THEN GOTO 140
```

iar după **THEN** instrucțiunea de salt necondiționat **GOTO**.

TEST

Rescrieți secvența de instrucțiuni din liniile 70–140 ale programului astfel încât pentru procesul de validare date, să realizați o structură de iterație de tip **CIT TIMP**.

```
70 FOR I=1 TO N
80 PRINT "R(" ; I; ")=" ;
90 INPUT R (I)
100 PRINT R (I)
110 IF R (I) > 0 THEN GO TO 140
120 PRINT "R (" ; I; ")=" ; R (I); "RAZA NEGATIVĂ"
125 PRINT "R (" ; I; ")=" ;
130 INPUT R (I)
135 GO TO 110
140 NEXT I
```

Aplicație. Introduceți și executați următoarele programe:

```
a) 10 FOR I=1 TO 10
20 IF I=9 THEN GO TO 40
30 NEXT I
40 PRINT "ÎN LINIA 40, I="; I
50 FOR I=1 TO 2
60 FOR J=1 TO 10
70 PRINT I; J,
80 NEXT J
90 NEXT I
100 END
```

```
b) 10 REM MAX (X1, X2, ..., XN)
20 REM MIN (X1, X2, ..., XN)
30 READ N
40 DIM X(N)
50 FOR I=1 TO N
60 READ X(I)
70 NEXT I
80 DATA 10
90 DATA 3, -8, 0, 7, 15, -25, -30, 8, 9, 10
100 LET M1=X(1)
110 LET L=1
```

```

120 LET M2=X(1)
130 LET K=1
140 FOR I=2 TO N
150     IF X(I) > M1 THEN GOTO 220
160     IF X(I) >= M2 THEN GOTO 220
170     LET M2=X(I)
180     LET K=I
190     GOTO 220
200     LET M1=X(I)
210     LET L=I
220 NEXT I
230 PRINT "MAX"; M1; "POZIȚIA"; L, "MIN="; M2; "POZIȚIA"; K
240 END

```

Remarcă. Încercați să realizați singuri, fără ajutorul nostru, specificațiile de programare și documentația de proiectare ale acestui program.

```

c) 10 PRINT "INTRODUCEȚI UN NUMĂR INTREG";
    20 INPUT N
    30 LET S=0
    40 FOR I=1 TO N
    50     LET S=S+(N/I=INT (N/I))
    60 NEXT I
    70 PRINT "S="; S
    80 END
    RUN

```

INTRODUCEȚI UN NUMĂR INTREG ? 1949
S=2

```

d) 10 FOR I=1 TO 20
    20     PRINT I; 20/I; INT (20/I); 20/I-INT (20/I)
    30 NEXT I
    40 END

```

Remarcă. Funcția **INT** (x) furnizează partea întreagă a lui x. Formatul general este:

Format general
INT ((expresie numerică))

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 212

Variabile indexate (numerice). Reguli

În limbajul BASIC-AMSTRAD variabilele indexate (numerice) prezintă următoarele **particularități**:

- numele unei variabile indexate este alcătuit dintr-un nume admis de variabilă simplă urmat de un ansamblu de indici cuprins între paranteze; indicii pot fi: nume de variabile, expresii, funcții care prin evaluare trebuie să dea o valoare întreagă pozitivă;
- este posibilă definirea prin același nume a unei variabile simple și a unei variabile indexate;
- valoarea minimă a unui indice este zero (primul element al unui tablou);
- variabilele indexate (numerice) pot fi de două tipuri: întreg, real. Tipul întreg se indică prin caracterul "0/" plasat după numele simbolic al variabilei.

În cadrul programului variabila indexată numerică cu un indice este **R(I)**.

DIM

În limbajul BASIC-AMSTRAD instrucțiunea **DIM** prezintă următoarele particularități:

- declararea explicită a unui tablou (prin **DIM**) trebuie plasată, în program înainte de prima referire a unui element al tabloului;
- dacă dimensiunea (**DIM**) unei variabile indexate nu este specificată atunci ea este stabilită implicit la 10;
- instrucțiunea **DIM** stabilește ca valoare inițială zero a tuturor variabilelor din tablou;
- când un tablou nu mai este necesar el poate fi șters (**ERASE**) cu instrucțiunea **ERASE**, al cărei format este

Format general		
<număr linie>	ERASE	<listă variabile>

De exemplu

```
10 DIM b (100)
20 ERASE b
```

are ca efect ștergerea din memorie a vectorului b în vederea utilizării acesteia pentru alte scopuri;

- într-un program BASIC-AMSTRAD există posibilitatea declarării unui tablou cu dimensiuni variabile.

În cadrul programului s-a declarat vectorul R

```
120 DIM R (N)
```

cu instrucțiunea **DIM** de dimensiuni variabile (valoarea lui N se citește în linia 90 a programului).

Aplicație. Se consideră vectorii:

$$a = [1 \ 2 \ 3]$$

și

$$b = [4 \ 5 \ 6]$$

Să se scrie un program BASIC care calculează produsul vectorial

$$a \times b = [-3, 6, -3]$$

```
5 REM PRODUSUL VECTORIAL A DOI VECTORI
10 FOR I=1 TO 3
20 READ A(I), B(I)
30 NEXT I
40 DATA 1, 4, 2, 5, 3, 6
50 LET C(1)=A(2)*B(3)-B(2)*A(3)
60 LET C(2)=-A(1)*B(3)+B(1)*A(3)
70 LET C(3)=A(1)*B(2)-B(1)*A(2)
80 FOR I=1 TO 3
90 PRINT C(I);
100 NEXT I
110 END
```

IF-THEN-ELSE

În limbajul BASIC-AMSTRAD instrucțiunea **IF-THEN-ELSE** prezintă următoarele particularități:

- atunci cînd rezultatul condiției (plasate între **IF** și **THEN**) necesită un salt de linie, instrucțiunea se poate formula ca în exemplul de mai jos:
 10 **IF** x=1 **THEN** 200
 sau:
 10 **IF** x=1 **GOTO** 200
 sau:
 10 **IF** x=1 **THEN GO TO** 200
- (relație) (v. formatul instrucțiunii din limbajul BASIC-PRAE) este de forma unei relații între două expresii (numerice sau șiruri) sau a unei reuniuni de relații prin operatorii logici: **AND**, **NOT**, **OR**, **XOR** (sau exclusiv).

Aplicație. Introduceți și executați următoarele programe:

- a) 10 **INPUT** "Astăzi sîntem în"; ziua
 20 **INPUT** "Precizați numărul lunii"; luna
 30 **IF** Ziua=1 **AND** luna=9 **THEN** 50
 40 **CLS**: **GOTO** 10
 50 **PRINT** "La mulți ani pentru 1 septembrie"
- b) 10 **CLS**
 20 **INPUT** "Precizați numărul lunii"; luna
 30 **IF** luna=12 **OR** luna=1 **OR** luna=2 **THEN** 50
 40 **GO TO** 10
 50 **PRINT** "Sîntem în anotimpul iarnal"
- c) 10 **CLS**
 20 **INPUT** "Precizați numărul lunii"; luna
 30 **IF NOT** (luna=6 **OR** luna=7 **OR** luna=8) **THEN** 50
 40 **GOTO** 10
 50 **PRINT** "Nu sîntem în anotimpul vara!"
- d) 10 **INPUT** "Astăzi sîntem în"; ziua
 20 **INPUT** "Precizați numărul lunii"; luna
 30 **IF NOT** (luna=12 **OR** luna=1) **AND** ziua=29 **THEN** 50
 40 **CLS** : **GOTO** 10
 50 **PRINT** "Nu este nici decembrie nici ianuarie dar poate fi un an bisect"

TEST

Care este rezultatul execuției următoarelor instrucțiuni

- a) 10 **PRINT** 10 **AND** 10
 R . 10
- b) 10 **PRINT** 10 **AND** 12
 R . 8
- c) 10 **PRINT** 10 **AND** 1000
 R . 8
- d) 10 **PRINT** 1000 **OR** 10
 R . 1002

Observație. **XOR** (sau exclusiv) dă un rezultat adevărat dacă cele două argumente sînt diferite (vezi tabela de adevăr).

ARGUMENTE		OPERATOR		
A	B	AND	OR	XOR
1010	0110	0010	1110	1100

Aplicație. Introduceți și executați următorul program:

```

10 CLS
20 A=0
30 PRINT "ZERO"
40 INPUT C$
50 CLS
60 A=A+1
70 IF A=1 THEN PRINT "UNU"
80 IF A=2 THEN PRINT "DOI"
90 IF A=3 THEN PRINT "TREI"
100 IF A=4 THEN PRINT "PATRU"
110 IF A=5 THEN PRINT "CINCI"
120 IF A<=5 THEN 40
130 GO TO 10
140 END

```

Remarci

- instrucțiunea **INPUT** din linia 40 are o importanță deosebită pentru bună desfășurare a lucrurilor din cadrul acestui program. Fără această instrucțiune, calculatorul ar fi afișat mesajele din liniile 70/80/90/100/110 doar pentru o fracțiune de secundă, căci urmare a execuției instrucțiunilor din liniile 120 și 130 ecranul s-ar fi șters imediat. Aceasta demonstrează valoarea inserării instrucțiunilor de tipul

```
40 INPUT C$
```

și a răspunde la aceste instrucțiuni cu o valoare nulă – prin simpla apăsare pe tasta [RETURN] numai după ce ați avut timp să vedeți rezultatele afișate pe ecran.

- Folosirea instrucțiunii

```
130 GO TO 10
```

dă naștere la o **buclă infinită**, programul neavând posibilitatea să se oprească de la sine. De fapt, există o instrucțiune **END** în linia 140 dar cum ea este precedată de instrucțiunea **GOTO** înseamnă că instrucțiunea **END** nu va fi niciodată executată. În acest fel, programul nu se sfârșește niciodată iar calculatorul va lucra până la scoaterea lui din funcțiune.

Ei bine, de fapt există o cale de ieșire din această buclă cauzată de instrucțiunea **GOTO** și anume acționarea tastei [ESC]. Deci, o cale de ieșire dintr-o buclă o reprezintă apăsarea tastei [ESC]. După aceasta, puteți modifica programul, puteți introduce instrucțiuni **PRINT** pentru a lista valorile variabilelor, puteți scrie un nou program ori să-l reluați pe cel vechi introducând o comandă **RUN**.

TEST

Precizați rezultatul execuției următoarelor instrucțiuni BASIC:

- 10 PRINT "<=>"; : GO TO 10
- 10 PRINT "<=>" : GO TO 10
- 10 PRINT "<=>", : GO TO 10
- 10 PRINT TAB (16); "<=>" : GO TO 10

Aplicație. Introduceți și executați următoarele programe:

- ```

10 FOR I=1 TO 6
20 INPUT "A, B"; A, B
30 IF A>B THEN PRINT "PRIMUL NUMĂR"; A; "ESTE MAI MARE DECIT AL DOILEA" : PRINT "NUMĂR"; B : GOTO 70
40 IF A=B THEN PRINT "PRIMUL NUMĂR"; A; "ESTE EGAL CU AL DOILEA" : PRINT "NUMĂR"; B : GOTO 70
50 PRINT "PRIMUL NUMĂR"; A; "ESTE MAI MIC DECIT AL DOILEA" : PRINT "NUMĂR"; B
70 NEXT I

```



- ```

b) 5 FOR I=1 TO 4
    10 INPUT "A, B, C"; A, B, C
    20 IF A*B>A+B*C THEN PRINT A; " *"; B; ">"; A; "+"; B; " *";
      C; GO TO 60
    30 IF A*B=A+B*C THEN PRINT A; " *"; B; "="; A; "+"; B; " *";
      C; GO TO 60
    40 PRINT A; " *"; B; "<"; A; "+"; B; " *"; C
    60 NEXT I

c) 10 INPUT "A, B"; A, B
    20 IF A>B THEN PRINT "A>" : PRINT "B" : GOTO 60
    30 IF A=B THEN PRINT "A=" : PRINT "B" : GO TO 60
    40 PRINT "A<" : PRINT "B"
    50 PRINT
    60 PRINT "TASTATI 1 PENTRU CONTINUARE"
    70 PRINT "TASTATI 0 PENTRU OPRIRE"
    80 INPUT N
    90 IF N=1 THEN 10
    100 IF N<>0 THEN 80
    110 PRINT
    120 END

d) 10 REM -32 767 și +32 767
    20 REM -----
    30 AMINUS %0 = -32760
    40 APLUS %0 = +32760
    50 PRINT "LOW VALUE", "HIGH VALUE"
    60 AMINUS %0 = AMINUS %0-1
    70 APLUS %0 = APLUS %0 + 1
    80 PRINT AMINUS %0, APLUS %0
    90 GO TO 60

e) 10 CLS
    20 FOR I=0 TO 13
    30 PRINT TAB (I), "<=>"
    40 NEXT
    50 GO TO 50

```

Observație. Remarcați efectul instrucțiunii

```
50 GO TO 50
```

Programul ciclează, ecranul rămânind tot timpul cu aceeași imagine – 14 vaporașe. Pentru ieșirea din buclă tastează [ESC].

WHILE–WEND

Instrucțiunea creează un ciclu alcătuit din mai multe instrucțiuni, atâta timp cât o condiție dată este adevărată.

Formatul general al instrucțiunii este

Format general
WHILE (secvență – instrucțiuni)
WEND

în care:

<relație> are semnificația cunoscută;

<secvență – instrucțiuni> reprezintă orice tip de instrucțiune BASIC-AMSTRAD.

Remarci

- **WHILE-WEND** implementează în BASIC structurat structura de iterație **CIT-TIMP**.
- **WHILE** indică punctul de intrare în ciclu.
- **WEND** precizează punctul de ieșire din ciclu.
- **WHILE-WEND** poate fi executat cel puțin de zero ori.
- Se pot folosi cicluri **WHILE-WEND** imbricate în care fiecare **WEND** închide pe cel mai apropiat **WHILE**.

Joc pe calculator

Introduceți și executați următorul program:

```

10 READ A
15 DATA 7
20 INPUT "INTRODUCEȚI UN NUMĂR (0 LA 9)"; X
30 PRINT "AȚI INTRODUS NUMĂRUL : "; X
40 WHILE A < > X
50     IF X < A THEN PRINT "NUMĂRUL"; X; "ESTE PREA MIC! INTRODUCETI
        ALTUL" : GOTO 70
60     IF X > A THEN PRINT "NUMĂRUL"; X; "ESTE PREA MARE! INTRODUCETI
        CEȚI ALTUL"
70     INPUT "INTRODUCEȚI UN NUMĂR (0 LA 9)"; X
80     PRINT "AȚI INTRODUS NUMĂRUL : "; X
90 WEND
100 PRINT "FELICITĂRI! NUMĂRUL ESTE : "; A
110 END

```

Aplicație. Să se scrie un program care calculează $\sin(x)$. Se va utiliza formula:

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \cdot \frac{x^{2n+1}}{(2n+1)!}$$

```

10 INPUT "X="; X
20 INPUT "E="; E
30 S=0
40 N=0
50 N1=1
60 T=(-1) ↑ N * X ↑ (2 * N+1)/N1
70 WHILE ABS(T) >= E
80     S=S+T
90     N=N+1
100    N1=N1 * 2 * N * (2 * N+1)
110    T=(-1) ↑ N * X ↑ (2 * N+1)/N1
120 WEND
130 PRINT "X="; X, "SIN(X)=", SIN(X), "S="; S
140 END

```

RUN

X=? 1

E=? 0.001

X=1 SIN(X)= . 84147098 S= . 84166667

Remarci

- **SIN(X)** este funcția aritmetică sinus de x . Are ca rezultat $\sin(x)$ calculat în simplă precizie, unde x este exprimat în radiani. Formatul general este:

Format general

SIN ((expresie numerică))

- **ABS(X)** dă valoarea absolută a lui X sau a expresiei numerice cuprinse între paranteze.

Formatul general este:

Format general
ABS ((expresie numerică))

Înapoi la program

În program instrucțiunea **WHILE-WEND** creează un ciclu compus din instrucțiunile:

```
170 WHILE R(I) < 0
180     PRINT "R(" ; I ; ")=" ; R(I) ;
190     PRINT "R(" ; I ; ")=" ;
200     INPUT R(I)
210     PRINT R(I)
220 WEND
```

De notat că bucla **WHILE-WEND** este inclusă în bucla **FOR-NEXT**. Remarcați, de asemenea, că în interiorul buclei **WHILE** valoarea variabilei *I* rămâne aceeași cit timp *R(I)* are o valoare negativă. La ieșirea din buclă, se dă controlul instrucțiunii **FOR-NEXT**.

Aplicație: Introduceți și executați următorul program:

<pre>10 PRINT "INTRODUCEȚI UN NU- MĂR INTREG (1-6 CIFRE)"; 20 INPUT N 30 WHILE N < > 999999 40 FOR I=5 TO 0 STEP -1 50 E=INT (N/10 ↑ I) 60 PRINT E 70 R=N-E * 10 ↑ I 80 N=R 90 NEXT I 100 PRINT "INTRODUCEȚI UN NU- MĂR INTREG (1-6 CIFRE)";</pre>	<pre>110 INPUT N 120 WEND RUN INTRODUCEȚI UN NUMAR INTREG (1-6 CIFRE)? 76321 7 6 3 2 1 INTRODUCEȚI UN NUMAR INTREG (1-6 CIFRE)? 999999</pre>
--	--

Joc pe calculator

Gândiți-vă la două numere (*A* și *B*) mai mici decît 10. Înmulțiți pe *A* cu 2 și adunați 5 la produsul obținut. Înmulțiți rezultatul obținut cu 5. Adunați la noul rezultat pe 8. Spuneți cit ați obținut și vă vom spune la ce numere v-ați gândit.

Indicație. Numărul obținut este de forma $10A+B=A$ zeci + B unități. Pentru determinarea lui *A* și *B* consultați aplicația precedentă.

TEST

Precizați care sînt rezultatele execuției următorului program. Introduceți și executați programul pe calculatorul dvs.

<pre>10 N=0 20 S=0 30 READ D 40 WHILE D < > 999 50 N=N+1 60 S=S+D 80 WEND 90 S=S/N 100 PRINT "MEDIA ARITMETICĂ=" ; S</pre>	<pre>110 120 REM BLOC DATE 130 DATA 98, 80, 73, 92, 77, 84, 83, 79, 87, 73 140 DATA 99, 63, 63, 92, 81, 93, 47, 53, 89, 100 150 DATA 98, 71, 73, 999 160 END RUN</pre>
--	--

□ Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag. 213

Variabile indexate (numerice). Reguli

În limbajul BASIC-COMMODORE variabilele indexate (numerice) prezintă următoarele **particularități**:

- numele unei variabile indexate este alcătuit dintr-un nume admis de variabila simplă urmat de unu, doi sau trei indici cuprinși între paranteze;
- indicii pot fi: nume de variabile, expresii, funcții;
- pot exista în cadrul aceluiași program variabile simple și variabile indexate cu același nume;
- valoarea minimă a unui indice este zero;
- variabilele indexate pot fi de două tipuri: întreg, real. Tipul întreg se indică prin caracterul "0/" plasat după numele simbolic al variabilei.

Revenind la program, semnalăm utilizarea de către program a variabilei indexate numerice, cu o dimensiune R(I).

DIM

Instrucțiunea **DIM** spre deosebire de cea din BASIC-AMSTRAD prezintă următoarea particularitate: atunci când dimensiunea unei variabile indexate nu este specificată atunci ea este stabilită implicit la 11.

Aplicații.

a) Se consideră vectorii:

$$a=[6 \ 2 \ 1]$$

și

$$b=[1 \ 2 \ 5].$$

Să se scrie un program BASIC care calculează

$$a+b=[7 \ 4 \ 6]$$

```

5 REM ADUNAREA A DOI          80 DATA 1, 2, 5
  VECTORI                      90 REM -----
10 FOR I=1 TO 3                100 FOR I=1 TO 3
20   READ A(I)                 110   LET C(I)=A(I)+B(I)
30 NEXT I                      120 NEXT I
40 DATA 6, 2, 1               130 FOR I=1 TO 3
50 FOR I=1 TO 3                140   PRINT C(I);
60   READ B(I)                 150 NEXT I
70 NEXT I                      160 END

```

b) Se consideră vectorii:

$$a=[3 \ 8 \ 2]$$

și

$$b=[1 \ 2 \ 3].$$

Să se scrie un program BASIC care calculează produsul scalar

$$a \cdot b=(3)(1)+(8)(2)+(2)(3)=25$$

```

5 REM * PRODUSUL SCALAR          60 P=0
  A DOI VECTORI *                70 FOR I=1 TO 3
10 FOR I=1 TO 3                   80 LET P=P+A(I) * B(I)
20 READ A(I), B(I)                90 NEXT I
30 NEXT I                          100 PRINT "PRODUSUL SCALAR=" ; P
40 DATA 3, 1, 8, 2, 2, 3         110 END
50 REM - - - - -

```

IF-THEN

În limbajul BASIC-COMMODORE instrucțiunea **IF-THEN** prezintă următoarele **particularități**:

Format general

```

(nr. linie) IF <relație> THEN { <acțiune> [ : <acțiune 1> ] ... }
                                { (nr. linie 1) }

```

- <relație> este de forma unei relații între două expresii (numerice sau șiruri) sau a unei reuniuni de relații prin operatorii logici: **AND**, **NOT**, **OR**;
- <acțiune> poate fi o instrucțiune **GOTO** sau o altă instrucțiune. După **THEN** poate urma o secvență de instrucțiuni.

Aplicație. Precizați rezultatul execuției următoarelor programe:

```

a) 10 X=13                          b) 10 X=80
   20 Y=40                            20 Y=15
   30 IF X>10 AND Y>20                30 IF Y=15 PRINT Y : PRINT X
      THEN 50                          RUN
   40 PRINT "CONDIȚIILE NU            15
      SINT ÎNDEPLINITE" : END          80
   50 PRINT "CONDIȚIILE SINT
      ÎNDEPLINITE"
   RUN
   CONDIȚIILE SINT ÎNDEPLINITE

```

În program instrucțiunile **IF-THEN** și **GOTO** s-au utilizat pentru simularea structurii de iterație de tip **DO WHILE**.

Aplicații

a) Să se scrie un program care calculează e^x .

Indicație. Se va utiliza formula

$$e^x = \sum_{n=0}^{10} \frac{x^n}{n!}$$

```

10 REM APLICAȚIE
20 PRINT "X=", "EXP (X)=", "S="
30 A(1)=1
40 FOR I=1 TO 9
50 A(I+1)=A(I)/(I+1)
60 NEXT I
70 FOR K=1 TO 11
80 X=(K-1) * .1
90 S=A(10)
100 FOR I=1 TO 9
110 L=10-I
120 S=S * X+A(L)

```

```

130          NEXT I
140          S=S * X+1
150          P=EXP (X)
160          PRINT X, P, S
170          NEXT K
180 END

```

RUN

X=	EXP (X)=	S=
0	1	1
.1	1.105171	1.105171
.2	1.221403	1.221403
.3	1.349859	1.349859

Remarcă. Funcția aritmetică **EXP(X)** calculează e^x . Formatul general este

Format general
EXP (<expresie numerică>)

b) Introduceți și executați următoarele programe:

b.1) 10 PRINT CHR\$ (147)	70 GOTO 10
20 A=0	90 PRINT "<=><=><=>"
30 A=A+1	100 B=0
40 IF A < 100 THEN 30	110 GO TO 20
50 IF B=1 THEN 90	120 END
60 B=1	
b.2) 10 PRINT CHR \$ (147)	80 C=1
20 C=0	90 GOTO 30
30 FOR I=1 TO 200	100 PRINT CHR\$ (147)
40 NEXT I	110 PRINT
50 IF C=1 THEN 100	120 PRINT "<=>"
60 PRINT CHR \$ (147)	130 GO TO 20
70 PRINT "<=>"	

□ Particularități ale programării în limbajul BASIC-80

vol. 2, pag. 214

Variabile indexate (numerice). Reguli

În limbajul BASIC-80 variabilele indexate (numerice) prezintă următoarele **caracteristici**:

- numele unei variabile indexate este orice nume de variabilă simplă acceptat de BASIC-80 urmat de una sau mai multe expresii întregi cuprinse între paranteze;
- este posibilă definirea prin același nume a unei variabile simple și a unei variabile indexate;
- variabilele indexate pot fi de două tipuri: întreg, real. Tipul întreg se indică prin caracterul "0/" plasat după numele simbolic al variabilei;
- valoarea minimă a unui indice este zero;
- pentru variabilele indexate se rezervă spațiu, după cum urmează: elemente întregi – 2 octeți/element; elemente simplă precizie – 4 octeți/element; elemente dublă precizie – 8 octeți/element.

Programul utilizează variabila indexată numerică cu un indice R(I).

DIM

În limbajul BASIC-80 instrucțiunea **DIM** prezintă următoarele particularități:

- dacă dimensiunea unei variabile indexate nu este specificată, atunci este stabilită implicit la 10;
- dacă dimensiunea maximă este depășită se afișează un mesaj de eroare;
- instrucțiunea stabilește pentru toate variabilele unui tablou, valoarea inițială zero;
- variabilele tablou ale unui program pot fi eliminate cu instrucțiunea **ERASE** (v. BASIC-AMSTRAD).

IF-THEN-ELSE și IF-GOTO-ELSE

În limbajul BASIC-80, instrucțiunea **IF-THEN-ELSE** (v. formatul general al instrucțiunii din limbajul BASIC-PRAE) prezintă următoarele particularități:

- (relație) este de forma unei relații între două expresii (numerice sau șiruri) sau a unei reuniuni de relații prin operatorii logici: **AND**, **OR**, **NOT**, **XOR** (sau exclusiv), **IMP** – implicație ($X \text{ IMP } Y=1$ dacă $X \leq Y$), **EQV** – echivalență ($X \text{ EQV } Y=1$ dacă $X=Y$);
- Operatorii relaționali admiși sînt: "="; "<"; ">"; "<=" sau "<="; ">=" sau ">="; "<>"; "<>" sau "><";
- Este posibil ca o instrucțiune **IF** să includă o altă instrucțiune **IF**;
- Cînd se folosește **IF** pentru a testa mărimea rezultatului unei operații cu numere în virgulă mobilă, se va ține seama de faptul că valoarea nu poate să fie exactă.

În următorul exemplu:

```
10 IF ABS (M-1.0) < 1.0E-6 THEN 500
```

valoarea lui M este 1.0 cu o aproximație de $1E-6$ (10^{-6}).

Aplicație. Se consideră următorul program BASIC:

```
10 GREȘIT=0
20 FOR I=1 TO 10
30   READ N1, N2
40   PRINT N1; "X"; N2; "=";
50   INPUT RĂSPUNS
60   IF RĂSPUNS <> N1 * N2 THEN GREȘIT=GREȘIT+1
70 NEXT
80 PRINT "AȚI DAT"; GREȘIT; "RĂSPUNSURI GREȘITE"
90 DATA 5, 5, 4, 4, 6, 6, 7, 7, 8, 8
100 DATA 4, 6, 4, 7, 5, 8, 5, 9, 6, 9
110 END
```

a) Introduceți și executați programul.

b) Adăugați instrucțiunile

```
65 IF RĂSPUNS <> N1 * N2 THEN PRINT "RĂSPUNS GREȘIT"
66 IF RĂSPUNS <> N1 * N2 THEN PRINT "RĂSPUNSUL ESTE"; N1 * N2
```

și executați programul.

c) Adăugați instrucțiunile:

```
60 IF RĂSPUNS <> N1 * N2 THEN GREȘIT=GREȘIT+1 : PRINT
   "RĂSPUNS GREȘIT" : PRINT "RĂSPUNSUL ESTE"; N1 * N2
65
66
```

și executați programul.

d) Adăugați instrucțiunea

```
60 IF RĂSPUNS <> N1 * N2 THEN GREȘIT=GREȘIT+1 :
   PRINT "RĂSPUNS GREȘIT" : PRINT "RĂSPUNSUL ESTE";
   N1 * N2 ELSE PRINT "FELICITĂRI"
```

și executați programul.

TEST

Precizați rezultatele execuției următoarelor programe:

a) 10 X=8
20 Y=2
30 IF X>Y THEN PRINT "X>Y" ELSE IF
Y>X THEN PRINT "Y>X" ELSE PRINT "X=Y"

R. X>Y

b) 10 A=2
20 B=5
30 C=7
10 IF A=B THEN IF B=C THEN PRINT "A=C" ELSE PRINT "A<>C"

R. A<>C

Observație. Limbajul BASIC-80 admite pentru instrucțiunea IF și formatul:

Format general
(nr. linie) IF <relație> GOTO (nr. linie 1/instrucțiune 1) [ELSE (nr. linie 2/instrucțiune 2)]

WHILE-WEND

WHILE-WEND creează un ciclu compus din mai multe instrucțiuni BASIC-80 atita timp cit o condiție dată este adevărată. Formatul general și remarcile privind instrucțiunea **WHILE-WEND** sînt identice cu cele de la limbajul BASIC-AMSTRAD cu singura deosebire că (secvență instrucțiuni) reprezintă orice tip de instrucțiune BASIC-80.

În cadrul programului instrucțiunile **WHILE** și **WEND** din liniile 170 și 220 formează o buclă pentru validarea razelor rezervoarelor.

Înapoi la program

V-ați pus vreodată întrebarea ce s-ar întîmpla dacă pentru „Număr rezervoare” am introduce, evident din greșeală, un număr zecimal, 5.60 de exemplu! Vă rugăm să încercați chiar în acest moment. Ce-ați constatat?

Deși calculatorul nu ne semnalează nici o greșeală (!), totuși știm bine că există. Pentru a evita astfel de situații vă recomandăm să utilizați

o variabilă de tip întreg ($N^0/0$) sau să introduceți în cadrul programului o secvență de instrucțiuni care să vă asigure că pentru „număr rezervor” s-a citit un număr întreg.

```
80 PRINT "Număr 'rezervoare=";
90 INPUT N
95 WHILE N-INT(N)<>0
96     INPUT "Număr rezervoare!"; N
98 WEND
```

Jocuri pe calculator

a) Introduceți și executați următoarele programe:

```
a.1) 10 RĂSPUNS=0
      20 A=11
      30 B=12
      40 WHILE RĂSPUNS<>A*B
      50     PRINT A; "X"; B; "=";
      60     INPUT RĂSPUNS
      70 WEND
      80 END

a.2) 10 RĂSPUNS=91
      20 WHILE RĂSPUNS>80
      30     INPUT "INTRODUCEȚI UN
      NUMĂR<80"; RĂSPUNS
      40 WEND
      50 END

a.3) 10 SCOR=0
      20 RĂSPUNS=0
      30 A=15
      40 WHILE RĂSPUNS<>A*A
      AND SCOR<4
      50     PRINT A; "X"; A; "=";
      60     INPUT RĂSPUNS
      70     SCOR=SCOR+1.
      80 WEND
      90 END

a.4) 10 INPUT "INTRODUCEȚI UN NU-
      MĂR DE LA 3 la 8"; N
      20 WHILE N<3 OR N>8
      30     INPUT "INTRODUCEȚI UN
      ALT NUMĂR"; N
      40 WEND
      50 PRINT "FELICITĂRI!"
      60 END
```

```
a.5) 5 PRINT "PENTRU OPRIRE
      TASTAȚI A=999"
      10 INPUT "A, B"; A, B
      20 WHILE A<>999
      30     IF A>B THEN PRINT "A>";
      PRINT "B"; GOTO 60
      40     IF A=B THEN PRINT "A=";
      PRINT "B"; GOTO 60
      50     PRINT "A<"; PRINT "B"
      60     PRINT
      70     INPUT "A, B"; A, B
      80 WEND
      90 END
      RUN
      A; B ? 4, 6
      A<
      B
      A, B? 80, 80
      A=
      B
      A, B? 98, 2
      A>
      B
      A, B? 999
      ? Redo from start
      A, B? 999,0
      OK

a.6) 20 Y=0
      30 Z=Y↑2
      40 IF Z>48 THEN END
      50 PRINT TAB(Z); "<=>"
      60 Y=Y+1
      70 GO TO 30
```

Observație. Același program (a.6) poate fi scris și sub forma:

```
20 Y=0
30 Z=Y↑2
40 WHILE Z<48
50     PRINT TAB(Z); "<=>"
60     Y=Y+1
70     Z=Y↑2
80 WEND
90 END
```

b) Scrieți un program BASIC pentru înmulțirea fracțiilor:

$$\frac{1}{2} \cdot \frac{4}{3}; \quad \frac{2}{9} \cdot \frac{6}{8}$$

```
10 READ N1; D1
20 DATA 1, 2
30 WHILE N1<>999
```

```
40 READ N2, D2
50 DATA 4, 3
60     N=N1*N2
```

70	N3=N	160	R=N-I*D
80	D=D1*D2	170	WEND
90	D3=D	180	PRINT "PRODUSUL ESTE"; N3/D; "/"; D3/D
100	I=INT (N/D)	190	READ N1, D1
110	R=N-I*D	200	WEND
120	WHILE R<>0	210	DATA 2, 9, 6, 8
130	N=D	220	DATA 999, 0
140	D=R	230	END
150	I=INT(N/D)		

TEST

Precizați rezultatul execuției următorului program:

10	READ A, B	60	WEND
15	DATA 3, 4	70	DATA 35, 7, 99, 34, 80, 25
20	WHILE A<>9999	80	DATA 838, 21, 97, 13, 80, 40
30	R=A-INT(A/B)*B	90	DATA 9999, 0
40	PRINT A; B; R; INT (A/B)	100	END
50	READ A, B		

GOSUB și RETURN

Să nu uităm de modulele comune de prelucrare: RIND și CIT-R (v. figurile 5.3 și 5.4) care așteaptă momentul programării lor în BASIC. Cum vom proceda? Pentru început, trebuie să ne preocupe locul de plasare a acestora.

Practica demonstrează necesitatea plasării lor la sfârșitul programului (v. instrucțiunea **STOP**), între **STOP** și **END**, având grijă ca ultima instrucțiune a fiecărui modul (putem să-l numim subprogram sau subrutină) să fie instrucțiunea **RETURN**. De notat că o subrutină BASIC nu are nume. Cum poate fi atunci apelată? Subrutinele pot fi apelate din orice punct al programului, chiar și din cadrul unei alte subrutine (pot exista situații când o subrutină se poate reapela) cu ajutorul instrucțiunii **GOSUB** al cărei format general este:

Formatul general	

	GOSUB (număr linie)
	.
(număr linie)	.
	.
	.
	RETURN

în care (număr linie) reprezintă numărul de linie al primei instrucțiuni din cadrul subrutinei.

Mecanismul (v. figura 5.11) este următorul. Instrucțiunea **GOSUB** lansează în execuție subrutina care începe la numărul de linie indicat în formatul general prin (număr linie) și reține instrucțiunea care urmează în secvență. Instrucțiunea **RETURN** marchează sfârșitul execuției subrutinei și redă controlul programului principal, urmînd a se executa instrucțiunea imediat următoare după **GOSUB**.

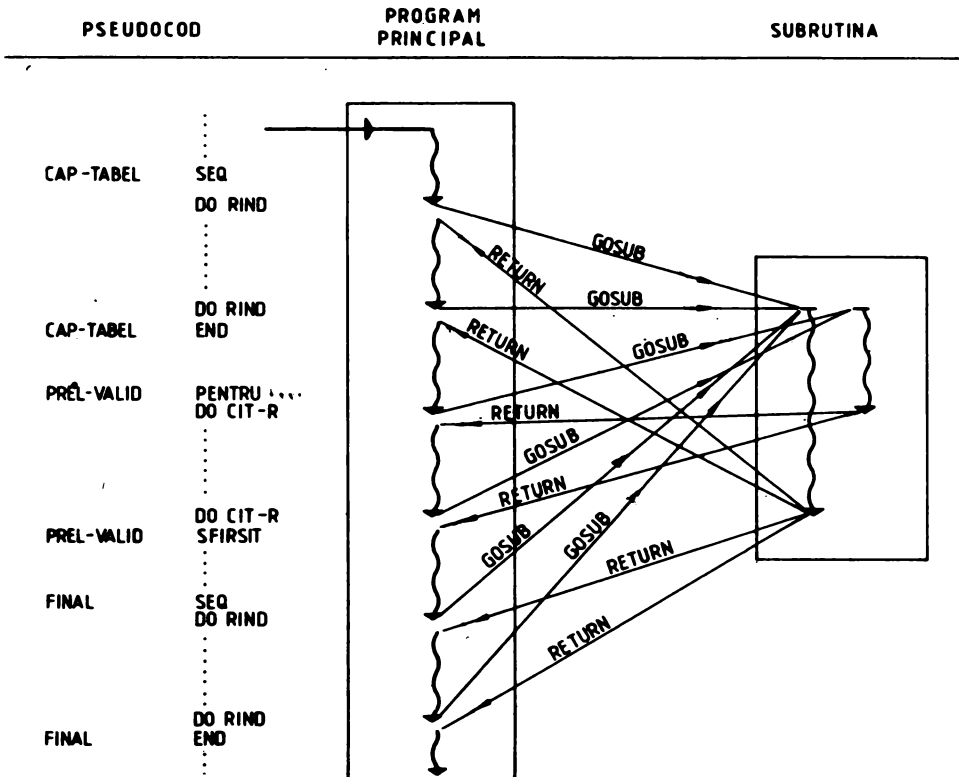


Fig. 5.11. Mecanismul de funcționare a instrucțiunilor **GOSUB** și **RETURN**.

Remarci

- Subrutinele sînt folosite pentru a realiza operații care se repetă frecvent.
- Avantajele lucrului cu subrutinele sînt: economisirea de timp la scrierea programului, reducerea consumului de memorie, creșterea gradului de lizibilitate a programului etc.
- O subrutină nu are argumente, schimbul de informații cu programul apelant realizîndu-se prin variabilele curente.
- Subrutina trebuie să conțină cel puțin o instrucțiune **RETURN**.
- Subrutina este activată în momentul în care în programul principal s-a citit un **GOSUB**.
- Instrucțiunea **RETURN** dezactivează subrutina.
- Subrutinele pot fi imbricate.
- Revenirea dintr-o subrutină apelată într-o subrutină apelantă se realizează cu respectarea regulii revenirii la punctul imediat următor punctului de ieșire.
- Profunzimea imbricării subrutinelor depinde numai de spațiul de memorie disponibil.

Aplicații

a) Să se calculeze expresia:

$$E = \frac{3! + 6!}{2 + 7!}$$

10 N=3
20 **GOSUB** 500

30 F3=F
40 N=6

```

50 GOSUB 500
60 F6=F
70 N=7
80 GOSUB 500
90 F7=F
100 E=(F3+F6)/(2+F7)
110 PRINT "E="; E
120 STOP
500 F=1
510 FOR I=1 TO N
520 F=F*I
530 NEXT I
540 RETURN
550 END

```

b) Într-un vector cu N elemente se citesc valori pozitive, negative și nule. Știind că valorile pozitive pot constitui laturile unui poligon, să se determine:

- numărul laturilor poligonului;
- perimetrul poligonului;
- aria poligonului pentru cazul cind numărul laturilor poligonului este 3.

```

10 INPUT "N="; N
20 DIM A(N)
30 GOSUB 100
40 GOSUB 200
50 PRINT "POLIGONUL ARE"; L;
  "LATURI"
60 GOSUB 300
70 PRINT "PERIMETRUL POLIGONULUI
  ESTE"; P
80 IF L=3 THEN GOSUB 400
90 STOP
95 END
100 REM*****
110 FOR I=1 TO N
120 INPUT A(I)
130 NEXT I
140 RETURN
200 REM*****
210 L=0
220 FOR I=1 TO N
230 IF A(I) <= 0 THEN 250
240 L=L+1
250 NEXT I
260 RETURN
300 REM*****
305 P=0
310 FOR I=1 TO N
320 IF A(I) <= 0 THEN 340
330 P=P+A(I)
340 NEXT I
400 REM*****
405 J=1
410 FOR I=1 TO N
420 IF A(I) <= 0 THEN 450
430 B(J)=A(I)
440 J=J+1
450 NEXT I
460 S1=0
470 FOR K=1 TO 3
480 S1=S1+B(K)
490 NEXT K
500 S1=S1/2
510 S1=1
520 FOR K=1 TO 3
530 S1=S1*(S1-B(K))
540 NEXT K
550 S1=S1↑0.5
560 PRINT "ARIA="; S1
570 RETURN

```

c) Scrieți un program pentru adunarea a două fracții.

```

5 REM ADUNAREA A DOUA
  FRAȚII
10 REM A/B+C/D
20 INPUT "A, B, C, D"; A, B, C, D
25 X=A
30 Y=B
40 GOSUB 400
50 D2=E
60 A1=A/D2
70 B1=B/D2
80 X=C
90 Y=D
100 GOSUB 400
110 D2=E
120 C1=C/D2
130 D1=D/D2
140 X=B1
150 Y=D1
160 GOSUB 400
170 D2=E
180 N3=(B1*D1)/D2
190 N4=(N3/B1)*A1+(N3/D1)*C1
200 PRINT N4
210 PRINT N3
220 X=N4
230 Y=N3
240 GOSUB 400
250 D2=E
260 F1=N4/D2
270 F2=N3/D2
280 PRINT A, B, C, D
290 PRINT F1, F2
300 STOP
320 END
400 E=X
410 F=Y
420 IF F=0 THEN 470
430 R=E-INT(E/F)*F
440 E=F
450 F=R
460 GOTO 420
470 RETURN

```

□ Particularități ale programării în limbajul BASIC-PLUS

vol. 2, pag. 215

Variabile indexate numerice. Reguli

În limbajul BASIC-PLUS variabilele indexate numerice prezintă următoarele **caracteristici**:

- numele unei variabile indexate este format dintr-un nume simbolic de variabilă, la care se adaugă una sau două expresii de indici cuprinse între paranteze;
- indicii pot fi nume de variabile, expresii, funcții care prin evaluare trebuie să dea o valoare întregă pozitivă;
- variabilele indexate pot fi de două tipuri: întreg, real; tipul întreg se indică prin caracterul "%/" plasat după numele simbolic al variabilei;
- într-un program pot exista variabile simple și variabile indexate cu același nume.

Programul utilizează variabila indexată (numerică) cu un indice R(I).

DIM

În limbajul BASIC-PLUS instrucțiunea **DIM** prezintă următoarele **particularități**:

- dimensiunea implicită a unui tablou este considerată ca fiind egală cu 10;
- mărimea spațiului alocat unui tablou depinde de dimensiunile declarate și de tipul său (întreg, real, șir) și se calculează cu formula:

$$S = \langle \text{dim } 1 \rangle * \langle \text{dim } 2 \rangle * K \text{ octeți}$$

în care:

$\langle \text{dim } 1 \rangle$ și $\langle \text{dim } 2 \rangle$ reprezintă dimensiunile tabloului și pot fi expresii numerice (întregi sau reale);

$$K = \begin{cases} 2, & \text{pentru tablouri numerice întregi,} \\ 4, & \text{pentru tablouri numerice reale sau șir.} \end{cases}$$

Observație. În calculul anterior nu intră spațiul ocupat de elementele tabloului și de caractere.

- numărul tablourilor ce pot fi utilizate într-un program este funcție numai de spațiul (maxim) necesar și de spațiul de memorie aflat la dispoziția utilizatorului;
- tabloul nu este alocat în momentul declarării lui ci atunci când se tratează efectiv un element al său; după alocare el poate fi redimensionat având în vedere dimensiunea spațiului fizic rezervat.

Aplicație. Introduceți și executați următorul program:

```
10 C=3.14159/180
20 FOR I=0 TO 80 STEP 10
30 PRINT SIN (I*C); COS (I*C); TAN (I*C)
40 NEXT I
50 END
```

Observație. **SIN**, **COS**, **TAN** calculează sinusul, cosinusul și tangenta expresiei I*C. Argumentele funcțiilor trigonometrice au fost transformate în radiani (v. linia 10).

TEST

Precizați rezultatele execuției următoarelor programe BASIC:

- a) 10 DIM A(50)
20 FOR I=1 TO 50
30 A(I)=1
40 NEXT I
- b) 10 DIM A(5)
20 DATA 8, -7, 5, -6, 4
30 FOR I=1 TO 5
40 READ A(I)
50 NEXT I
- c) 10 FOR I=1 TO 10
20 FOR K=1 TO 20
30 PRINT I+K,
40 NEXT K
- d) 10 FOR I=1 TO 10
20 FOR K=1 TO 20
30 PRINT I+K,
40 NEXT K
50 NEXT I
60 FOR I=1 TO 50
70 FOR I=1 TO 10
80 PRINT I/2
90 NEXT I
100 NEXT I

IF-THEN-ELSE și IF-GOTO-ELSE

În limbajul BASIC-PLUS instrucțiunile **IF-THEN-ELSE** și **IF-GOTO-ELSE** (v. formatul general al instrucțiunilor din limbajul BASIC-80) prezintă următoarele **particularități**:

- <relație> este de forma unei relații între două expresii (numerice sau șiruri) sau a unei reuniuni de relații prin operatorii logici: **AND, OR, NOT, XOR, IMP, EQV**;
- operatorii relaționali sint: "="; "<"; "<="; ">"; ">="; "≠" (diferit);
- <instrucțiune 1>/<instrucțiune 2> nu poate fi: **DIM, DEF, DATA, REM, FNEND, END, COM, FOR, NEXT**.

Aplicație. Introduceți și executați următoarele programe (unele incomplete!):

- a) 10 REM PROGRAMUL EFECTUEAZĂ
PRODUSUL
20 REM MAI MULTOR PERECHI DE
NUMERE (N)
30 PRINT "INTRODUCEȚI NUMĂRUL
DE PERECHI DE NUMERE";
40 INPUT N
50 FOR I=1 TO N
60 INPUT "A, B"; A, B
70 IF A/2=INT (A/2) .THEN GOTO ?
80 LET P=P+B
90 LET A= INT (A/2)
100 LET B=2*B
110 IF A>=1 THEN GOTO ?
120 PRINT "PRODUSUL ESTE:"; P
130 NEXT I
140 END
- 40 READ X(I)
50 NEXT I
60 DATA 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
70 S=0
80 FOR I=1 TO N
90 S=S+X(I)
100 NEXT I
110 PRINT "SUMA NUMERELOR DE LA
1 LA 10 ESTE"; S
120 END
- 5 REM 1×2×3×4×5×6×7×8×
×9×10
10 READ N
20 DIM X(N)
30 FOR I=1 TO N
40 READ X(I)
50 NEXT I
60 DATA 10
70 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
80 P=1
90 FOR I=1 TO N
100 IF X(I)=0 THEN PRINT "P=0" :
END
110 P=P*X(I)
120 NEXT I
130 PRINT "1×2×3×4×5×6×7×8×
×9×10="; P
- RUN
INTRODUCEȚI NUMĂRUL DE PERECHI
DE NUMERE ? 1
A, B ? 17, 12
PRODUSUL ESTE: 204
- b) 5 REM 1+2+3+4+5+6+7+8+9+
+10
10 READ N
20 DIM X(N)
30 FOR I=1 TO N

```

d) 10 INPUT "PRIMUL NUMĂR"; A
    20 INPUT "AL DOILEA NUMĂR"; B
    30 PRINT
    40 IF NOT (A>B) THEN PRINT
      "A=SAU<B": GOTO 80
    50 IF NOT (A=B) THEN PRINT
      "A(SAU)B": GO TO 80
    60 PRINT "A=SAU>B"
    70 PRINT A, B
    80 PRINT : PRINT
    90 PRINT "TASTAȚI O DACĂ
      NUMAI CONTINUAȚI"
    100 PRINT "TASTAȚI ORICE NUMĂR
      PENTRU CONTINUARE"
    110 INPUT N
    120 IF NOT (N=0) THEN 10
    130 PRINT
    140 END

e) 10 A0/0=120
    20 PRINT ">>>";
    30 A0/0=A0/0-1
    40 IF A0/0=20 THEN PRINT : PRINT
      TAB (10); "A0/0=20" : PRINT

50 IF A0/0>-1 GO TO 20
60 PRINT

f) 10 AREAL = .999999999
    20 LINIE0/0 = 1
    30 PRINT "NR. LINIE" , "REAL"
    40 PRINT
    50 PRINT LINIE0/0 , AREAL
    60 AREAL = AREAL * 10
    70 LINIE0/0 = LINIE0/0 + 1
    80 GO TO 50
    90 END

g) 10 DIM F(20)
    20 PRINT
    30 F(1)=1
    40 F(2)=1
    50 FOR I=3 TO 20
    60   F(I)=F(I-1)+F(I-2)
    70 NEXT I
    80 FOR I=1 TO 20
    90   PRINT F(I),
    100 NEXT I
    110 PRINT
    220 END

```

Înapoi la program

În cadrul programului controlul logic formal asupra datelor de intrare este asigurat prin intermediul instrucțiunilor **IF-THEN** (v. linia 170) și **GO TO** (v. linia 190). Mesajul "EROARE DE INTRODUCERE DATE ***** RAZA NEGATIVĂ **** RELUARE" oferă posibilitatea de a identifica valoarea negativă a razei și de a o corecta prin intermediul instrucțiunii **INPUT** (v. linia 140).

```

100 REM CITIREA DINAMICĂ ȘI VALIDARE DATE
110 DIM R(N)
120 FOR I=1 TO N
130   PRINT "REZERVORUL NUMĂRUL"; I
140   INPUT "RAZA REZERVORULUI"; R(I)
150   PRINT "AȚI SPECIFICAT"; R(I)
160   PRINT
170   IF R(I) > 0 THEN 200
180   PRINT "EROARE DE INTRODUCERE DATE * RAZA NEGATIVĂ * RELUARE"
190   GO TO 130
200 NEXT I

```

Cu valorile corecte ale razelor se calculează și însumează cantitatea de benzină din rezervoare. Conversația purtată de noi cu un minicalculator CORAL prin intermediul programului BASIC-PLUS este:

```

: RUN
  NUMĂR REZERVOARE ==?          6
  AȚI SPECIFICAT                6 REZERVOARE
  REZERVORUL NUMĂRUL            1
  RAZA REZERVORULUI             ?2
  AȚI SPECIFICAT                2
  REZERVORUL NUMĂRUL            2
  RAZA REZERVORULUI             ?-2.3
  AȚI SPECIFICAT                -2.3

```

EROARE DE INTRODUCERE DATE * * * * * RAZA NEGATIVĂ * * * * * RELUARE

REZERVORUL NUMĂRUL	2
RAZA REZERVORULUI	?2.3
AȚI SPECIFICAT	2.3
REZERVORUL NUMĂRUL	3
RAZA REZERVORULUI	?4
AȚI SPECIFICAT	4
REZERVORUL NUMĂRUL	4
RAZA REZERVORULUI	?6.2
AȚI SPECIFICAT	6.2
REZERVORUL NUMĂRUL	5
RAZA REZERVORULUI	?-3
AȚI SPECIFICAT	-3

EROARE DE INTRODUCERE DATE * * * * * RAZA NEGATIVĂ * * * * * RELUARE

REZERVORUL NUMĂRUL	5
RAZA REZERVORULUI	?3
AȚI SPECIFICAT	3
REZERVORUL NUMĂRUL	6
RAZA REZERVORULUI	?10.0
AȚI SPECIFICAT	10

CALCULUL MASEI TOTALE DE BENZINA
DIN 6 REZERVOARE CILINDRICE ECHILATERALE

RAZA	MASA
2	35.1858
2.3	53.5133
4	281.487
6.2	1048.22
3	118.752
10	4398.23

MASA TOTALĂ = 5 935.39 TONE

STOP AT LINE 440

Aplicație. Să se scrie un program BASIC care calculează $\cos(x)$ cu relația:

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots + (-1)^{n+1} \cdot \frac{x^{2n-2}}{(2n-2)!}$$

10 PRINT "X=";	90 S=S+((-1)↑(N+1) * X↑(2 * N-2)/F)
20 INPUT X	100 IF ABS(S-S1)/(ABS(S)+ABS(S1)) > 1E-6 THEN 40
30 S1=S=N=F=1	110 PRINT "COSINUS CALCULAT CU COS (X)" ; COS (X)
40 PRINT N ; S	120 END
50 N=N+1	
60 S1=S	
70	
80 F=F * (2 * N-2) * (2 * N-3)	

Particularități ale programării în limbajul ABASIC

vol. 2, pag. 215

Variabile indexate numerice. Reguli

În limbajul ABASIC variabilele indexate prezintă următoarele **caracteristici**:

- numele unei variabile indexate este alcătuit dintr-un nume admis de variabila simplă urmat de una, două sau mai multe expresii de indici cuprinse între paranteze;
- indicii pot fi nume de variabile, expresii, funcții care prin evaluare trebuie să dea o valoare întregă pozitivă;
- variabilele indexate pot fi de două tipuri: întreg, real; tipul întreg se indică prin caracterul "0/" plasat după numele simbolic al variabilei;
- într-un program pot exista variabile simple și variabile indexate cu același nume;
- valoarea minimă a unui indice este zero sau unu (v. opțiunea **BASE**);

În cadrul programului variabila indexată numerică (cu un indice) este R(I).

DIM

În limbajul ABASIC instrucțiunea **DIM** prezintă următoarele **particularități**:

- numărul de dimensiuni ale tablourilor declarate este limitat numai de lungimea liniei ABASIC (80 de caractere);
- pentru a preciza rangul primului element al unui tablou (0 sau 1) utilizatorul dispune de instrucțiunea **OPTION BASE** al cărei format este:

<hr style="border: 1px solid black;"/> Format general <hr style="border: 1px solid black;"/>
<hr style="border: 1px solid black;"/> OPTION BASE (expresie) <hr style="border: 1px solid black;"/>

în care (expresie) poate lua valorile 0 sau 1.

IF

În limbajul ABASIC instrucțiunea **IF** prezintă trei formate generale.

Formatul – 1: IF–THEN–ELSE

Instrucțiunea **IF–THEN–ELSE** prezintă același format ca și instrucțiunea omologă din limbajul BASIC-80, cu următoarele **particularități**:

- operatorii relaționali admiși sint:
"="; "<"; ">"; "<="; ">="; "<>" și "==" (aproximativ egal);
- operatorii logici admiși sint: **NOT, AND, OR, XOR, EQV, IMP.**

Observație. Pentru variabilele numerice, operatorul "==" are semnificația unei egalități aproximative, cu o diferență mai mică de 10^{-8} .

- instrucțiunea (**IF–THEN–ELSE**) trebuie să fie conținută pe o singură linie program;
- dacă o instrucțiune **IF** este urmată pe aceeași linie de alte instrucțiuni **IF** atunci fiecare **ELSE** corespunde **IF**-ului cel mai apropiat;
- (instrucțiune 1)/(instrucțiune 2) poate fi orice tip de instrucțiune care admite scrierea pe o singură linie program.

Aplicație. Scrieți un program BASIC care generează și tipărește toate punctele pitagorice cu valori ale ipotenuzei mai mici ca 101.

```

10 FOR I=3 TO 98
20   FOR J=I+1 TO 99
30     K1=SQR (I * I+J * J)
40     IF K1 <> INT (K1) THEN 70
50     IF K1 > 100 THEN 80
60     PRINT I; J; K1
70   NEXT J
80 NEXT I
90 END
RUN
```

Formatul – 2: IF–GOTO

Instrucțiunea **IF–GO TO** are formatul:

Format general	
(nr. linie) IF (relație) GOTO (nr. linie)	

în care (relație) și (nr. linie) au semnificațiile cunoscute.

Observație. Instrucțiunea trebuie să fie conținută pe o singură linie program.

Formatul – 3: IF–THEN–ELSE–ENDIF

Instrucțiunea are formatul:

Format general	
(nr. linie) IF (relație) [THEN]	(bloc – instrucțiuni 1)
[ELSE	(bloc – instrucțiuni 2)]
ENDIF	

în care (bloc instrucțiuni) reprezintă o secvență de instrucțiuni ABASIC, scrise pe una sau mai multe linii program.

Remarcă. Instrucțiunea poate ocupa mai multe linii terminal având în vedere restricțiile de mai jos:

- cuvântul cheie **ELSE** (dacă este folosit) trebuie să fie singur pe linia program;
- cuvântul cheie **ENDIF** trebuie să fie singur pe linia program;
- **IF** trebuie să fie ultima instrucțiune în linie.

Aplicație. Să se verifice dacă punctele $A(x_1, y_1)$, $B(x_2, y_2)$ și $C(x_3, y_3)$ sunt coliniare. Programul va afișa unul din mesajele: "PUNCTE COLINIARE"; "PUNCTE NECOLINIARE"; "PUNCTE VERTICAL COLINIARE".

Specificațiile de programare și documentația de proiectare sint prezentate în modulele de analiză și proiectare structurată, fig. 5.12.

TABELA DE VARIABLE

Variabile de intrare	Variabile de stare	Variabile de ieșire
X1,Y1: coordonatele punctului A	D1: diferența absciselor punctelor A și B	M1: panta dreptei determinate de punctele A și B
X2,Y2: coordonatele punctului B	D2: diferența ordonatelor punctelor B și C	M2: panta dreptei determinate de punctele B și C
X3,Y3: coordonatele punctului C		

a)

DATELE DE INTRARE

X1	Y1	X2	Y2	X3	Y3
6	5	0	7	-9	10
1	2	3	4	5	7
1	2	1	4	2	3
.5	2	5	-3	5	50
999	0				

b)

Fig. 5.12. Modulele de analiză și proiectare structurată: a) tabela de variabile; b) datele de intrare;

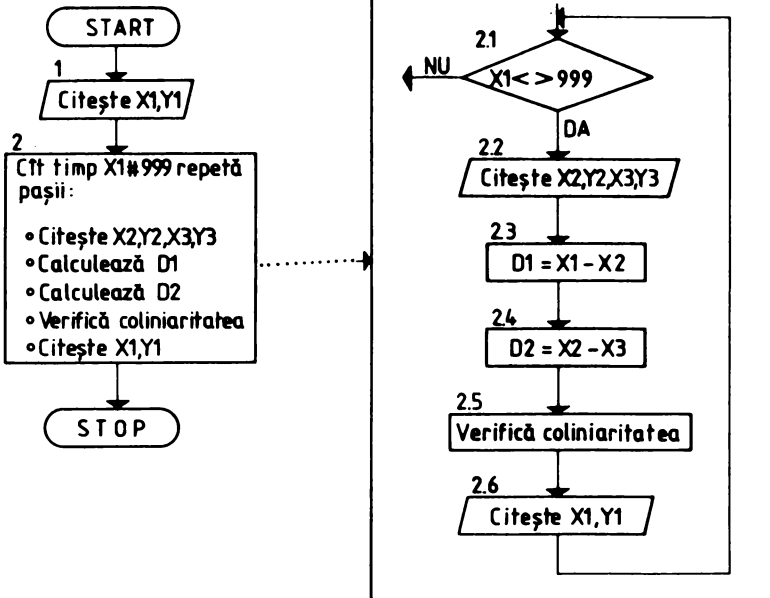
PSEUDOCODUL

```

APLIC      SEQ
              READ X1, Y1
WHILE      CIT TIMP X1 (< > 999
W          SEQ
              READ X2, Y2, X3, Y3
              PRINT X1, Y1, X2, Y2, X3, Y3
              D1=X1-X2
              D2=X2-X3

W          END
D          DACA D1 * D2 < > 0
              M1=(Y1-Y2) / D1
              M2=(Y2-Y3) / D2
D1         DACA M1 < > M2
              PRINT ""PUNCTE NECOLINAIRE"
D1         IN CAZ CONTRAR
              PRINT "PUNCTE NECOLINIARE"
D1         SFIRȘIT
D          IN CAZ CONTRAR
D2         DACA D1 < > 0
              PRINT "PUNCTE NECOLINIARE"
D2         IN CAZ CONTRAR
D3         DACA D2 < > 0
              PRINT "PUNCTE NECOLINIARE"
D3         IN CAZ CONTRAR
              PRINT "PUNCTE VERTICAL COLINIARE"
D3         SFIRȘIT
D2         SFIRȘIT
D          SFIRȘIT
WHILE      READ X1, Y1
APLIC      SFIRȘIT
              END
    
```

c)



d)

c) pseudocodul; d) schema logică.

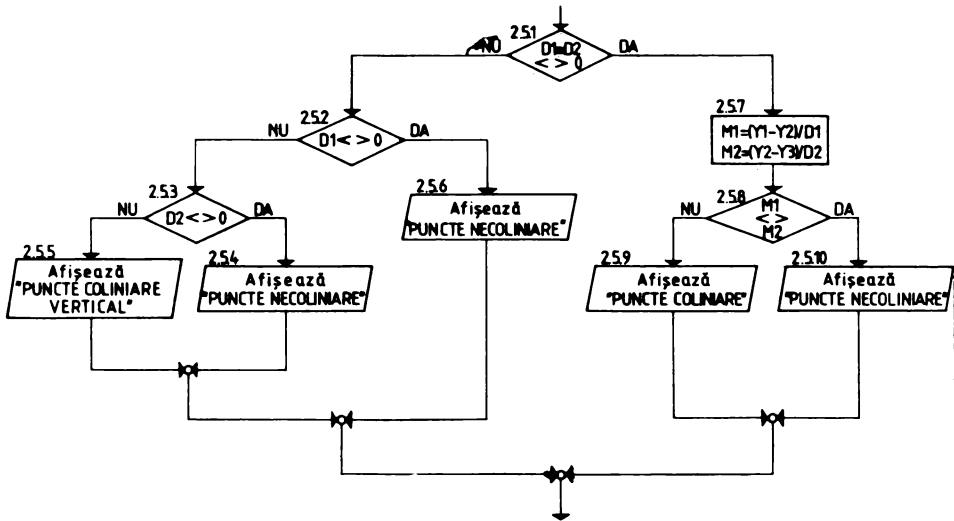


Fig. 5.12. d) schemă logică.

WHILE-ENDWHILE

WHILE-ENDWHILE creează un ciclu compus din mai multe instrucțiuni ABASIC atâta timp cât o condiție dată este adevărată. Formatul general al instrucțiunii este:

Format general
WHILE < relație >
< secvență instrucțiuni >
ENDWHILE

în care <secvență-instrucțiuni> reprezintă orice tip de instrucțiune ABASIC.

Observație. În loc de **ENDWHILE** se poate folosi și **WEND** (v. remarcile de la limbajele BASIC-AMSTRAD și BASIC-80). Cît privește programul ABASIC el este identic cu cel realizat în BASIC-80, pentru calculatoarele M 118, TPD, JUNIOR.

Aplicație. Să se scrie un program BASIC care determină punctul de intersecție a două drepte de ecuație $y=mx+n$.

```

10 READ M1, N1
20 WHILE M1 < > 999
30 PRINT "Y=( "; M1; " ) * X+( " ; N1 ; " )"
40 READ M2, N2
50 PRINT "Y=( "; M2; " ) * X+( " ; N2 ; " )"
60 IF M1 < > M2 THEN
70 X=(N2-N1) / (M1-M2)
80 Y=M1 * X+N1
90 PRINT "PUNCTUL DE INTERSECȚIE" ; "( " ; X ; " , " ; Y ; " )"
100 PRINT
110 ELSE
120 PRINT "DREPTE PARALELE"
130 PRINT
140 ENDIF
  
```

```

150 READ M1, N1
160 WEND
170 DATA 3, -7, -4, 14, 1, 2, 3, 4,
180 DATA 5, 3, 5, 8, 2, -11, 5, -11
190 DATA 999,0
200 END

```

Observație. Variabilele de intrare M1, N1, M2, N2 reprezintă panta și termenul liber pentru fiecare ecuație în parte. X, Y (variabilele de ieșire) reprezintă coordonatele punctului de intersecție.

TEMA 5

Răspundeți prin DA sau NU la următoarele întrebări:

- O variabilă indexată se declară folosind instrucțiunea **DIM**.
- Într-o instrucțiune **DIM** se specifică între paranteze numărul de elemente din tablou.
- În BASIC-aMIC numărul de indici ai unei variabile indexate este 1 sau 2 (vector sau matrice).
- În BASIC-PRAE numărul de indici ai unei variabile indexate este 1 la 255.
- În aceeași instrucțiune **DIM** tablourile numerice și cele șir pot fi declarate intercalat.
- Indicele (indexul) unei variabile indexate poate fi o constantă, o variabilă numerică simplă (de tip întreg) sau expresii aritmetice.
- Tablourile numerice pot avea una, două sau mai multe dimensiuni.
- Pentru introducerea datelor în mod static se utilizează instrucțiunile **READ** și **DATA**.
- Instrucțiunea **DATA** poate apare oriunde în cadrul programului.
- Instrucțiunea **READ** are același efect cu **INPUT**, cu deosebirea că datele nu sînt introduse de la tastatură, ci sînt citite dintr-un bloc de date, definit cu instrucțiunea **DATA**.
- Condiția ce apare între cuvintele cheie **IF** și **THEN** este de forma unei relații între două expresii numerice sau șiruri.
- Limbajul BASIC-aMIC nu este un limbaj structurat.
- Instrucțiunea **IF–THEN–ELSE** codifică structura de selecție.
- Instrucțiunea


```

10 IF A<5 THEN 70

```

 are ca efect saltul la linia 70 atunci cînd condiția este adevărată.
- Instrucțiunea


```

10 IF A=3 THEN 20 ELSE 80

```

 are ca efect saltul la linia 20 atunci cînd condiția este adevărată și saltul la linia 80 atunci cînd condiția nu este adevărată.
- În limbajul BASIC-aMIC într-o instrucțiune **IF–THEN** se pot defini condiții compuse utilizînd operatorii logici binari **AND**, **OR**, **NOT**.

- Dacă două relații sînt reunite prin operatorul logic **AND** perechea relațiilor este adevărată numai dacă ambele relații sînt adevărate.
- Dacă două relații sînt reunite prin operatorul logic **OR** perechea relațiilor este adevărată dacă cel puțin una dintre relații este adevărată.
- În BASIC-PRAE pot fi declarate tablouri cu dimensiuni variabile.
- În BASIC HC-85, TIM S, SPECTRUM numele unei variabile indexate este alcătuit dintr-o singură literă.
- În BASIC HC-85, TIM S, SPECTRUM variabilele șir și variabilele tablou-șir nu pot avea același nume.
- În BASIC-TIM S o variabilă numerică simplă și o variabilă indexată pot avea același nume.
- În BASIC-SPECTRUM cu o instrucțiune **DIM** poate fi definită numai o singură variabilă indexată.
- În BASIC HC-85, TIM S, SPECTRUM operatorii logici (**AND, OR, NOT**) pot fi considerați și folosiți ca funcții.
- În BASIC-AMSTRAD valoarea minimă a unui indice este zero.
- În BASIC-aMIC variabilele indexate pot fi de tip întreg și real.
- În BASIC-PRAE, dacă dimensiunea (**DIM**) unei variabile indexate nu este specificată, ea se consideră implicit ca fiind 10.
- În limbajul BASIC-AMSTRAD există instrucțiunile **WHILE-WEND**.
- **WHILE-WEND** implementează structura de iterație **CIT-TIMP**.
- În BASIC-COMMODORE variabilele indexate pot fi de două tipuri: întreg și real.
- În BASIC-80 variabilele tablou ale unui program pot fi eliminate cu instrucțiunea **ERASE**.
- În limbajul BASIC-PLUS se poate utiliza instrucțiunea **IF-GOTO-ELSE**.
- În limbajul ABASIC instrucțiunea **IF** prezintă trei formate generale.
- În limbajul BASIC-80 în loc de **WEND** se poate folosi și **ENDWHILE**.

Înlocuiți cuvintele care lipsesc din următoarele propoziții:

- a) Un vector este
- b) Structura **CIT TIMP** se mai numește și deoarece
Structura mai este cunoscută și sub denumirea de „buclă de tip **DO**
- c) Variabilele **AGA (2)**, **B (7)** sînt variabile iar **AGA (2)** și **B (7)** sînt variabile
- d) Într-un program BASIC, pot exista variabile și variabile indexate cu același nume.
- e) Pentru introducerea dinamică a vectorilor de date se folosește instrucțiunea plasată într-o buclă **FOR-NEXT**.
- f) Pentru introducerea statică a vectorilor de date se folosesc instrucțiunile, plasate într-o buclă **FOR-NEXT**.
- g) Instrucțiunea are același efect cu **INPUT**, cu deosebirea că datele nu sînt introduse de la tastatură, ci sînt citite dintr-un definit cu instrucțiunea
- h) În BASIC structurat buclă de tip se codifică cu instrucțiunile **WHILE-WEND**, unde **WHILE** indică iar **WEND**

- i) Pentru simularea iterației **CIT TIMP** în limbajele BASIC , se aplică următoarea metodologie:
- j) Instrucțiunea **IF-THEN** este o instrucțiune de salt iar instrucțiunea definește un salt necondiționat.
- k) Structura de selecție poate fi cu sau cu alternative.
- l) Structura **IF-THEN-ELSE** este specifică limbajului BASIC implementat pe calculatoarele iar **IF-GOTO-ELSE**
- m) Instrucțiunea **IF-THEN** (nr. linie) este specifică limbajului BASIC implementat pe calculatoarele iar **IF-THEN** (nr. linie/instrucțiune)
- n) Operatorii logici **AND, NOT, OR** sînt admiși de limbajul BASIC implementat pe calculatoarele
- o) Operatorii logici **EQV** și **IMP** sînt admiși de limbajul BASIC implementat pe calculatoarele
- p) Operatorul relațional "**==**" (aproximativ egal) este admis de limbajul BASIC implementat pe calculatoarele
- r) Instrucțiunea **IF-THEN-ELSE-ENDIF** este specifică limbajului BASIC implementat pe calculatoarele
- s) În limbajul BASIC implementat pe calculatoarele structura de iterație se codifică cu instrucțiunile **WHILE-WEND**.

Să se scrie cîte un program BASIC pentru fiecare din problemele de mai jos:

- a) Să se efectueze împărțirea:

$$\frac{x^4 + 2x^3 - 13x^2 - 14x + 24}{x - 1}$$

- b) Ca urmare a aplicării metodei de determinare a diametrelor medii aritmetice, geometric și armonic unui grup de particule (395) măsurate cu micrometrul optico-electronic într-un eșantion de lichid (folosit la o instalație de acționare de la o mașină-unealtă cu comandă numerică) s-au obținut următoarele rezultate:

Numărul intervalului	Intervalul J	Diametrul, d_{nj} mediu al intervalului	Nr. de particule n_j din intervalul j
1	0- 4,9	2,5	10
2	5- 9,9	7,5	15
3	10-14,9	12,5	52
4	15-19,9	17,5	85
5	20-24,9	22,5	81
6	25-29,9	27,5	51
7	30-34,9	32,5	36
8	35-39,9	37,5	27
9	40-44,9	42,5	17
10	45-49,9	47,5	10
11	50-54,9	52,5	7
12	55-59,9	57,5	4

Să se calculeze:

- diametrul mediu aritmetic, $d_{n,a}$;
- diametrul mediu geometric, $d_{n,g}$;
- diametrul mediu armonic, $d_{n,ar}$.

Să se verifice relația:

$$d_{n,ar} < d_{n,g} < d_{n,a}$$

În caz de eroare, se va afișa mesajul "Relația $d_{n,ar} < d_{n,g} < d_{n,a}$ NU ESTE ÎNDEPLINĂ".

Indicație. Pentru determinarea celor trei diametre se vor utiliza [] următoarele relații de calcul:

$$d_{n,a} = \frac{n_1 d_{n1} + n_2 d_{n2} + \dots + n_r d_{nr}}{n_1 + n_2 + \dots + n_r} = \frac{\sum_{j=1}^r (n_j d_{nj})}{\sum_{j=1}^r n_j}$$

$$d_{n,g} = \sqrt[r]{d_{n1} \cdot d_{n2} \cdot \dots \cdot d_{nr}} = \sqrt[r]{\prod_{j=1}^r d_{nj}}$$

$$d_{n,ar} = \sum_{j=1}^r n_j \sum_{j=1}^r \left(\frac{d_{nj}}{n_j} \right)$$

Observație. În ultimele trei relații j reprezintă numărul curent al intervalului în care au fost grupate măsurătorile, în cazul nostru, $r=12$ intervale.

- c) Să se verifice dacă triunghiul cu virfurile în punctele $A(x_1, y_1)$, $B(x_2, y_2)$, $C(x_3, y_3)$ este dreptunghic.
- d) Să se arate că patrulaterul cu virfurile în

$$M(1,5), N\left(\frac{2}{5}, \frac{19}{5}\right), P(2, 3), Q\left(\frac{13}{5}, \frac{21}{5}\right)$$

este dreptunghi.

- e) Să se verifice coliniaritatea a n puncte.
- f) Să se verifice dacă punctele $M(1,1)$, $N(-5,5)$ și $P(0, \frac{5}{3})$ sînt coliniare.
- g) Să se determine punctul de intersecție a două drepte de ecuații:

$$a_1x + b_1y + c_1 = 0$$

și

$$a_2x + b_2y + c_2 = 0$$

- h) Să se determine punctul de intersecție a două mediane ale unui triunghi.
- i) Să se scrie ecuația perpendicularei care trece prin mijlocul dreptei determinate de punctele A și B de coordonate x_1, y_1 respectiv x_2, y_2 .

- j) Să se scrie ecuația dreptei care trece prin punctul $M(x_0, y_0)$ și este perpendiculară pe dreapta determinată de punctele A și B de coordonate x_1, y_1 respectiv x_2, y_2 .
- k) Să se determine distanța de la un punct la o dreaptă de ecuație $ax + by + c = 0$.
- l) Să se scrie ecuația dreptei determinate de punctele $A(x_1, y_1)$ și $B(x_2, y_2)$. Discuție.
- m) Să se scrie ecuația dreptei care conține punctul $M(-1, 3)$ și care este perpendiculară pe dreapta $3x - 5y = 0$.
- n) Fie punctele $P(2, 81)$, $M(0, -79)$, $N(-159, 3)$, $Q(161, -1)$. Să se arate că dreptele PM și NQ sînt perpendiculare.
- o) Să se verifice dacă triunghiul cu vîrfurile în punctele $(1, 2)$, $(5, -1)$, $(6, 15)$ și $(0, 0)$, $(0, 3)$, $(4, 0)$ este dreptunghic.

Și în cadrul acestei teme se reia problema din conversația precedentă, complicînd-o sub următoarele aspecte:

- se citește de la tastatură numărul de articole (maximum 5);
- se memorează într-un vector, printr-o procedură de introducere statică a datelor, cantitățile lunare de jucării fabricate;
- pentru fiecare articol (tip jucărie) se calculează procentul cantității totale trimestriale față de cantitatea totală anuală;
- pentru fiecare articol se afișează: număr trimestru (I, II, III, IV), cantitatea totală trimestrială, procentul cantității trimestriale față de cantitatea anuală.

Se reia problema din conversația precedentă, complicînd-o după cum urmează:

Pentru un număr variabil de articole se citesc de la tastatură consumul normal și consumul realizat. Se calculează pentru fiecare articol abaterea în procente față de consumul normal și se tipăresc: codul articolului (1, 2, ..., m), proveniența (R.S.R. sau import) și abaterea în procente. Locul de fabricație a articolului se citește sub forma codificată: 1 pentru „import” și 2 pentru „R.S.R.”, dar se va edita în clar (R.S.R. sau IMPORT).

SOLUȚIA TEMEI 5

Nu răspundem.

Nu răspundem.

Programele BASIC sînt:

```
a) 10 DIM P(15), Q(15)
    20 READ N
    30 PRINT "P(X)=";
```

```
40 FOR I=N+1 TO 1 STEP -1
    50 READ P(I)
    60 PRINT P(I);
```

```

70 NEXT I
80 REM -----
90 PRINT
100 PRINT "IMPĂRȚIT la X-";
110 INPUT X0
120 PRINT "CITUL";
130 P3=Q(N+1)=P(N+1)
140 FOR I=N TO 1 STEP-1
150 Q(I)=P3=P3*X0+P(I)
160 NEXT I
170 REM -----
180 REM AFIȘARE REZULTAT
190 FOR I=N+1 TO 2 STEP-1
200 PRINT Q(I)
210 NEXT I
220 REM -----
230 PRINT "RESTUL="; Q(1)
240 DATA 4
250 DATA 1, 2, -13, -14, 24.
260 END.
RUN
P(X)=1 2 -13 -14 24
IMPĂRȚIT LA X-?+1
CITUL 1 3-10-24 RESTUL=0

```

```

c) 10 T=0
20 WHILE T < > 9999
30 FOR I=1 TO 3
40 READ X(I), Y(I)
50 PRINT "(": X(I); ", ";
Y(I); ");";
60 NEXT I
70 FOR I=1 TO 3
80 X(I+3)=X(I)
90 Y(I+3)=Y(I)
100 NEXT I
110 FOR I=1 TO 3
120 A=X(I)-X(I+1)
130 B=Y(I)-Y(I+1)
140 D(I)=SQR (A*A+B*B)
150 D(I+3)=D(I)
160 NEXT I
170 FOR I=1 TO 3
180 R=D(I) ↑ 2
190 S=D(I+1) ↑ 2+D(I+
+2) ↑ 2
200 IF R=S THEN 240
210 NEXT I
220 PRINT "TRIUNGHIIUL NU
ESTE DREPTUNGHIC"
230 GOTO 260
240 PRINT "TRIUNGHIIUL ESTE
DREPTUNGHIC"
250 PRINT "IPOTENUZA ESTE
DETERMINATĂ DE";
X (I+1); Y(I+1)

```

```

260 READ T
270 WEND
280 DATA 1, 2, 5, -1, 6, 15, 9
290 DATA 0, 0, 0, 3, 4, 0, 9 9 9 9
300 END

```

```

e) 10 INPUT "N="; N
20 DIM X(N), Y(N)
30 FOR I=1 TO N
40 PRINT "X("; I; ")=";
50 INPUT X(I)
60 PRINT "Y("; I; ")=";
70 INPUT Y(I)
80 NEXT I
90 M1=(Y(2)-Y(1))/(X(2)-X(1))
100 FOR I=2 TO N-1
110 M2=(Y(I+1)-Y(I))/(X(I+
+1)-X(I))
120 IF M1 < > M2 THEN 170
130 M1=M2
140 NEXT I
150 PRINT N; "PUNCTE COLINIARE"
160 STOP
170 PRINT "PUNCTE NECOLINIARE"
180 END

```

```

RUN
N=?5
X(1)=?1
X(2)=?3
X(3)=?5
X(4)=?7
X(5)=?9
Y(1)=?2
Y(2)=?4
Y(3)=?6
Y(4)=?8
Y(5)=?10

```

5 PUNCTE COLINIARE

```

RUN
N=?4
X(1)=?1
X(2)=?3
X(3)=?2
X(4)=?3
Y(1)=?8
Y(2)=?9
Y(3)=?7
Y(4)=?2
PUNCTE NECOLINIARE

```

g) Fiind date două drepte de ecuație:

$$a_1x + b_1y + c_1 = 0$$

și

$$a_2x + b_2y + c_2 = 0$$

în urma rezolvării sistemului de ecuații rezultă coordonatele (x, y) punctului de intersecție.

$$x = \frac{b_1 \cdot c_2 - b_2 \cdot c_1}{b_2 \cdot a_1 - b_1 \cdot a_2},$$

și

$$y = \frac{a_2 \cdot c_1 - a_1 \cdot c_2}{a_1 \cdot b_2 - a_2 \cdot b_1}.$$

Atunci cînd $b_2 \cdot a_1 - b_1 \cdot a_2 = 0$, rezultă:

$$\frac{a_1}{b_1} = \frac{a_2}{b_2}, \text{ sau}$$

$-\frac{a_1}{b_1} = -\frac{a_2}{b_2}$; ce reprezintă condiția de paralelism a două drepte ($y = -$

$\frac{a_1}{b_1}x - \frac{c_1}{b_1}$ și $y = -\frac{a_2}{b_2}x - \frac{c_2}{b_2}$; cele două drepte au pantele $-\frac{a_1}{b_1}$ și $-\frac{a_2}{b_2}$ egale).

Cînd b_1 sau b_2 sînt zero, rezultă

$$a_1x + c_1 = 0, x = -\frac{c_1}{a_1},$$

și

$$a_2x + c_2 = 0, x = -\frac{c_2}{a_2}.$$

```

10 READ A1, B1, C1
20 WHILE A1 < > 999
30 READ A2, B2, C2
40 PRINT A1; "*X+("; B1; ")
   *Y+("; C1; ")=0"
50 PRINT A2; "*X+("; B2; ")
   *Y+("; C2; ")=0"
60 D=A1*B2-A2*B1
70 IF D < > 0 THEN
80 N1=A2*C1-A1*C2
90 N2=B1*C2-B2*C1
100 X=N2/D
110 Y=N1/D
120 PRINT "SOLUȚIA ESTE:
   ("; X; ", "; Y; ")"
130 ELSE
140 IF B1=0 THEN
150 IF C1/A1=C2/A2 THEN
160 PRINT "DREPTLE
   COINCID"
170 ELSE
180 PRINT "DREPTLE
   PARALELE"
190 ENDIF
200 ELSE
210 IF C1/B1=C2/B2 THEN
220 PRINT "DREPTLE
   COINCID"
230 ELSE
240 PRINT "DREPTLE
   PARALELE"
250 ENDIF
260 ENDIF
270 ENDIF
280 READ A1, B1, C1
290 WEND
300 DATA 1, 1, 1, 1, 2, 2, 2
310 DATA 1, 1, 2, 3, 4, 5, 6
320 DATA 1, 5, 3, 10, 10, 6, -3
330 DATA 1, 3, 0, 4, 7, 0, -8
340 DATA 999, 0, 0
350 END

```

i) 10 READ X1, Y1
20 WHILE X1 < > 999
30 READ X2, Y2
40 PRINT X1; Y1; X2; Y2
50 XO=(X1+X2)/2
60 YO=(Y1+Y2)/2
70 IF (X1-X2) < > 0
80 IF (Y1-Y2) < > 0 THEN
90 M=(Y2-Y1)/(X2-X1)
100 MO=-1/M
110 N=YO-MO * XO
120 PRINT MO, N
130 ELSE

```

140     PRINT "X="; XO
150     ENDIF
160     ELSE
170     IF (Y1-Y2)<>0 THEN
180         PRINT "Y="; YO
190     ELSE
200         PRINT "PUNCTELE
        COINCID"
210     ENDIF
220     ENDIF
230     READ X1, Y1
235     WEND
240     DATA 1, 2
250     DATA 4, 7, 4, 7, 4, 3
260     DATA -3, 9, 4, 5, 4, -7, 4,
        -7, 999, 0
270     END

j) 10 INPUT "XO, YO"; XO, YO
    30 WHILE XO<>999
    35     INPUT "X1, Y1"; X1, Y1
    40     INPUT "X2, Y2"; X2, Y2
    50     V=Y2-Y1
    60     H=X2-X1
    70     IF V<>0 THEN
    80         IF H<>0 THEN
    90             M=V/H
    100            MO=-1/M
    110            N=YO-MO * XO
    120            PRINT MO, N
    130        ELSE
    140            PRINT "Y=", YO
    150        ENDIF
    160    ELSE
    170        IF H<>0 THEN
    180            PRINT "X=", XO
    190        ELSE
    200            PRINT "PUNCTELE
            COINCID"
    210        ENDIF
    220    ENDIF
    230 INPUT "XO, YO"; XO, YO
    240 WEND
    250 END

```

□ Pentru scrierea programului s-au definit variabilele de intrare – Z (număr articole), C (cantitățile lunare de jucării pentru un anumit tip de articol); variabilele de stare – S (cantitate totală anuală de jucării), L, I, K, R (variabile de lucru), C1 (cîțul împărțirii), R (restul împărțirii), T (cantitate totală trimestrială); variabila de ieșire – P (consumuri procentuale trimestriale).

```

5 REM PROGRAMUL CITEȘTE CANTITĂȚILE LUNARE (C) DE JUCĂRII PENTRU
10 REM UN NUMĂR (Z) VARIABIL DE ARTICOLE, CALCULEAZĂ
15 REM PROCENTUL (P) CANTITĂȚII TOTALE TRIMESTRIALE
20 REM FAȚA DE CANTITATEA TOTALĂ ANUALĂ (S) ȘI
25 REM TIPĂREȘTE REZULTATUL
30 PRINT "INDICAȚI NUMĂRUL DE ARTICOLE";
40 INPUT Z
50 DIM C(12), P(4)
60 PRINT "TRIMESTRUL 1 TRIMESTRUL 2 TRIMESTRUL 3 TRIMESTRUL 4"
70 PRINT
75 FOR K=1 TO Z
80     FOR I=1 TO 12
90         READ C (I)
100        NEXT I
110        DATA 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120
120        DATA 11, 21, 31, 41, 51, 61, 71, 81, 91, 101, 111, 121
130        DATA 12, 22, 32, 42, 52, 62, 72, 82, 92, 103, 112, 122
140        DATA 13, 23, 33, 43, 53, 63, 73, 83, 93, 102, 113, 123
150        DATA 14, 24, 34, 44, 54, 64, 74, 84, 94, 104, 114, 124
160        REM CALCUL CANTITATE TOTALĂ ANUALĂ JUCĂRII
170        S=0
180        FOR I=1 TO 12
190            S=S+C(I)
200        NEXT I
210        REM CALCUL PROCENTE
220        FOR L=1 TO 12
230            C1=INT(L/3)*3
240            R=L-C1
250            IF R<>1 THEN 270
260            T=0
270            T=T+C(L)
280            IF R<>0 THEN 300

```

```

290 P (C1/3)=T*100/S
300 NEXT L
310 REM EDITARE REZULTATE
320 FOR R=1 TO 4
330 PRINT P(R),
340 NEXT R
345 PRINT
350 NEXT K

```

Remarci

- Introduceți și executați programul aplicației.
- În momentul cînd noi am introdus pentru prima dată programul, am tastat linia 290, de maniera

```
290 P(C1)=T*100/S
```

Analizați singuri implicațiile datorate tastării greșite a acestei instrucțiuni, simulînd cu creionul și hîrtia modul de funcționare a procedurii de calcul procente din liniile 210–300. Atenție la valorile lui C1 și P!

Instrucțiunea (corectă)

```
290 P(C1/3)=T*100/S
```

mai poate fi scrisă și în alt mod? Poate fi înlocuită cu instrucțiunea

```
290 P(L/3)=T*100/S
```

Nu vă grăbiți cu răspunsul. Dacă aveți dificultăți apelați la prietenul dvs. calculatorul.

Observație. Rețineți procedura de calcul "procente.

```

210 REM CALCUL PROCENTE
220 FOR L=1 TO 12
230 C1=INT (L/3)*3
240 R=L-C1
250 IF R<>1 THEN 270
260 T=0
270 T=T+C(L)
280 IF R<>0 THEN 300
290 P (C1/3)=T*100/S
300 NEXT L

```

Pentru scrierea programului s-au definit variabilele de intrare – M (numărul de articole), N (consumul normal), F (codul de fabricație), R (consumul realizat) și variabila de ieșire A (abaterea în procente față de consumul normal).

```

5 REM PROGRAMUL CITEȘTE CONSUMUL NORMAL (N)
10 REM ȘI CONSUMUL REALIZAT (R) PENTRU UN
15 REM NUMĂR VARIABIL DE ARTICOLE (M)
20 REM PROGRAMUL CALCULEAZĂ ABATEREA
25 REM ÎN PROCENTE (A) FAȚĂ DE CONSUMUL NORMAL
50 REM *****
60 INPUT "NUMĂR ARTICOLE"; M
70 FOR L=1 TO M
80 PRINT "CONSUMUL NORMAL PENTRU ARTICOLUL:"; L;
90 INPUT N
100 PRINT
110 PRINT "CODUL DE FABRICAȚIE 1 (IMPORT)/2(R.S.R.)";
120 INPUT F
130 PRINT
140 PRINT "CONSUMUL REALIZAT";
150 INPUT R
160 PRINT
170 A=(R-N)*100/N

```

```
180 IF F=1 THEN 200
190 PRINT "ARTICOLUL", I, "R.S.R.", "ABATEREA=", A, "%"
195 GOTO 210
200 PRINT "ARTICOLUL"; I, "IMPORT", "ABATEREA="; A; "%"
210 NEXT I
220 END
```

RUN

```
NUMAR ARTICOLE ? 3
CONSUMUL NORMAL PENTRU ARTICOLUL : 1 ? 30
CODUL DE FABRICAȚIE 1 (IMPORT)/2 (R.S.R.) ? 1
CONSUMUL REALIZAT ? 15
ARTICOLUL 1 IMPORT ABATEREA=-50%
```

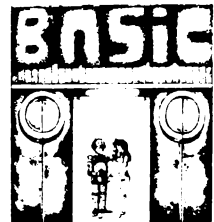
```
.
.
.
```

Remarci

- Introduceți și executați programul aplicației.
- În momentul introducerii programului nu uitați să tastați și linia 195. În absența liniei 195 s-ar fi întâmplat ceva grav?
- Rescrieți secvența de instrucțiuni din liniile 190–200 utilizând facilitățile limbajului GW-BASIC.

CONVERSAȚIA 6

Proiectarea și realizarea unui program pentru editarea raportului săptăminal privind livrările zilnice de lichide. Variabile indexate alfanumerice. Instrucțiuni matriciale. Funcțiile DEF FN și RANDOMIZE. Instrucțiunea PRINT AT. Programe utile pentru matematică și fizică. JOCURI, APLICAȚII și TESTE pentru cititor



EXEMPLELE 6 – PC, m, M, F

□ De la problemă la program

Vom aborda și în cadrul acestei conversații aceeași problemă, complicînd-o după cum urmează.

Se citesc de la tastatură livrările de benzină efectuate zilnic (luni, marți, miercuri, joi, vineri, sîmbătă) din trei rezervoare (cilindrice echilaterale) R1, R2, R3. Se dorește ca, pentru fiecare rezervor în parte, să se afișeze, pe zile, cantitățile de benzină livrate beneficiarilor. Se va tipări, de asemenea, media livrărilor pe zile și pe rezervoare. La finele listei se va afișa maximul livrărilor, precizîndu-se atît rezervorul cît și ziua respectivă.

□ Analiza problemei

Formatul datelor de ieșire, datele de intrare, tabela de variabile, specificațiile de programare și alocarea funcțiilor de prelucrare sînt ilustrate în modulul de analiză structurată, fig. 6.1.

FORMATUL DATELOR DE IEȘIRE

Nume program: EXEMPLELE 6 PC, m, M, F

ZIUA	R1	R2	R3	MEDIA
	**	**	**	** **
Lu	**	**	**	** **
Mi	**	**	**	** **
Je	**	**	**	** **
Vi	**	**	**	** **
Sîmbăta	**	**	**	** **
MEDIA	**,**	** **	** **	
MAXIMUL LIVRĂRIILOR LA REZERVORUL	**			
IN ZIUA DE	=	*****		

e)

Fig. 6.1. a-e) Modulul de analiză structurată: a) formatul datelor de ieșire;

DATE INTRARE

Nume program: EXEMPLELE 6 – PC, m, M, F

INTRODUCEȚI LIVRARILE DE BENZINĂ

PE ZILE IN ORDINEA

	R1	R2	R3
LUNI:			
REZERVORUL 1 ? 15			
REZERVORUL 2 ? 18			
REZERVORUL 3 ? 30			
MARȚI:			
REZERVORUL 1 ? 23			
REZERVORUL 2 ? 40			
REZERVORUL 3 ? 0			
MIERCURI:			
REZERVORUL 1 ? 55			
REZERVORUL 2 ? 26			
REZERVORUL 3 ? 1			
JOI:			
REZERVORUL 1 ? 0			
REZERVORUL 2 ? 0			
REZERVORUL 3 ? 25			
VINERI:			
REZERVORUL 1 ? 12			
REZERVORUL 2 ? 60			
REZERVORUL 3 ? 60			
SÎMBĂȚA:			
REZERVORUL 1 ? 60			
REZERVORUL 2 ? 60			
REZERVORUL 3 ? 60			

b)

TABELA DE VARIABLE

Nume program: EXEMPLELE 6 – PC, m, M, F

Variabile de intrare	Variabile de stare	Variabile de ieșire
A: matricea livrărilor zilnice de benzină	C\$: vectorul zilelor săptămânii L: index coloană (aMIC) I: index Z: index S: total general livrări (zi/rezervor) M: index coloană (aMIC) C: numărul rezervorului R: ziua maximului livrărilor P: număr linie (aMIC); (HC-85) Q: număr coloană (aMIC); (HC-85)	B: vectorul livrărilor medii zilnice D: vectorul livrării medii pe rezervor H: maximul livrărilor C: numărul rezervorului R: ziua maximului livrărilor

c)

b) datele de intrare; c) tabela de variabile;

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 6 – PC, m, M, F

Descrierea programului. Programul editează situația livrărilor de benzină efectuate zilnic din trei rezervoare cilindrice echilaterale.

Livrările, pe zile și pentru fiecare rezervor în parte se citesc de la tastatură. Programul mai afișează maximul livrărilor însoțit de rezervorul și ziua respectivă.

Intrări. Valorile livrărilor, pe zile, pentru fiecare rezervor.

Ieșiri. Lista cu situația livrărilor.

Lista de funcțiuni ale programului

1. Citește zilele săptămânii.
2. Citește livrările zilnice pentru fiecare din cele trei rezervoare.
3. Insumează livrările pentru fiecare rezervor.
4. Calculează media livrărilor pe zile.
5. Inițializează variabila S.
6. Insumează livrările în total general livrări rezervor.
7. Calculează media livrărilor pe rezervor.
8. Afișează ziua, livrările din rezervoare și media livrărilor pe zi.
9. Afișează un rând de " - ".
10. Afișează media livrărilor pe rezervor.
11. Calculează maximul livrărilor.
12. Inițializează variabilele R, C și H.
13. Afișează maximul livrărilor, numărul rezervorului și ziua.
14. Afișează cap tabel.
15. Stop.
16. Calculează P (aMIC).
17. Calculează Q (aMIC).
18. Șterge ecranul.

d)

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 6 – PC, m, M, F

Modul	Funcțiuni
CAP-TAB	14, 18 (aMIC, PRAE)
CIT-ZILE	1
CIT-LIV	2
SUM-LIV	5, 3
MED-LIV	4
SUM-REZ	5, 6
MED-REZ	7
EDIT-RIND	8, 9, 16, 17 (aMIC, HC-85)
EDIT-COL	10, 17 (aMIC, HC-85)
CALC-MAX	11, 12
EDIT-MAX	13
FINAL	15

e)

d) specificații de programare; e) alocarea funcțiunilor de prelucrare.

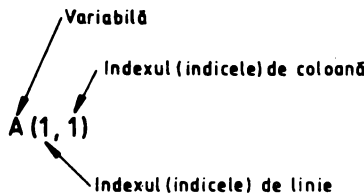
Ne vom opri, în cele ce urmează, asupra tabelii de variabile în care s-au definit două noi tipuri de variabile – variabila structurată (de intrare) de tip matrice (A) și variabila structurată alfanumerică de tip vector C\$ (de stare).

Variabile structurate de tip tablou (matrice)

Tablourile (matricile sau masivele) sînt un ansamblu de elemente dispuse pe linii și coloane. Prelucrările ce se efectuează asupra elementelor unei matrice sînt funcție de valorile unor indici de poziție, ce precizează coordonatele (poziția) liniei și ale coloanei corespunzătoare. Pentru EXEMPLELE 6 – PC, m, M, F s-a definit variabila structurată de tip matrice (tablou) A în care se citesc cantitățile de benzină livrate zilnic din fiecare rezervor. Tabloul conține 6 linii (Z) și 3 coloane (I) pentru livrările zilnice din cele trei rezervoare.

		I → Rezervoare		
		1	2	3
Z ↓ Zile	1	15	18	30
	2	23	40	0
	3	55	26	1
	4	0	0	25
	5	12	60	60
	6	60	60	60

Observație. Z și I sînt definite ca două variabile de stare, reprezentînd indexul (indicele) de linie, respectiv indexul (indicele) de coloană.



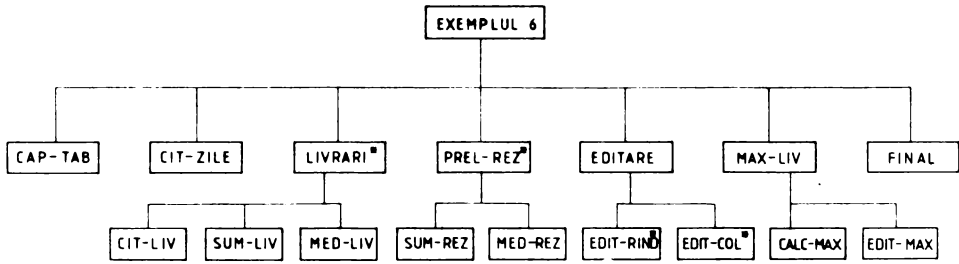
Notă : Se citește "A de unu, unu"

Variabila de stare C\$ (șir sau alfanumerică) a fost introdusă pentru definirea vectorului în care urmează să se citească, în mod static zilele săptămînil.

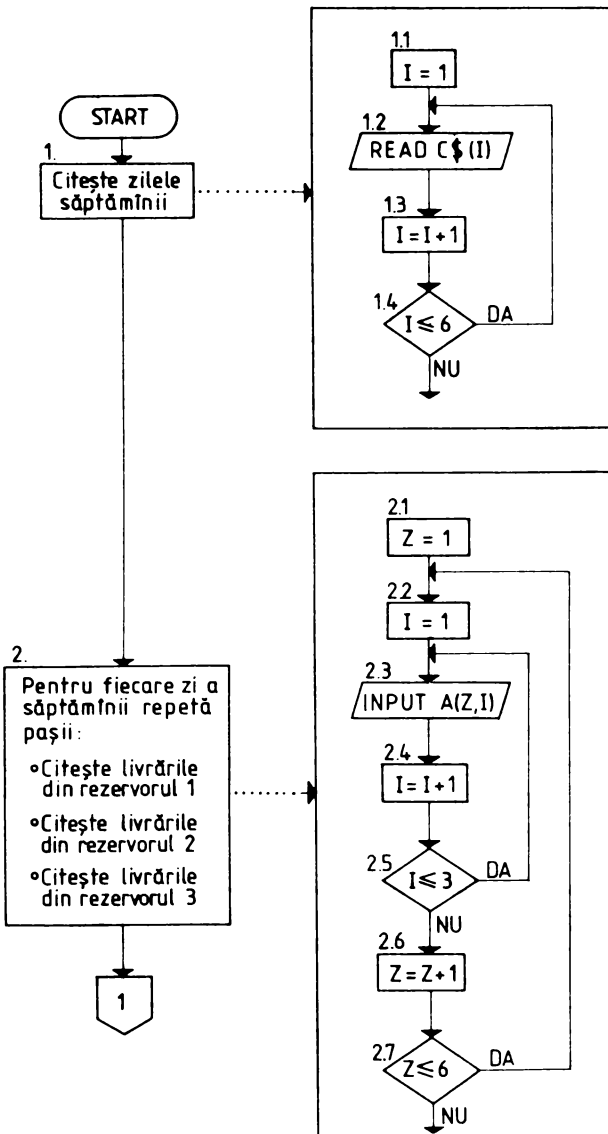
C\$(1)	C\$(2)	C\$(3)	C\$(4)	C\$(5)	C\$(6)
LUNI	MARȚI	MIERCURI	JOI	VINERI	SIMBĂȚĂ

□ Proiectarea programului

Diagrama de structură, schema logică și pseudocodul sint prezentate în modul de proiectare structurată, fig. 6.2.



a)



b)

Fig. 6.2. Modul de proiectare structurată: a) diagrama de structură; b) schema logică a programului BASIC-αMIC;

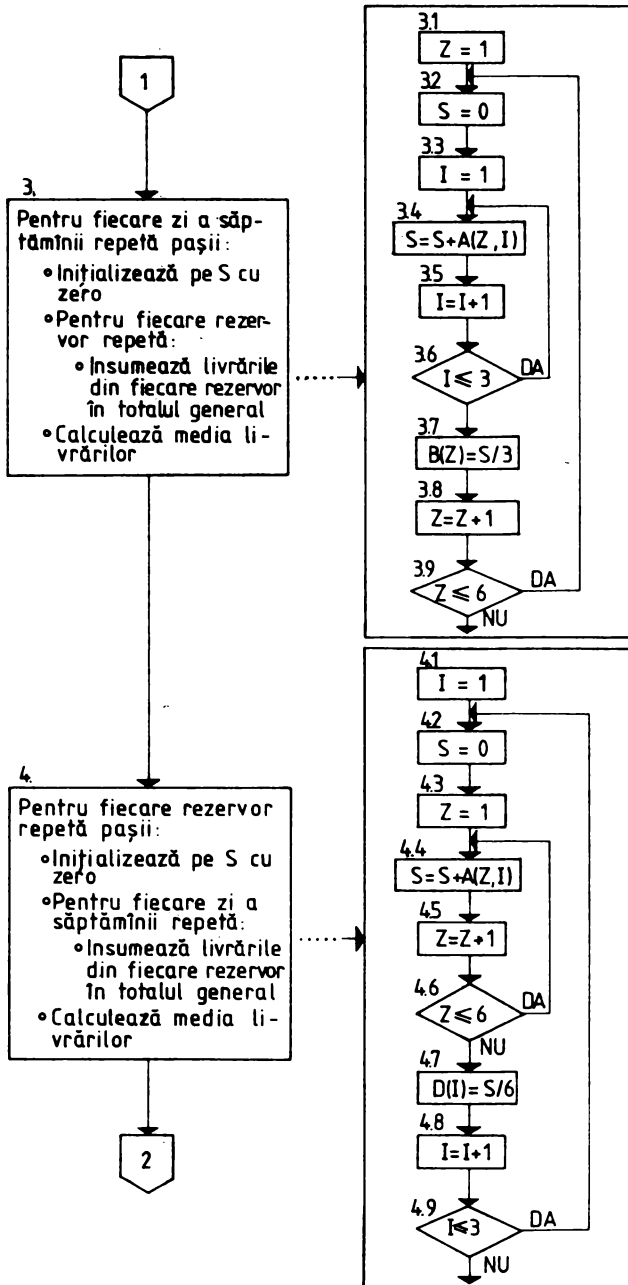


Fig. 6.2(b).

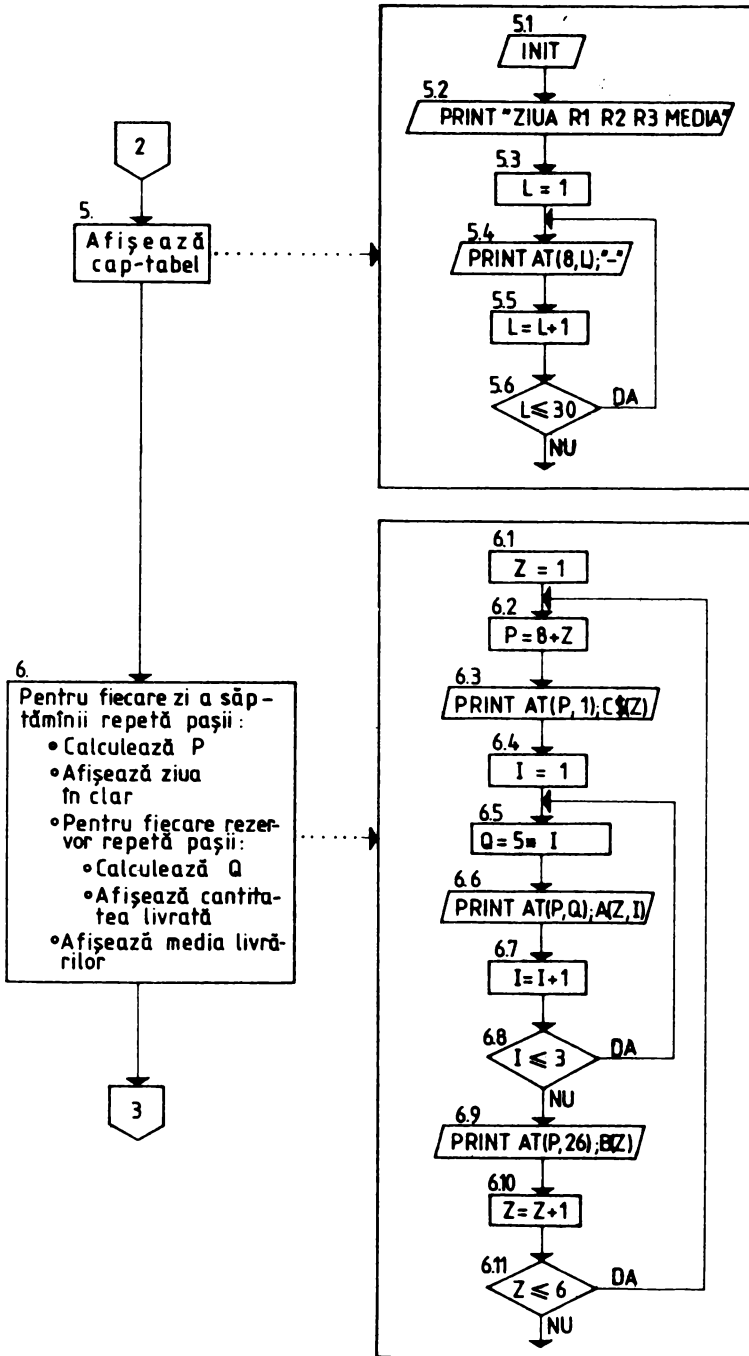


Fig. 6.2(b).

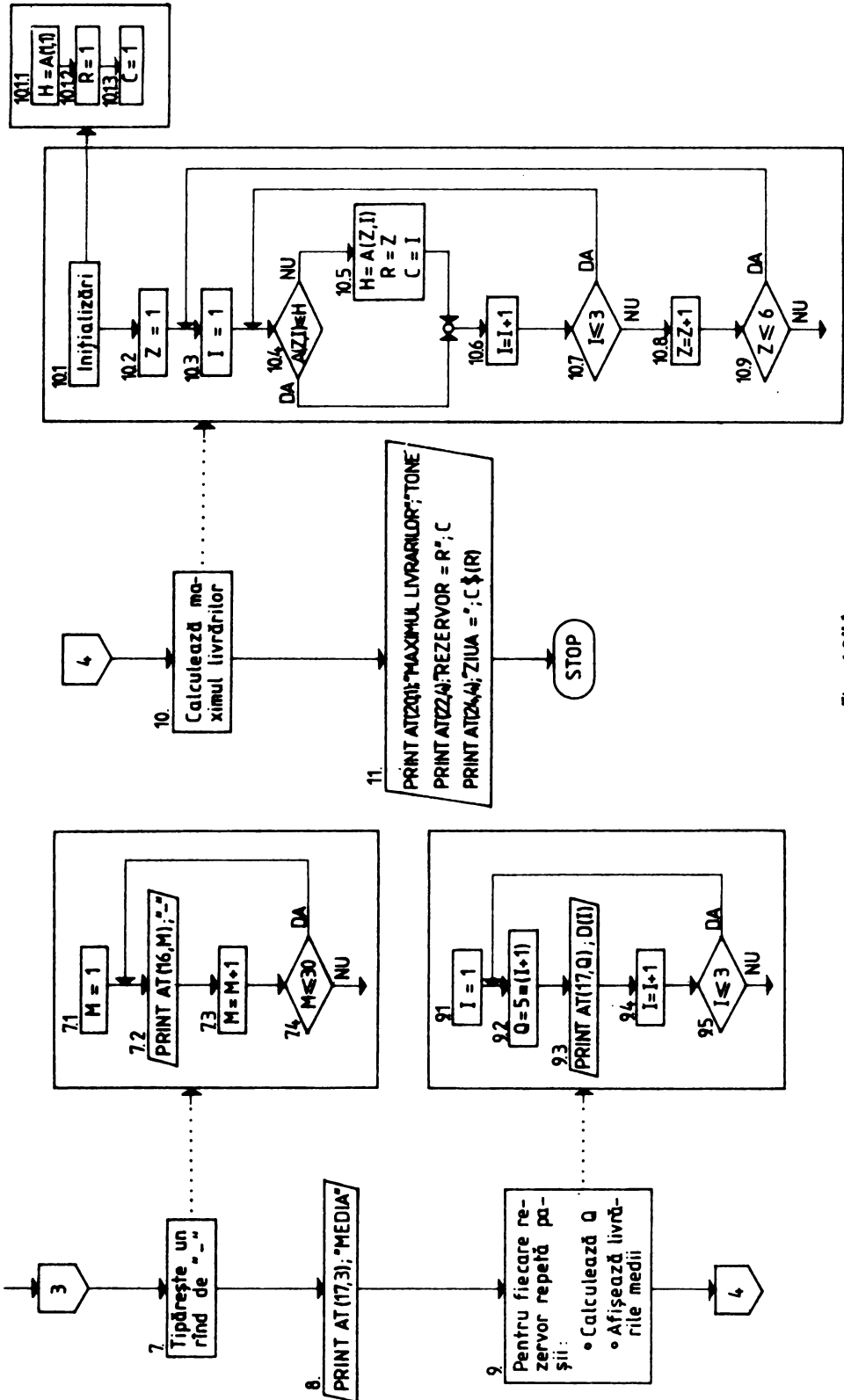


Fig. 6.2(b).

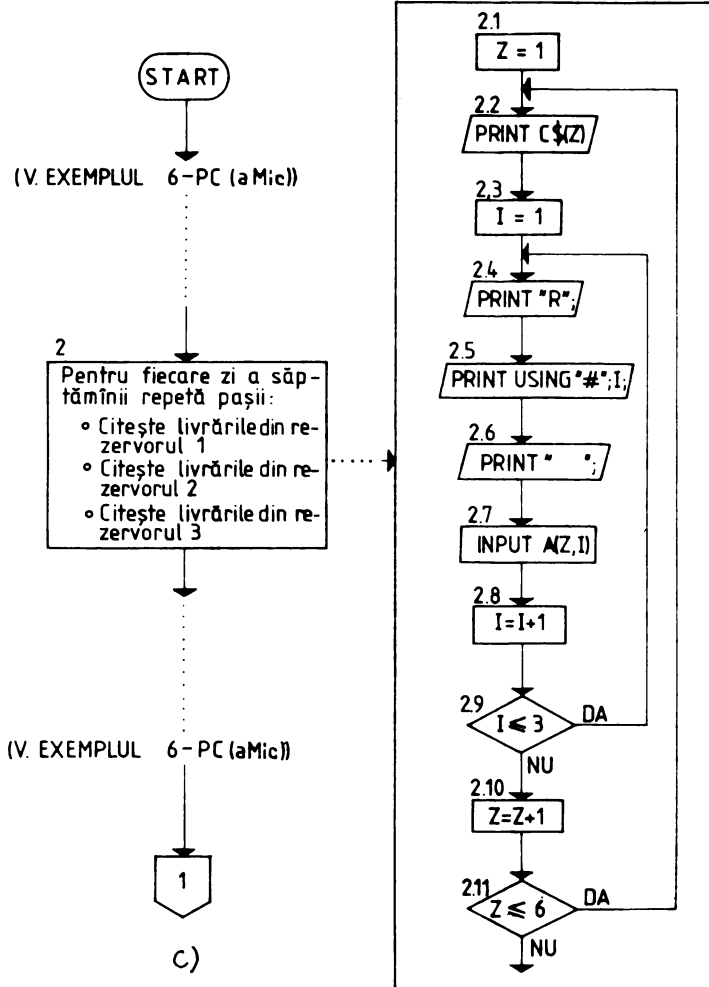


Fig. 6.2. c) schema logică a programului BASIC-PRAE;

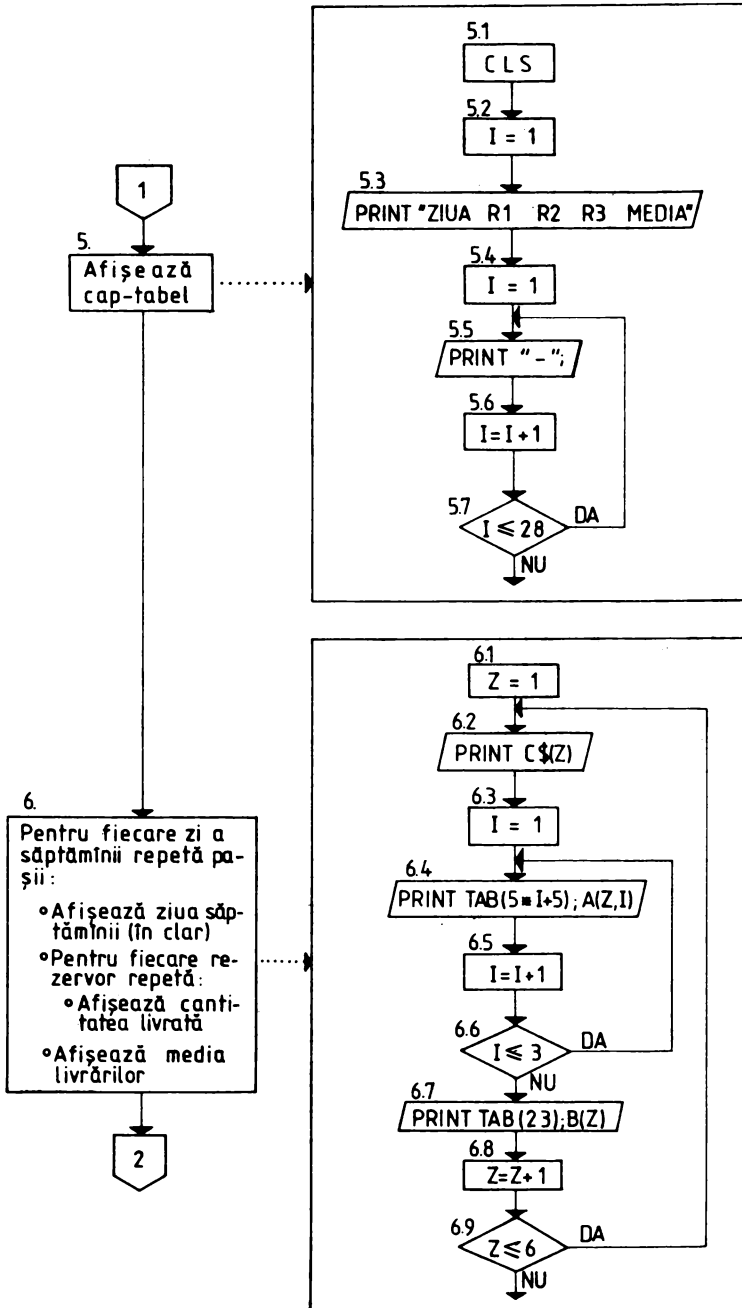


Fig. 6.2(c).

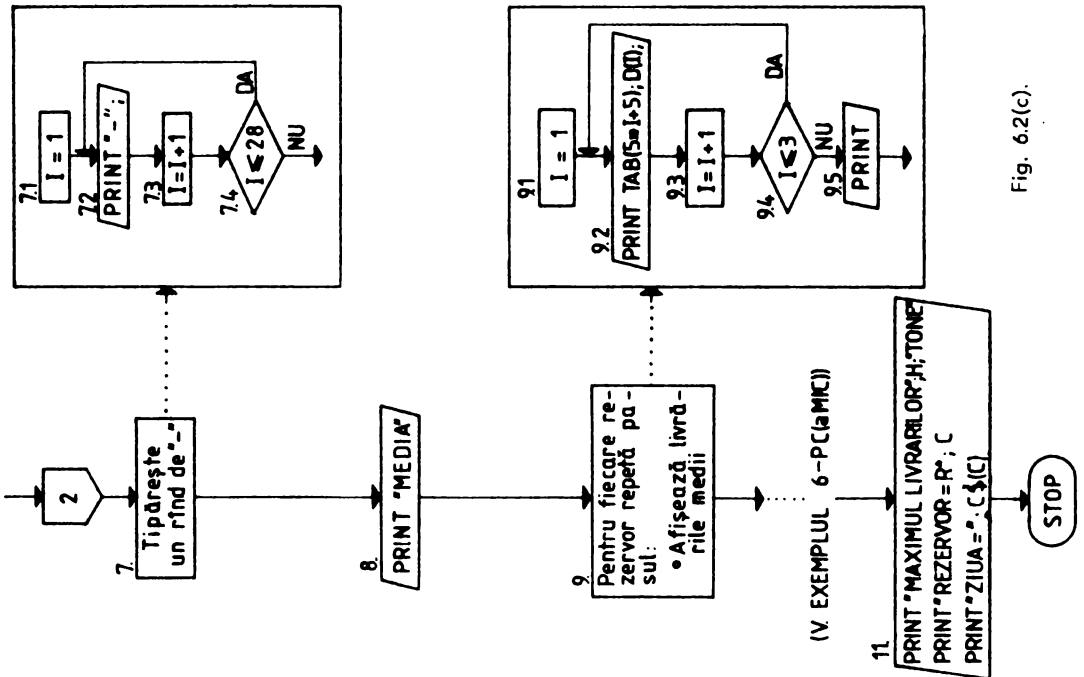
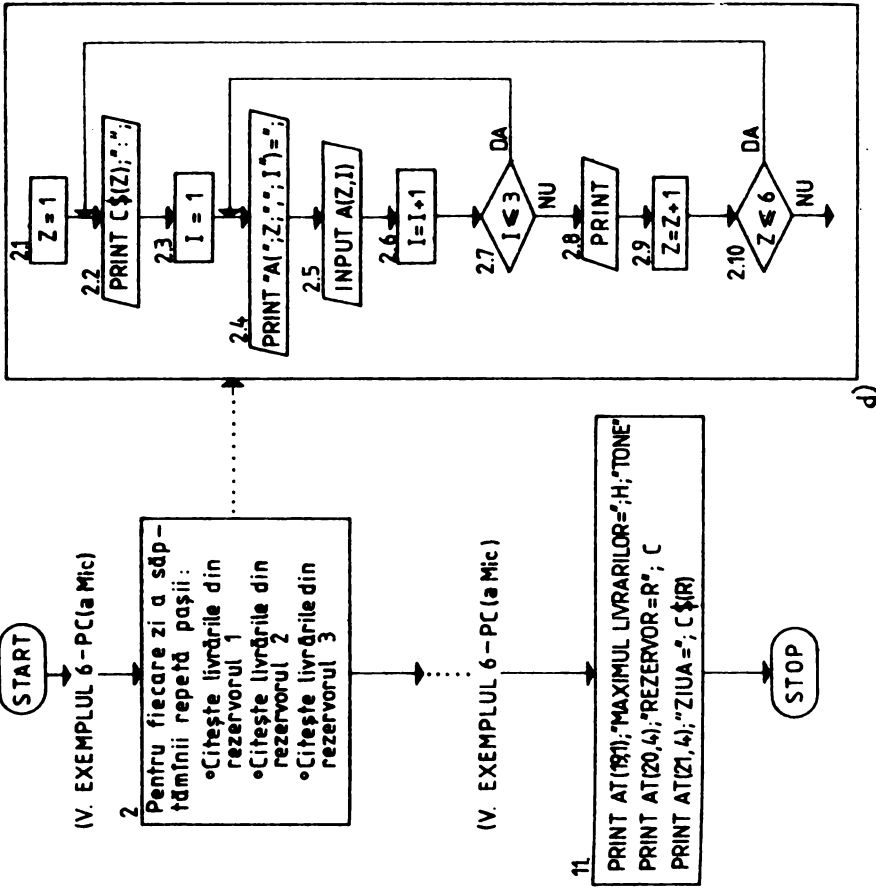
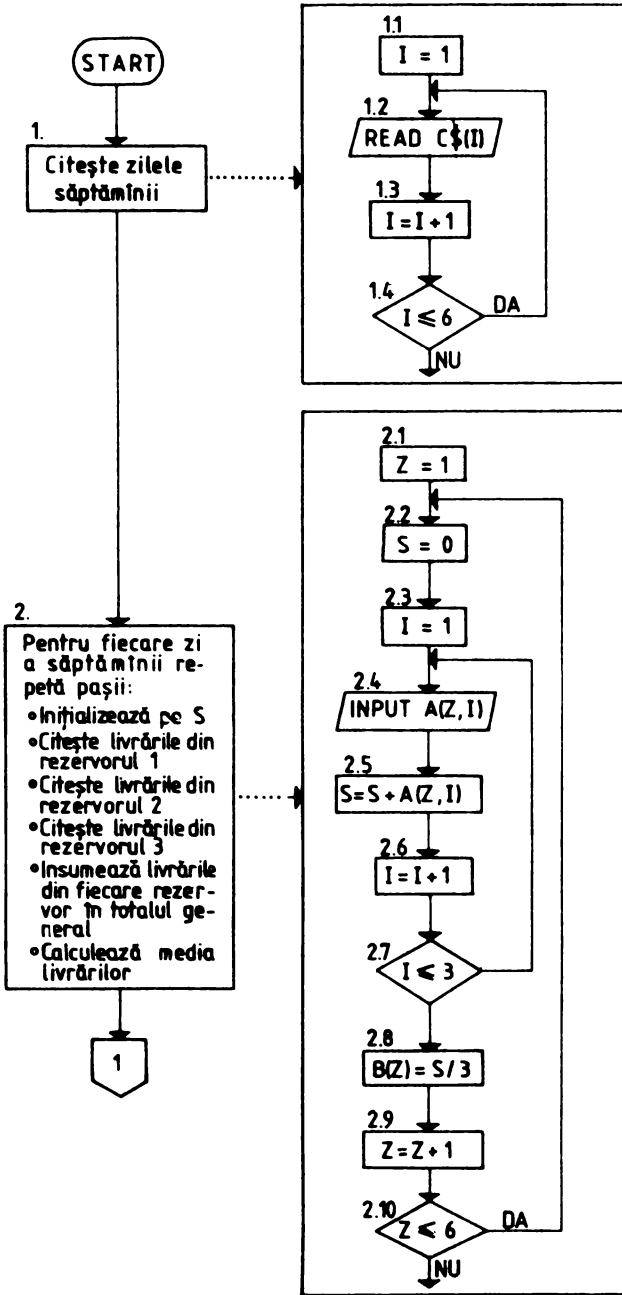


Fig. 6.2(c).



d)

Fig. 6.2(d). schema logică a programului BASIC HC-85, TIM S, SPECTRUM;



e)

Fig. 6.2. e) schema logică a programului BASIC-AMSTRAD, BASIC-80, ABASIC.

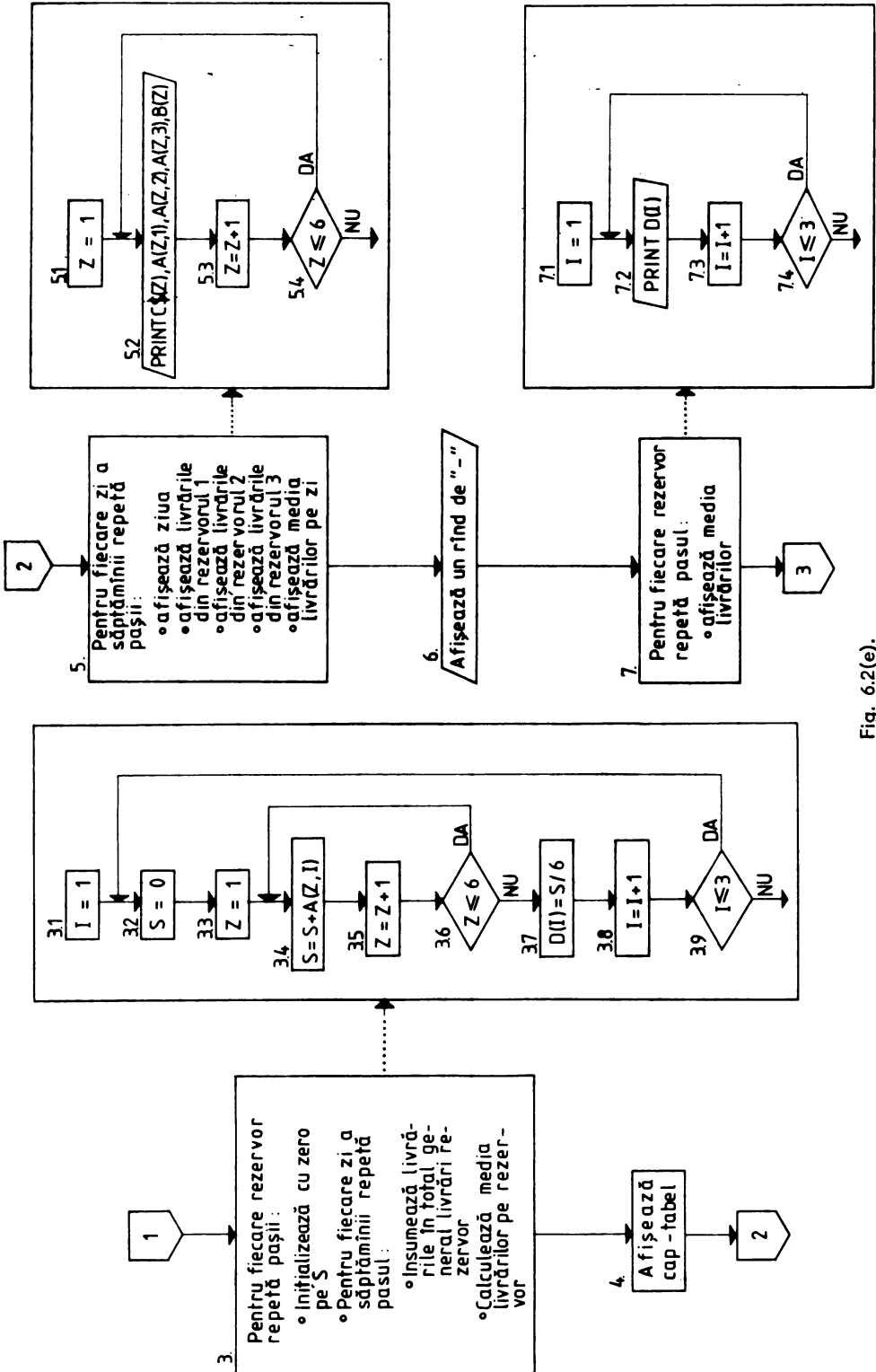
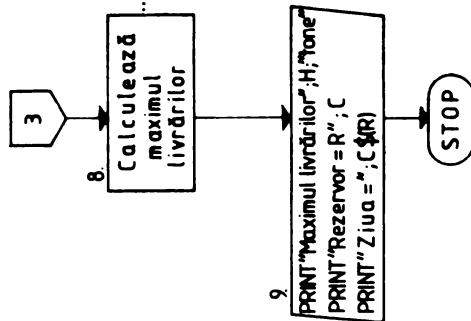
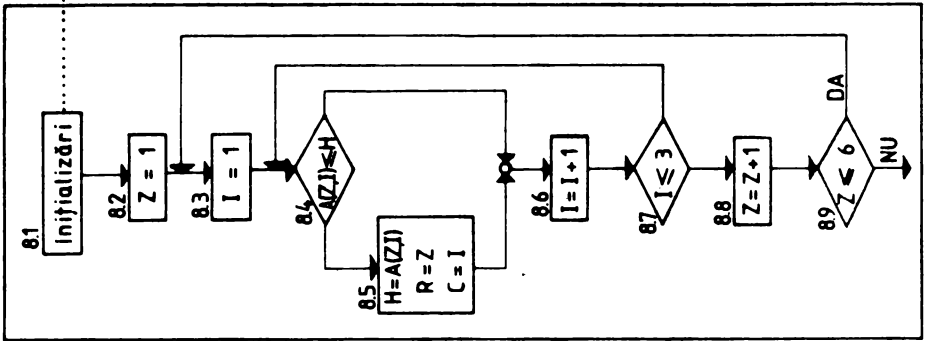
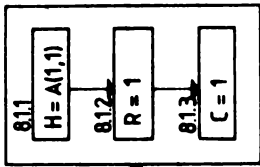
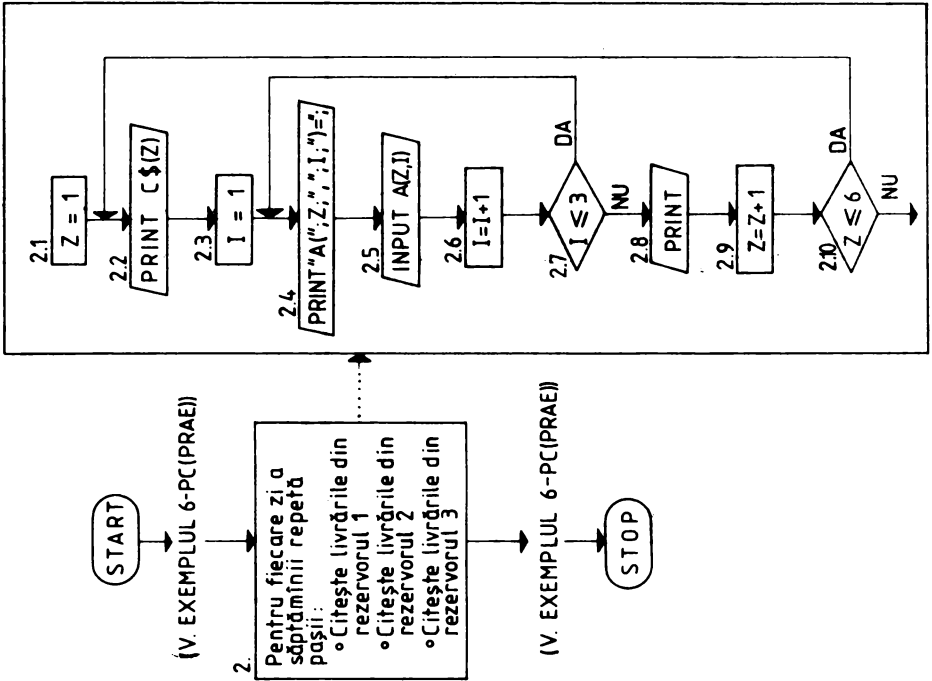


Fig. 6.2(e).



e)



f)

Fig. 6.2(f). f) schema logică a programului BASIC-COMMODORE;

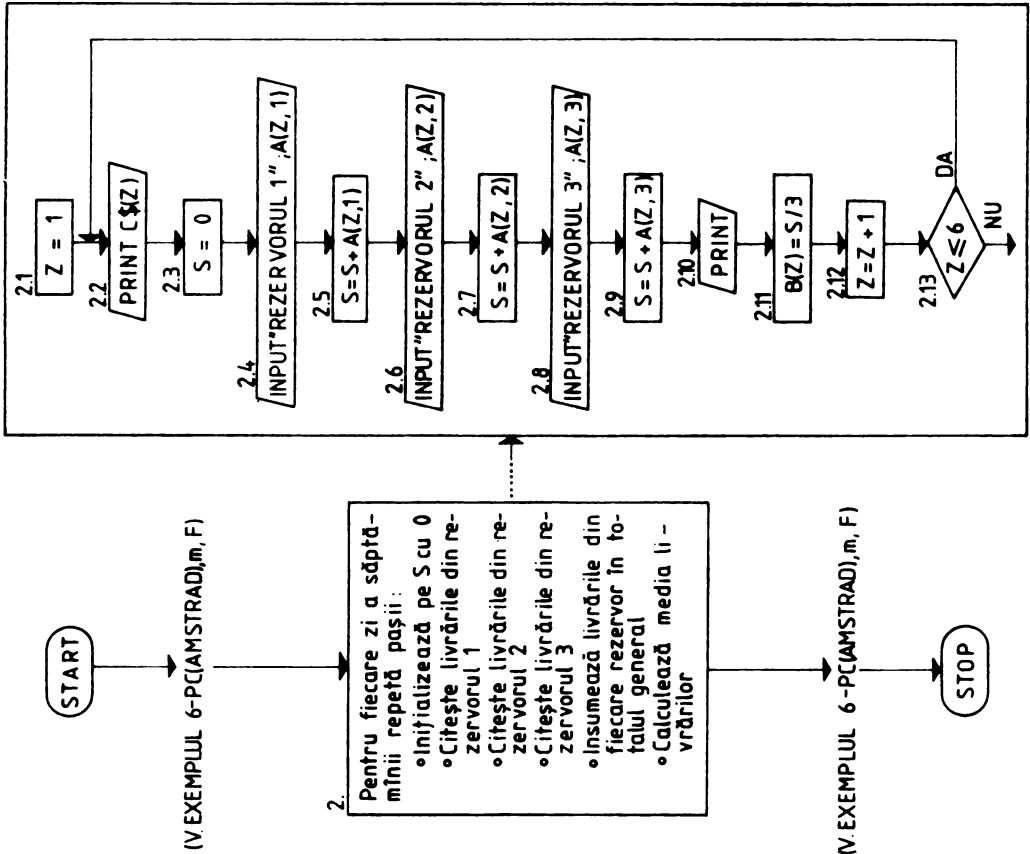


Fig. 6.2(g).

PSEUDOCODUL	
Nume program: EXEMPLUL 6 – PC (aMIC)	
EX 6 PC AMIC	SEQ
CIT-ZILE	PENTRU I DE LA 1 LA 6
	READ C\$(I)
CIT-ZILE	SFIRȘIT
LIVRARI	PENTRU Z DE LA 1 LA 6
C	PENTRU I DE LA 1 LA 3
	INPUT A(Z,I)
C	SFIRȘIT
LIVRARI	PENTRU Z DE LA 1 LA 6
PREL-ZILE	S=0
C1	PENTRU I DE LA 1 LA 3
	S=S+A(Z,I)
C1	SFIRȘIT
	B(Z)=S/3
PREL-ZILE	SFIRȘIT
PREL-REZ	PENTRU I DE LA 1 LA 3
	S=0
C2	PENTRU Z DE LA 1 LA 6
	S=S+A(Z,I)
C2	SFIRȘIT
	D(I)=S/6
PREL-REZ	SFIRȘIT

Fig. 6.2(h).

g) schema logică a programului BASIC-PLUS;
h) pseudocodul programului BASIC-aMIC;

PSEUDOCODUL

Nume program: EXEMPLUL 6 – PC (aMIC)

```

INIT
CAP-TAB      SEQ
              PRINT "ZIUA R1 R2 R3 MEDIA"
C3           PENTRU L DE LA 1 LA 30
              PRINT AT (8, L); " - "
C3           SFIRȘIT
CAP-TAB      END
EDIT-RIND    SEQ
C4           PENTRU Z DE LA 1 LA 6
              P=8+Z
              PRINT AT (P, 1); C$(Z)
C5           PENTRU I DE LA 1 LA 3
              Q=5*I
              PRINT AT (P, Q); A(Z, I)
C5           SFIRȘIT
              PRINT AT (P, 26); B(Z)
C4           SFIRȘIT
EDIT-RIND    END
RINDLIN      SEQ
C6           PENTRU M DE LA 1 LA 30
              PRINT AT (16, M); " - "
C6           SFIRȘIT
RINDLIN      END
EDIT-COL     SEQ
C7           PENTRU I DE LA 1 LA 3
              Q=5*(I+1)
              PRINT AT (17, Q); D(I)
C7           SFIRȘIT
EDIT-COL     END
MAX-LIV      PENTRU Z DE LA 1 LA 6
CALC-MAX     PENTRU I DE LA 1 LA 3
MAX2         DACĂ A(Z, I) > H
MAX3         SEQ
              H=A(Z, I)
              R=Z
              C=I
MAX3         END
MAX2         SFIRȘIT
CALC-MAX     SFIRȘIT
MAX-LIV      SFIRȘIT
EDIT-MAX     SEQ
              PRINT AT (20, 1); "MAXIMUL LIVRĂRILOR"; H; "TONE"
              PRINT AT (22, 4); "REZERVOR=R"; C
              PRINT AT (24, 4); "ZIUA="; C$(R)
EDIT-MAX     END
EX 6 PC AMIC END

```

Fig. 6.2(h) continuare.

PSEUDOCODUL

Nume program: EXEMPLUL 6 – PC (PRAE)

```

EX 6 PC PRAE      SEQ
(v. EXEMPLUL 6 – PC (aMIC))
LIVRĂRI          PENTRU Z DE LA 1 LA 6
                  PRINT C$ (Z)
C                PENTRU I DE LA 1 LA 3
C1              SEQ
                  PRINT "R";
                  PRINT USING "#"; I;
                  PRINT " ";
                  INPUT A (Z, I)
C1              END
C                SFIRȘIT
LIVRĂRI          SFIRȘIT
(v. EXEMPLUL 6 – PC (aMIC))
CAP-TAB         SEQ
                  PRINT "ZIUA R1 R2 R3 MEDIA"
C3              PENTRU I DE LA 1 LA 28
                  PRINT " - ";
C3              SFIRȘIT
CAP-TAB         END
EDIT-RIND       SEQ
C4              PENTRU Z DE LA 1 LA 6
                  PRINT C$ (Z)
C5              PENTRU I DE LA 1 LA 3
                  PRINT TAB (5 * I + 5); A (Z, I)
C5              SFIRȘIT
                  PRINT TAB (23); B(Z)
C4              SFIRȘIT
EDIT-RIND       END
RINDLIN         SEQ
C6              PENTRU I DE LA 1 LA 28
                  PRINT " - ";
C6              SFIRȘIT
RINDLIN         END
EDIT-COL        SEQ
C7              PENTRU I DE LA 1 LA 3
                  PRINT TAB (5 * I + 5); D(I);
C7              SFIRȘIT
EDIT-COL        END
(v. EXEMPLUL 6 – PC (aMIC))
EDIT-MAX        SEQ
                  PRINT "MAXIMUL LIVRĂRILOR"; H; "TONE"
                  PRINT "REZERVOR=R"; C
                  PRINT "ZIUA="; C$ (R)
EDIT-MAX        END
EX 6 PC PRAE    END

```

Fig. 6.2(i). i) pseudocodul programului BASIC-PRAE;

PSEUDOCODUL

Nume program: EXEMPLUL 6 – PC (HC-85, TIM S, SPECTRUM)

```

EX 6 PC HC 85   SEQ
(v. EXEMPLUL 6 – PC (aMIC)
LIVRĂRI        PENTRU Z DE LA 1 LA 6
                PRINT C$(Z); " : " ;
C              PENTRU I DE LA 1 LA 3
CC            SEQ
                PRINT "A (" ; Z; ", " ; " )=" ;
                INPUT A (Z, I)
CC            END
C             SFIRȘIT
LIVRĂRI        SFIRȘIT
                (v. EXEMPLUL 6 – PC (aMIC)
EDIT-MAX       SEQ
                PRINT AT (19, 1); "MAXIMUL LIVRĂRILOR=" ; H; "TONE"
                PRINT AT (20, 4); "REZERVOR=R" ; C
                PRINT AT (21, 4); "ZIUA=" C$(R)
EDIT-MAX       END
EX 6 PC HC 85  END

```

j)

PSEUDOCODUL

Nume program: EXEMPLELE 6 – PC (AMSTRAD), m, F

```

EXEMPLUL6     SEQ
CIT-ZILE      PENTRU I DE LA 1 LA 6
                READ C$(I)
CIT-ZILE      SFIRȘIT
LIVRĂRI       PENTRU Z DE LA 1 LA 6
                S=0
C             PENTRU I DE LA 1 LA 3
                INPUT A (Z, I)
                S=S+A(Z, I)
C             SFIRȘIT
                B(Z)=S/3
LIVRĂRI       SFIRȘIT
PREL-REZ      PENTRU I DE LA 1 LA 3
                S=0
E             PENTRU Z DE LA 1 LA 6
                S=S+A (Z, I)
E             SFIRȘIT
                D(I)=S/6
PREL-REZ      SFIRȘIT
EDIT-RIND     PENTRU Z DE LA 1 LA 6
                PRINT C$(Z), A(Z,1), A(Z,2), A(Z,3), B(Z)
EDIT-RIND     SFIRȘIT
G             PENTRU M DE LA 1 LA 70
                PRINT " - " ;
G             SFIRȘIT
EDIT-COL      PENTRU I DE LA 1 LA 3
                PRINT D(I)
EDIT-COL      SFIRȘIT

```

k)

Fig. 6.2. j) pseudocodul programului BASIC HC-85, TIM S, SPECTRUM; k) pseudocodul programului BASIC-AMSTRAD, BASIC-80, ABASIC.

PSEUDOCODUL

Nume program: EXEMPLELE 6 – PC (AMSTRAD), m, F

```

K          SEQ
           H=A(1,1)
           R=1
           C=1

K          END
MAX-LIV   PENTRU Z DE LA 1 LA 6
CALC-MAX  PENTRU I DE LA 1 LA 3
MAX 2     DACĂ A (Z, I) > H
MAX 3     SEQ
           H=A(Z,I)
           R=Z
           C=I

MAX 3     END
MAX 2     SFIRȘIT
CALC-MAX  SFIRȘIT
MAX-LIV   SFIRȘIT
           Afișează cap-tabel

EDIT-MAX  SEQ
           PRINT H
           PRINT C
           PRINT C$(R)

EDIT-MAX  END
EXEMPLUL 6  END

```

k) continuare

PSEUDOCODUL

Nume program: EXEMPLUL 6 – PC (COMMODORE)

```

EX 6 PC COMMO  SEQ
(v. EXEMPLUL 6 – PC (PRAE))
LIVRĂRI       PENTRU Z DE LA 1 LA 6
               PRINT C$(Z); " : "
C             PENTRU I DE LA 1 LA 3
CC           SEQ
               PRINT "A (" ; Z ; " , " ; " )=" ;
               INPUT A (Z, I)

CC           END
C           SFIRȘIT
LIVRĂRI       SFIRȘIT
(v. EXEMPLUL 6 – PC (PRAE))
EX 6 PC COMMO  END

```

l)

Fig. 6.2(i) l) pseudocodul programului BASIC-COMMODORE;

PSEUDOCODUL

Nume program: EXEMPLUL 6 – M

```

EX 6 PLUS      SEQ
(v. EXEMPLUL 6 – PC (AMSTRAD), m, F)
LIVRĂRI      PENTRU Z DE LA 1 LA 6
L1           SEQ
              PRINT C$(Z)
              S=0
              INPUT "REZERVORUL 1"; A(Z,1)
              S=S+A(Z,1)
              INPUT "REZERVORUL 2"; A(Z,2)
              S=S+A(Z,2)
              INPUT "REZERVORUL 3"; A(Z,3)
              S=S+A(Z,3)
              PRINT
              B(Z)=S/3
L1           END
LIVRĂRI      SFIRȘIT
(v. EXEMPLUL 6 – PC (AMSTRAD), m, F)
EX 6 PLUS    END

```

Fig. 6.2. m) pseudocodul programului BASIC-PLUS.

□ Codificarea în limbajul BASIC-aMIC

vol. 2, pag. 218

Variabile indexate alfanumerice (șir)

Variabilele șir (alfanumerice) vă permit să manipulați date alfanumerice. Le puteți utiliza într-una din următoarele instrucțiuni BASIC: **DIM**, **LET**, **PRINT**, **INPUT**, **READ**, **DATA**, **IF-THEN-ELSE** etc. Pentru calculatorul aMIC numele unei variabile șir este format dintr-o literă urmată de caracterul "\$".

Puteți utiliza și tablouri de variabile șir, tablourile fiind formate din mai multe variabile șir, cu aceeași lungime și același nume. Trebuie avut grijă ca orice tablou de variabile șir pe care-l folosim să-l declarăm în instrucțiunea **DIM**, precizând atât numărul de variabile cât și lungimea acestora. Așa se și explică prezența instrucțiunii

```
5 DIM C$(6, 8), B(6), D(3)
```

în cadrul programului în care declarăm vectorul **C\$** cu șase elemente (zilele lucrătoare ale săptămânii) de lungime 8 caractere (v. ziua de miercuri – cea mai lungă!)

Observație. Tablourile numerice și cele șir pot fi declarate intercalat, în aceeași instrucțiune **DIM** (v. linia 5).

Puteți introduce valori pentru variabilele șir (indexate) folosind una din instrucțiunile **INPUT** sau **READ**.

Cu secvența de instrucțiuni:

```

9 FOR I=1 TO 6
10   READ C$(I)
12 NEXT I
15 DATA LUNI, MARȚI, MIERCURI,
    JOI, VINERI, SIMBĂTA

```

se citesc (static) în variabilele indexate $C\$(1)$, $C\$(2)$, ..., $C\$(6)$ valorile: LUNI, MARTI, ..., SIMBĂȚĂ.

Reguli

- Variabilele șir (exemplu A\$) nu trebuie declarate într-o instrucțiune **DIM**, ele putând fi utilizate direct în instrucțiunile de atribuire sau de intrare/ieșire;
- Tablourile șir trebuie declarate într-o instrucțiune **DIM** înainte de utilizare (folosirea lor se face prin specificarea unei anumite variabile indexate șir și nu a întregului tablou);
- La execuția instrucțiunii **DIM** pentru tablouri de variabile șir se alocă cite un octet pentru fiecare caracter (element);
- Dacă expresia șir atribuită unei variabile șir, este mai lungă decât dimensiunea variabilei (stabilită prin **DIM** sau printr-o atribuire anterioară), atunci se atribuie variabilei șir doar prima parte din expresia șir (egală cu lungimea variabilei) restul expresiei șir ignorându-se;
- Dacă lungimea șirului din membrul drept este mai mică decât lungimea variabilei șir, restul de caractere se completează cu spații;
- Este posibilă utilizarea unor subșiruri (porțiuni) dintr-un șir desemnat de o variabilă șir. Se folosește notația

⟨n1⟩ TO ⟨n2⟩

unde:

- ⟨n1⟩ reprezintă indicele primului caracter al subșirului;
- ⟨n2⟩ este indicele ultimului caracter al subșirului.

Observație. Indicii n1 și n2 pot fi constante, variabile sau expresii, valoarea lor fiind cuprinsă între 1 și lungimea șirului. În cazul absenței indicelui ⟨n1⟩ se consideră că subșirul începe cu primul caracter al șirului iar în cazul absenței indicelui ⟨n2⟩ se consideră că subșirul se termină cu ultimul caracter al subșirului. Se pot utiliza subșiruri ale variabilelor șir ce formează un tablou de variabile șir. Pentru subșiruri formate dintr-un singur caracter, se poate folosi un singur indice, cel al caracterului selectat.

PRINT AT

Cu instrucțiunea **PRINT AT** puteți tipări oriunde pe suprafața ecranului (32 linii×30 coloane) valori, mesaje explicative etc. indiferent de poziția curentă, în care urmau să se afișeze acestea.

Formatul general al instrucțiunii este:

Format general
PRINT AT (⟨nr. linie⟩, ⟨nr. coloană⟩)

în care:

- ⟨nr. linie⟩ reprezintă numărul de linie, cu valori cuprinse între 1 și 32;
- ⟨nr. coloană⟩ reprezintă numărul de coloană cu valori cuprinse între 1 și 30.

Observație. Punctul de coordonate (1,1) se află în colțul din stînga sus al ecranului iar punctul de coordonate (32, 30) în dreapta jos.

În program instrucțiunea **PRINT AT** s-a folosit în liniile 85–95; 405; 415; 510; 525; 535; 605; 715; 855–865.

Aplicație. Analizați singuri, fără ajutorul nostru, instrucțiunile 85–95, 405, 415 din cadrul programului și simulați cu creionul și hirtia modul cum funcționează fiecare dintre ele.

Reguli

- Într-o instrucțiune **PRINT AT** (limbajul BASIC-aMIC) coordonatele pentru ⟨nr. linie⟩ și ⟨nr. coloană⟩ se transmit între paranteze;
- **PRINT AT** este foarte des întrebuițată în afișarea de valori sau texte (mesaje) pe un desen realizat cu instrucțiunile grafice BASIC-aMIC (v. conversațiile 12, 13).

Instrucțiuni matriciale. MAT INPUT

Pentru citirea dinamică a matricei A (6×3 elemente) vom utiliza instrucțiunea matricială **MAT INPUT**.

100 **MAT INPUT** A(6, 3)

De notat că matricea A nefiind declarată în instrucțiunea **DIM** din linia 5 a trebuit definită (specificate dimensiunile) în instrucțiunea **MAT** (v. linia 100).

În Fig. 6.3 se prezintă o imagine a elementelor componente ale acestui tablou ca rezultat al execuției instrucțiunii **MAT INPUT** din program.

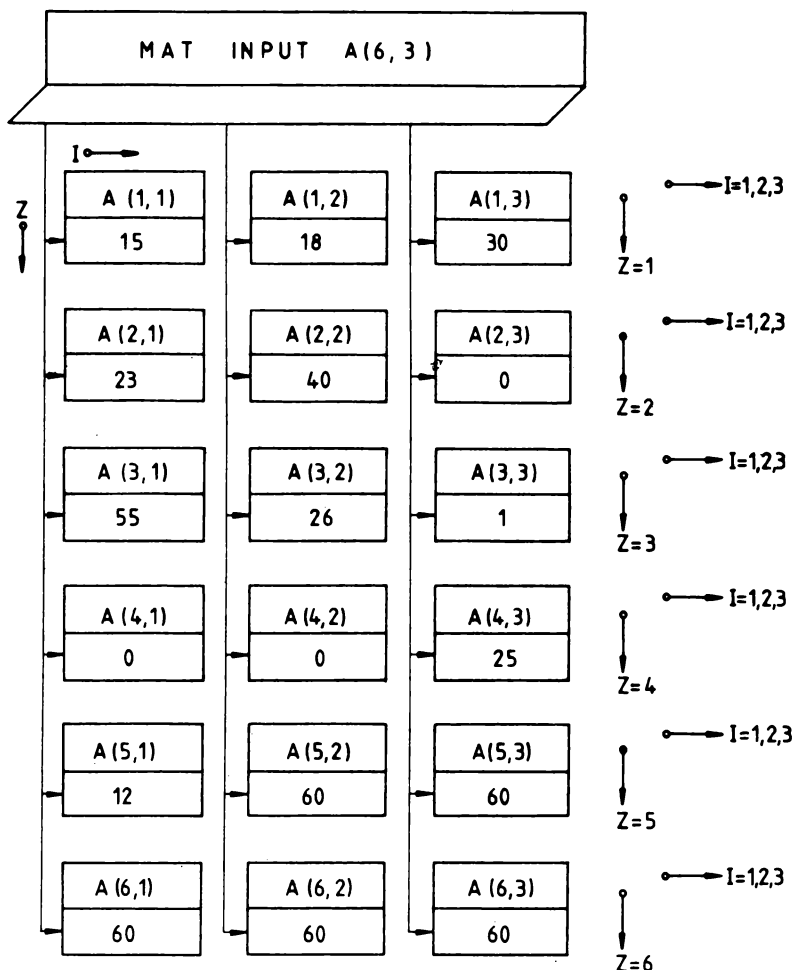


Fig. 6.3.

Observație. Instrucțiunea **MAT INPUT** spre deosebire de instrucțiunile cunoscute pentru prelucrarea tablourilor nu face referiri la fiecare element al tabloului.

În BASIC-aMIC instrucțiunile matriciale pot fi împărțite în două categorii: a) instrucțiuni de intrare/ieșire (**MAT INPUT**, **MAT READ**, **MAT PRINT**); b) instrucțiuni de atribuire – în care matricei din membrul stâng i se atribuie rezultatul unei operații matriciale (Exemplu: **MAT X=Y**; **MAT X=Y * Z** etc.).

Reguli

- Datorită prefixului **MAT**, utilizat în instrucțiuni, tablourile vor fi numite matrice (indiferent dacă au una sau două dimensiuni);
- Matricele introduse prin instrucțiunile de intrare/ieșire sau matricea din membrul stâng al unei instrucțiuni de atribuire, pot fi alocate la execuția instrucțiunii **MAT** respective, nefiind necesară declararea lor în instrucțiunea **DIM**;
- În cazul în care matricele au fost declarate în instrucțiunea **DIM**, sau au fost introduse printr-o instrucțiune **MAT** anterioară, ele pot fi redimensionate dacă: a) numărul de dimensiuni ale tabloului se păstrează; b) numărul de elemente al matricei redimensionate nu depășește numărul de elemente al matricei inițiale;
- Indecșii specificați într-o instrucțiune matricială pot fi: constante, variabile sau expresii.

În figura 6.4 se prezintă instrucțiunile matriciale (input/output și de atribuire) acceptate de limbajul BASIC-aMIC. Citeva exemple privind utilizarea acestor instrucțiuni pot fi urmărite în tabelul 6.1.

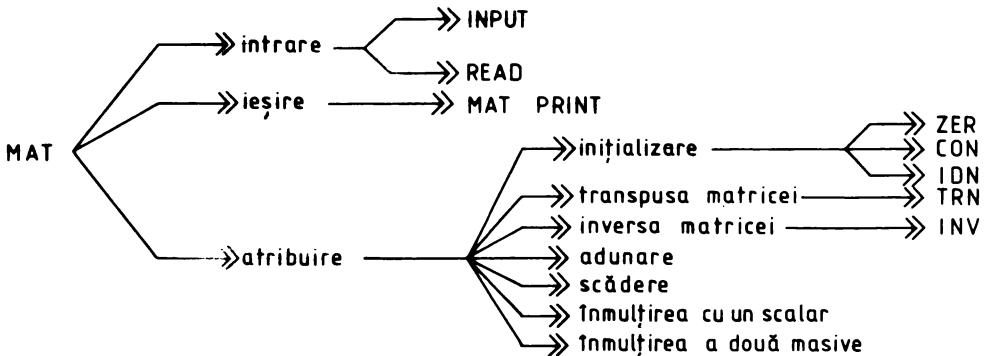


Fig. 6.4.

Tabelul 6.1

Instrucțiune	Funcțiune
MAT INPUT X, Y	Citește dinamic datele pentru matricile X, Y
MAT READ X, Y	Citește static datele pentru matricile X și Y
MAT PRINT X, Y	Tipărește linie cu linie valorile din matricile X și Y
MAT X=Y	Atribuie matricei A valorile matricei Y (pentru aceleași dimensiuni)
MAT X=Y+Z	Adună matricea Y cu matricea Z. Rezultatul se atribuie matricei X
MAT X=Y-Z	Scade matricile Y și Z. Rezultatul se atribuie matricei X
MAT X=Y * Z	Înmulțește matricea Y cu Z
MAT X=(expresie) * Y	Înmulțește matricea Y cu scalarul (expresie)
MAT X=INV(B)	Inversează matricea pătrată B
MAT X=TRN(Y)	Calculează transpusa matricei Y
MAT X=ZER	Inițializează cu zero elementele matricei X
MAT X=CON	Inițializează cu unu elementele matricei X
MAT X=IDN	Inițializează cu unu elementele diagonalei principale

TEST

a) În programul

```

10 DIM C(20)
20 READ N
30 FOR I=1 TO N
40   C(I)=0
50 NEXT I
60 DATA 7
70 END

```

liniile 30, 40, 50 pot fi înlocuite cu o singură instrucțiune **MAT**;
 30 **MAT C=**

R. 30 **MAT C=ZER (N)**

b) În loc de:

```

10 C(1)=0
20 C(2)=0

```

putem scrie

```

10 MAT C=

```

R. 10 **MAT C=ZER (2)**

c) Ce este greșit în instrucțiunea de mai jos?

```

10 DIM A(10)
20 MAT A= ZER (11)

```

R. Vectorul A are maximum 10 elemente (v. linia 10).

d) Pentru fiecare din instrucțiunile de mai jos, înlocuiți-le cu instrucțiunea **MAT ZER** corespunzătoare.

d1) 10 **DIM A(6)**
 20 **A(1)=0**
 30 **A(2)=0**
 40 **A(3)=0**
 50 **END**

d2) 10 **DIM A (99)**
 20 **FOR I=1 TO 99**
 30 **A(I)=0**
 40 **NEXT I**
 50 **END**

R. d1) **MAT A=ZER (3)**; d2) **MAT A=ZER (99)**

e) Precizați ce va afișa calculatorul?

```

10 DIM A(7)
20 MAT A=ZER (5)
30 MAT PRINT A(5)
40 END
RUN

```

R.

0
 0
 0
 0
 0

f) Scrieți o instrucțiune **MAT PRINT** care să înlocuiască următoarea buclă **FOR-NEXT** (se omit etichetele).

```

FOR I=1 TO N
  PRINT C (I)
NEXT I

```

R. **MAT PRINT C(N)**

g) Scrieți o instrucțiune **MAT INPUT** care să înlocuiască următoarea buclă **FOR-NEXT**.

```
FOR I=1 TO M
  INPUT A(I)
NEXT I
```

R. **MAT INPUT A (M)**

Aplicație. Să se rezolve sistemul de ecuații

$$\begin{cases} 2x-9y-5z=2 \\ 7x-6y+5z=-35 \\ 9x-6y+5z=-39 \end{cases}$$

Fie C matricea coeficienților, S vectorul (coloană) soluției și K vectorul termenilor liberi.

$$C = \begin{bmatrix} 2 & -9 & -5 \\ 7 & -6 & +5 \\ 9 & -6 & +5 \end{bmatrix} \quad S = \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad K = \begin{bmatrix} 2 \\ -35 \\ -39 \end{bmatrix}$$

Sistemul devine:

$$C * S = K \quad (1)$$

sau

$$C^{-1} * C * S = C^{-1} * K \quad (2)$$

sau

$$I * S = C^{-1} * K \quad (3)$$

de unde rezultă:

$$S = C^{-1} * K \quad (4)$$

```
10 DIM C(3,3), S(3,1), K(3,1), N(3,3)
20 MAT READ C, K
30 MAT N= INV (C)
40 MAT S= N*K
50 PRINT "SOLUȚIE"
60 MAT PRINT S
70 DATA 2, -9, -5, 7, -6, 5, 9, -6, 5
80 DATA 2, -35, -39
90 END
```

Revenind la PRINT AT

Pentru afișarea pe suprafața ecranului în puncte de coordonate date (calculate) a livrărilor zilnice și a livrării medii calculate, pe zile, s-au scris următoarele instrucțiuni:

```
500 FOR Z=1 TO 6
505   P=8+Z
510   PRINT AT (P, 1); C$(Z)
515   FOR I=1 TO 3
520     Q=5*I
525     PRINT AT (P, Q); A(Z, I)
530   NEXT I
535   PRINT AT (P, 26); B (Z)
540 NEXT Z
```

Pentru înțelegerea mecanismului de funcționare a instrucțiunilor **PRINT AT**:

```
510 PRINT AT (P, 1); C$ (Z)
525 PRINT AT (P, Q); A(Z,I)
535 PRINT AT (P,26); B(Z)
```

consultați fig. 6.5.

NR.LINE	INSTRUCTIUNE	Z	P	I	Q	C\$(Z)	B(Z)	A(Z,I)	
505	$P = 8 + Z$	1	9	-	-	-	-	-	
510	PRINT AT(P,1); C\$(Z)	1	9	-	-	LUNI	-	-	
520	$Q = 5 * I$	1	9	1	5	LUNI	-	15	
510	PRINT AT(P,Q); A(Z,I)	1	9	1	5	LUNI	-		
520	$Q = 5 * I$	1	9	2	10	LUNI	-	15	
510	PRINT AT(P,Q); A(Z,I)	1	9	2	10	LUNI	-		
520	$Q = 5 * I$	1	9	3	15	LUNI	-	18	
510	PRINT AT(P,Q); A(Z,I)	1	9	3	15	LUNI	-		
535	PRINT AT(P,26); B(Z)	1	9	4	15	LUNI	21	30	
...									
505	$P = 8 + Z$	6	14	-	-	VINERI	40	60	
510	PRINT AT(P,1); C\$(Z)	6	14	-	-	SIMBATA	40	60	
520	$Q = 5 * I$	6	14	1	5	SIMBATA	40	60	
510	PRINT AT(P,Q); A(Z,I)	6	14	1	5	SIMBATA	40	60	
520	$Q = 5 * I$	6	14	2	10	SIMBATA	40	60	
510	PRINT AT(P,Q); A(Z,I)	6	14	2	10	SIMBATA	40	60	
520	$Q = 5 * I$	6	14	3	15	SIMBATA	40	60	
510	PRINT AT(P,Q); A(Z,I)	6	14	3	15	SIMBATA	40	60	
535	PRINT AT(P,26); B(Z)	6	14	4	15	SIMBATA	60	60	

Fig. 6.5.

Cu setul de instrucțiuni:

```
600 FOR M=1 TO 30
605 PRINT AT (16, M); "-"
610 NEXT M
```

se trasează pe linia 16 a ecranului un rând de "-", începînd cu coloana 1 pînă în coloana 30 inclusiv. Pentru afișarea livrărilor medii calculate, (pe rezervor) s-au scris instrucțiunile:

```
700 PRINT AT (17, 3); "MEDIA"
705 FOR I=1 TO 3
710 Q=5*(I+1)
715 PRINT AT (17, Q); D (I)
720 NEXT I
```

În figura 6.6 se ilustrează modul în care poate fi utilizată instrucțiunea **PRINT AT**

```
715 PRINT AT (17, Q); D(I)
```

NR.LINIE	INSTRUCȚIUNE	I	Q	D(I)	
710	$Q = 5 * (I + 1)$	1	10	27.5	
715	PRINT AT(17,Q); D(I)	1	10	27.5	
710	$Q = 5 * (I + 1)$	2	15	34	
715	PRINT AT(17,Q); D(I)	2	15	34	
710	$Q = 5 * (I + 1)$	3	20	29.3333	
715	PRINT AT(17, Q); D(I)	3	20	29.3333	

Fig. 6.6.

atunci cînd unul din parametri (17) se menține constant.

Aplicație

Completați programul cu următoarea secvență de instrucțiuni:

```
18 INIT
20 FOR J=1 TO 4
25 PRINT AT (15, 2); " ? "
30 PRINT AT (16, 2); " ? "
35 PRINT AT (17, 2); " ? "
40 PRINT AT (18, 2); " ? "
45 FOR K=1 TO 150
50 NEXT K
55 INIT
60 NEXT J
```

Introduceți în instrucțiunile 25, 30, 35, 40 (v. semnul întrebării) – caracterele semi-grafice (de pe tastatură) corespunzătoare afișării mesajului:

EXEMPLUL 6

Aplicație. Introduceți și executați următoarele programe:

- a) 10 DIM X(3,4), Y(6,10), Z(8,9)
 20 READ L, C
 30 MAT READ X, Y(2,5), Z(L,C)
 40 MAT PRINT Y, Y; Z
 50 DATA 4,3
 60 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
 70 DATA 13, 14, 15, 16, 17, 18, 19, 20, 21, 22
 80 DATA 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34
 90 END
- b) 10 FOR I=1 TO 4
 20 FOR J=1 TO 4
 30 A(I, J)=-8
 40 NEXT J
 50 NEXT I
 60 FOR I=1 TO 4
 70 FOR J=1 TO 4
 80 PRINT A(I, J);
 90 NEXT J
 95 PRINT
 96 PRINT
 100 NEXT I
 110 END
- c) 10 DIM A(4,4)
 20 MAT READ A
 30 MAT PRINT A;
 40 DATA -8, -8, -8, -8, -8, -8, -8, -8
 50 DATA -8, -8, -8, -8, -8, -8, -8, -8
 60 END

Prelucrarea matricelor. Determinarea maximului livrărilor

În cadrul programului intervin prelucrări ale datelor organizate în vectori și matrici.

Astfel, valorile livrărilor medii pe zile calculate în bucla **FOR-NEXT** din liniile 200–230 se depun în variabila B(Z),

```
200 FOR Z=1 TO 6
205   S=0
210   FOR I=1 TO 3
215     S=S+A(Z,I)
220   NEXT I
225   B(Z)=S/3
230 NEXT Z
```

iar valorile livrărilor medii pe fiecare rezervor se depun în variabila D(I).

```
300 FOR I=1 TO 3
305   S=0
310   FOR Z=1 TO 6
315     S=S+A(Z,I)
320   NEXT Z
325   D(I)=S/6
330 NEXT I
```

În tabelele 6.2, 6.3 și figurile 6.7, 6.8 puteți urmări modul de desfășurare a calculelor pentru fiecare caz în parte.

Tabelul 6.2

Nr. linie	Instrucțiune BASIC	Z	I	A	S	B
0	1	2	3	4	5	6
205	S=0	1	0	0	0	0
215	S=S+A(Z,I)	1	1	A(1,1) 15	A(1,1) 15	0
215	S=S+A(Z,I)	1	2	A(1,2) 18	A(1,1)+A(1,2) 15+18	0
215	S=S+A(Z,I)	1	3	A(1,3) 30	A(1,1)+A(1,2)+A(1,3) 15+18+30	0
225	B(Z)=S/3	1	4	A(1,3) 30	A(1,1)+A(1,2)+A(1,3) 15+18+30	21
205	S=0	2	4	-	0	21
215	S=S+A(Z,I)	2	1	A(2,1) 23	A(2,1) 23	21
215	S=S+A(Z,I)	2	2	A(2,2) 40	A(2,1)+A(2,2) 23+40	21
215	S=S+A(Z,I)	2	3	A(2,3) 0	A(2,1)+A(2,2)+A(2,3) 23+40+0	21
225	B(Z)=S/3	2	4	A(2,3) 0	A(2,1)+A(2,2)+A(2,3) 23+40+0	21
205	S=0	3	4	-	-	-
215	S=S+A(Z,I)	3	1	A(3,1) 55	A(3,1) 55	21
215	S=S+A(Z,I)	3	2	A(3,2) 26	A(3,1)+(3,2) 55+26	21
215	S=S+A(Z,I)	3	3	A(3,3) 1	A(3,1)+A(3,2)+A(3,3) 55+26+1	21
225	B(Z)=S/3		4	-	A(3,1)+A(3,2)+A(3,3) 55+26+1	27.333

Tabelul 6.2 (continuare)

0		2	3	4	5	6
205	$S=0$	4	4	—	0	27.3333
215	$S=S+A(Z,I)$	4	1	$A(4,1)$ 0	$A(4,1)$ 0	27.3333
215	$S=S+A(Z,I)$	4	2	$A(4,2)$ 0	$A(4,1)+A(4,2)$ $0+0$	27.3333
215	$S=S+A(Z,I)$	4	3	$A(4,3)$ 25	$A(4,1)+A(4,2)+A(4,3)$ $0+0+25$	27.3333
225	$B(Z)=S/3$	4	4	—	$A(4,1)+A(4,2)+A(4,3)$ $0+0+25$	8.3333
205	$S=0$	5	4	—	0	8.3333
215	$S=S+A(Z,I)$	5	1	$A(5,1)$ 12	$A(5,1)$ 12	8.3333
215	$S=S+A(Z,I)$	5	2	$A(5,2)$ 60	$A(5,1)+A(5,2)$ $12+60$	8.3333
215	$S=S+A(Z,I)$	5	3	$A(5,3)$ 60	$A(5,1)+A(5,2)+A(5,3)$ $12+60+60$	8.3333
225	$B(Z)=S/3$	5	4	—	$A(5,1)+A(5,2)+A(5,3)$ $12+60+60$	44
205	$S=0$	6	4	—	0	44
215	$S=S+A(Z,I)$	6	1	$A(6,1)$ 60	$A(6,1)$ 60	44
215	$S=S+A(Z,I)$	6	2	$A(6,2)$ 60	$A(6,1)+A(6,2)$ $60+60$	44
215	$S=S+A(Z,I)$	6	3	$A(6,3)$ 60	$A(6,1)+A(6,2)+A(6,3)$ $60+60+60$	44
225	$B(Z)=S/3$	6	4	—	$A(6,1)+A(6,2)+A(6,3)$ $60+60+60$	60

Tabelul 6.3

Nr. linie	Instrucțiune BASIC	I						Z	A						S	D
		1	2	3	4	5	6		7	8	9	10	11	12		
0		1	2	3	4								5		6	
305	S=0	1		0	0								0		0	
315	S=S+A(Z,I)	1	1	1	A(1,1) 15							A(1,1) 15			0	
315	S=S+A(Z,I)	1	2	2	A(2,1) 23							A(1,1)+A(2,1) 15+23			0	
315	S=S+A(Z,I)	1	3	3	A(3,1) 55							A(1,1)+A(2,1)+A(3,1) 15+23+55			0	
315	S=S+A(Z,I)	1	4	4	A(4,1) 0							A(1,1)+A(2,1)+A(3,1)+A(4,1) 15+23+55+0			0	
315	S=S+A(Z,I)	1	5	5	A(5,1) 12							A(1,1)+A(2,1)+A(3,1)+A(4,1)+A(5,1) 15+23+55+0+12			0	
315	S=S+A(Z,I)	1	6	6	A(6,1) 60							A(1,1)+A(2,1)+A(3,1)+A(4,1)+A(5,1)+A(6,1) 15+23+55+0+12+60			0	
325	D(I)=S/6	1	7	7	-							15+23+55+0+12+60			27.5	
305	S=0	2	7	7	-							0			27.5	
315	S=S+A(Z,I)	2	1	1	A(1,2) 18							A(1,2) 18			27.5	
315	S=S+A(Z,I)	2	2	2	A(2,2) 40							A(1,2)+A(2,2) 18+40			27.5	
315	S=S+A(Z,I)	2	3	3	A(3,2) 26							A(1,2)+A(2,2)+A(3,2) 18+40+26			27.5	

315	$S=S+A(Z,I)$	2	4	$A(4,2)$ 0	$A(1,2)+A(2,2)+A(3,2)+A(4,2)$ $18+40+26+0$	27.5
315	$S=S+A(Z,I)$	2	5	$A(5,2)$ 60	$A(1,2)+A(2,2)+A(3,2)+A(4,2)+A(5,2)$ $18+40+26+0+60$	27.5
315	$S=S+A(Z,I)$	2	6	$A(6,2)$ 60	$A(1,2)+A(2,2)+A(3,2)+A(4,2)+A(5,2)+A(6,2)$ $18+40+26+0+60+60$	34
325	$D(I)=S/6$					
305	$S=0$	3	7		0	34
315	$S=S+A(Z,I)$	3	1	$A(1,3)$ 30	$A(1,3)$ 30	34
315	$S=S+A(Z,I)$	3	2	$A(2,3)$ 0	$A(1,3)+A(2,3)$ 30+0	34
315	$S=S+A(Z,I)$	3	3	$A(3,3)$ 1	$A(1,3)+A(2,3)+A(3,3)$ 30+0+1	34
315	$S=S+A(Z,I)$	3	4	$A(4,3)$ 25	$A(1,3)+A(2,3)+A(3,3)+A(4,3)$ 30+0+1+25	34
315	$S=S+A(Z,I)$	3	5	$A(5,3)$ 60	$A(1,3)+A(2,3)+A(3,3)+A(4,3)+A(5,3)$ 30+0+1+25+60	34
315	$S=S+A(Z,I)$	3	6	$A(6,3)$ 60	$A(1,3)+A(2,3)+A(3,3)+A(4,3)+A(5,3)+A(6,3)$ 30+0+1+25+60+60	34
325	$D(I)=S/6$					29,333

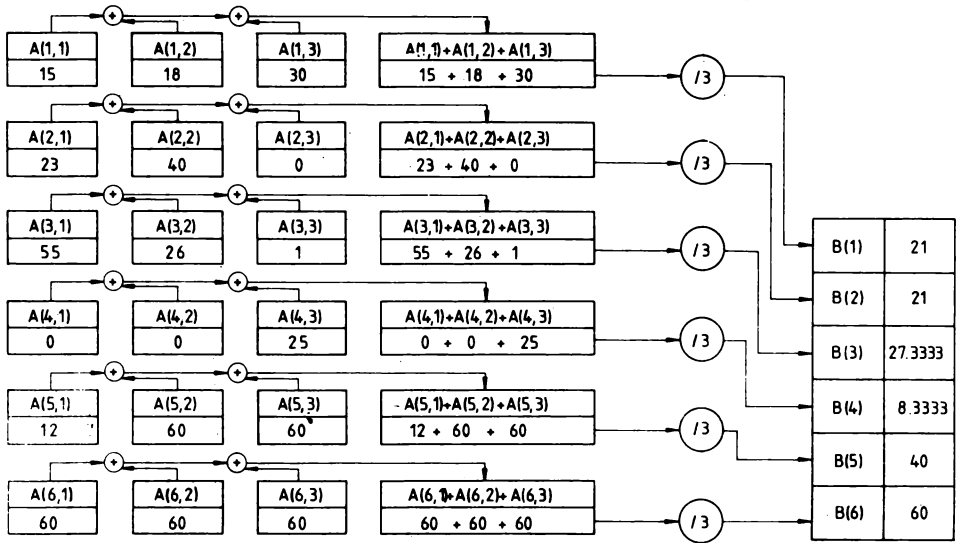


Fig. 6.7.

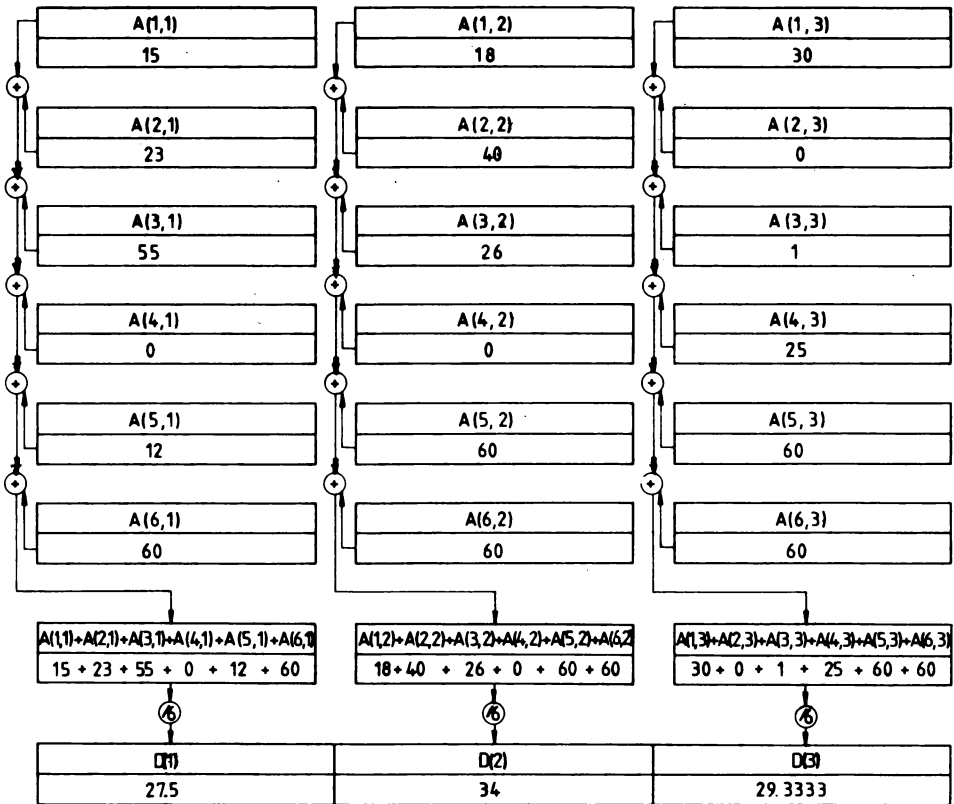


Fig. 6.8.

Cu secvența de instrucțiuni:

```

800 H=A (1,1)
805 R=1
810 C=1
815 FOR Z=1 TO 6
820   FOR I=1 TO 3
825     IF A (Z, I) <= H THEN 845
830     H=A (Z, I)
835     R=Z
840     C=I
845   NEXT I
850 NEXT Z

```

se determină, în variabila H, maximul livrărilor. Valoarea finală a variabilelor R și C indică numărul rezervorului respectiv ziua când s-a atins un maxim al livrărilor.

Codificarea în limbajul BASIC-PRAE

vol. 2, pag. 218

Variabile indexate șir (alfanumerice)

Numele unei variabile șir (alfanumerice) se definește la fel ca și numele unei variabile numerice cu deosebirea că la o variabilă șir se pune după acest nume semnul dolarului. Variabilele șir ocupă loc imediat după variabilele numerice. Pe doi octeți se scrie din nou numele variabilei, urmat nu de valoarea efectivă a acestei variabile ci de un așa-zis „pointer” (arătător) care ne indică unde anume se găsește această valoare. De notat că, valoarea unei variabile șir se poate găsi în program sau într-o zonă a memoriei numită zona de manevră pentru șiruri. Limbajul BASIC-PRAE permite definirea variabilelor indexate de tip șir pentru prelucrarea vectorilor și masivelor de șiruri de caractere.

Reguli

- Variabila indexată de tip șir constă dintr-un nume de variabilă BASIC-PRAE, caracterul \$ și unul sau doi indici între paranteze;
- Limitele între care pot varia indicii unei variabile indexate de tip șir pot fi declarate: a) implicit (prin prima referință); b) explicit (prin instrucțiunea DIM);
- Expresiile de indici nu pot fi decât expresii aritmetice;
- Este posibilă utilizarea simultană a unei variabile simple șir și a unei variabile indexate șir cu același nume;
- Variabilele de tip șir sînt inițializate la execuția unui program cu valori egale cu „șirul vid” (șirul de caractere avînd lungimea zero);
- Variabilele șir nu trebuie declarate într-o instrucțiune DIM;
- Este posibilă utilizarea unor subșiruri dintr-un șir desemnat de o variabilă (v. limbajul BASIC-aMIC).

Aplicație. Introduceți și executați următorul program:

```

10 INPUT "NUME"; N$
20 FOR I=1 TO 3

```

```

30   READ A$, B$
35   IF N$=A$ THEN 70
40   NEXT I
50   PRINT "NUME NECUNOSCU"
60   STOP
70   PRINT A$, B$
80   DATA ZAMFIRESCU, COLENTINA 12-BUCUREȘTI
90   DATA IONESCU, PARIS 13-BUCUREȘTI
100  DATA POPA, ROMA 10-BUCUREȘTI
120  END

```

PRINT AT

După cum am văzut în conversațiile de pînă acum, instrucțiunea **PRINT** afișează întotdeauna rezultatele pe ecran începînd din ultimul rînd și defilînd apoi ecranul. Instrucțiunea **PRINT AT** ne oferă posibilitatea de a alege punctul de pe ecran în care să se afișeze rezultate, mesaje etc.

Formatul general al instrucțiunii este

Format general
PRINT AT <x>, <y>; <lista>

în care:

<x>, <y> reprezintă coordonatele punctului de pe ecran de unde dorim să scriem; <lista> are aceeași semnificație ca la **PRINT**.

Remarci

- Ecranul TV, din punctul de vedere al PRAE-ului este împărțit în 256 de puncte pe orizontală și 256 de puncte pe verticală, deci un total de $256 \times 256 = 65536$ pixeli (pictural elements).
- Punctul de coordonate (0,0) este stabilit în colțul din stînga sus al ecranului.
- BASIC-PRAE acceptă ca limite pentru <x> și <y> domeniul de valori $0 \div 255$.
- Coordonatele x și y nu se introduc între paranteze;
- <x> și <y> pot fi expresii numerice.
- Deoarece pe ecran un caracter ocupă un cîmp de 8×8 puncte, rezultatelor expresiilor x și y li se va considera partea întreagă, și eventual vor fi micșorate pentru a obține un număr multiplu de opt.

Înapoi la program

Diferențele de scriere a programului apar în instrucțiunile de citire ale matricei de date A (limbajul BASIC-PRAE nu acceptă instrucțiuni matriciale!) cit și în modul de utilizare a instrucțiunii **PRINT AT** (v. dimensiunile lui x și y).

```

100  FOR Z=1 TO 6
110  PRINT, C$(Z); " : "
120  FOR I=1 TO 3
130  PRINT "R";
135  PRINT USING "#"; I;
138  PRINT " ";
140  INPUT A(Z, I)
150  NEXT I
160  NEXT Z

```

Observație. Remarcați în linia 135 utilizarea formatului de editare "#" pentru variabila I (indexul rezervorului). În limbajul BASIC-PRAE instrucțiunea **PRINT USING** admite specificații de format atât pentru numere cât și pentru șiruri. Simbolurile folosite pentru editarea valorilor numerice sint: "#", "-", " ", " ", "*", ".", "^". Ele au aceeași semnificație ca și în limbajul BASIC-PLUS, BASIC-80.

În cadrul programului am preferat scrierii cu **PRINT AT** scrierea cu **PRINT TAB** în care am precizat doar coloana în care urmează să se afișeze rezultatele (v. liniile 515, 535, 710).

Remarcați de asemenea în linia 5 a programului dimensionarea matricei A (6×3 elemente).

Aplicație. Introduceți și executați următoarele programe:

a)

```
10 CLS
20 PRINT "COD"; "CANTITATE"; "PREȚ"
30 FOR I=1 TO 4
40 READ COD, CANTITATE, PREȚ
50 PRINT COD; CANTITATE; PREȚ
60 NEXT I
70 END
80 DATA 10, 3, 8
90 DATA 20, 4, 7
100 DATA 30, 6, 9
110 DATA 40, 5, 6
120 END
```

b) Modificați programul precedent de maniera:

```
20 PRINT "COD"; "CANTITATE"; "PREȚ"; "VALOARE"
50 PRINT COD; CANTITATE; PREȚ; CANTITATE*PREȚ
```

după care executați programul.

c)

```
10 PRINT "COD"; "DENUMIRE"; "CANTITATE"; "PREȚ"; "VALOARE"
20 FOR I=1 TO 4
30 READ COD, DENUMIRE$, CANTITATE, PREȚ
40 PRINT COD; DENUMIRE$; CANTITATE; PREȚ; CANTITATE * PREȚ
50 NEXT I
60 DATA 10, A1-B, 3, 82, 20, A2B, 4,7.50,30, A3-B, 6, 9.25, 40, A4-B, 5, 6.30
70 END
```

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 219

Variabile indexate șir (alfanumerice)

În limbajul BASIC HC-85, TIM S, SPECTRUM variabilele șir și variabilele indexate șir prezintă următoarele particularități:

- numele variabilelor șir și al variabilelor indexate șir este format dintr-o literă urmată de semnul dolar;
- variabilele indexate și variabilele șir pot avea lungimi și număr de dimensiuni de lungime arbitrară;
- variabilele șir și variabilele indexate șir nu pot avea același nume;
- rezervarea de spațiu pentru o variabilă indexată șir se face prin utilizarea instrucțiunii **DIM**;
- variabilele șir sint memorate pe un număr de:

3+(lungimea șirului)

octeți;

- variabilele indexate șir sînt memorate pe un număr de $4+2*(\text{numărul dimensiunilor})+(\text{numărul total de caractere})$ octeți;
- toate variabilele indexate șir vor avea aceeași lungime dată de ultima dimensiune **DIM**;
- la memorarea elementelor unui masiv (matrice) indicele dimensiunii cea mai din stînga variază cel mai încet;
- apelarea elementelor unui șir se face prin specificarea caracterului din șir, sau a poziției inițiale și finale a subșirului ce se apelează;
- este posibilă utilizarea unor subșiruri dintr-un șir desemnat de-o variabilă (v. limbajul BASIC-aMIC).

PRINT AT

În instrucțiunea **PRINT** din limbajul BASIC HC-85, TIM S, SPECTRUM se pot specifica linia (0–21) și coloana (0–31) de unde dorim să se facă afișarea valorilor, folosind instrucțiunea **PRINT AT** (v. limbajul BASIC-PRAE).

Reguli

- BASIC HC-85, TIM S, SPECTRUM acceptă ca limite pentru x și y domeniul de valori 0–21 (x), 0–31 (y).
- x și y pot fi expresii numerice.
- Ecranul alfanumeric conține 24 de linii a câte 32 coloane (caractere).
- Liniile 22 și 23 sînt rezervate mesajelor de sistem.
- Caracterul de adresă alfanumerică (0, 0) este fixat în colțul din stînga sus al ecranului.

Înapoi la program

În comparație cu programul BASIC-aMIC diferențele de scriere apar în instrucțiunea **DATA** (constantele șir se pun între ghilimele).

```
15 DATA "LUNI", "MARȚI", "MIERCURI", "JOI", "VINERI", "SÎMBĂTĂ"
```

și în instrucțiunile de citire a matricei de date **A**.

```
100 FOR Z=1 TO 6
110 PRINT CS(Z); " : "
115 FOR I=1 TO 3
120 PRINT "A("; Z; ", "; I; ")=";
125 INPUT A(Z,I)
130 NEXT I
140 PRINT
150 NEXT Z
```

Observație. În BASIC HC-85, TIM S, SPECTRUM lipsesc instrucțiunile matriciale.

Deoarece ecranul alfanumeric conține numai (0–21) linii operaționale modificate liniile 855, 860; 865 ale programului BASIC-aMIC după cum urmează:

```
855 PRINT AT (19,1); "MAXIMUL LIVRĂRILOR="; H; "TONE"
860 PRINT AT (20,4); "REZERVOR=R"; C
865 PRINT AT (21,4); "ZIUA="; CS(R)
900 END
```

Aplicație. Introduceți în program o secvență de instrucțiuni care să afișeze la începutul programului mesajul:

EXEMPLUL 6

scris cu caractere semigrafice (v. conversația 12).

Numere aleatoare

Numerele aleatoare reprezintă "inima" multor jocuri logice și programe destinate, în special microcalculatoarelor și calculatoarelor personale.

Dumneavoastră vă puteți întâlni cu numerele aleatoare atunci când veți juca jocuri ce implică aruncarea unui zar, aruncarea cu banul sau extragerea unor numere din pălărie.

Deși toți avem o idee despre ce înseamnă o succesiune de numere aleatoare este totuși dificil să definim clar noțiunea.

Pentru a clarifica totuși noțiunea vă propunem să aruncăm o privire asupra numerelor obținute în urma aruncării unui zar (cu șase fețe) de 15 ori în două experimente "gândite".

Să presupunem că în primul experiment au ieșit numerele:

1, 5, 2, 4, 6, 3, 2, 1, 6, 3, 5, 4, 3, 4, 2

iar în cel de-al doilea experiment s-au obținut numerele:

6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6.

Dacă succesiunea de numere din primul experiment s-ar fi produs "la noroc" cit privește cel de-al doilea experiment începem să devenim suspicioși și să dezaprobăm zarul. Cu alte cuvinte, la sfârșitul unei succesiuni mai lungi de numere ne așteptăm să aplicăm "legile norocului"

Aplicație. Introduceți și executați următorul program:

10 FOR I=1 TO 5	RUN
20 LET X=RND	0. 85560608
30 PRINT X	0. 98418181
40 NEXT I	0. 71151733
50 END	0. 36412048
	0. 66645813

În mod cert numerele de mai sus arată ca niște numere aleatoare și dacă veți tasta **RUN** veți obține o listă complet diferită.

După cum constatați, funcția **RND** generează numere aleatoare în intervalul 0-1 cu 0 inclusiv dar cu 1 exclusiv.

Cum putem să mărim intervalul pentru a obține și alte numere aleatoare?

Foarte simplu, multiplicând **RND** cu un număr.

Astfel,

RND generează un număr real aleator în intervalul 0,1;

5*RND generează un număr real aleator în intervalul 0,6;

52*RND generează un număr real aleator în intervalul 0,52.

Vă puteți gândi la **RND** ca la un "factor de conversie" care se schimbă după bunul său plac. Următorul program ilustrează această idee.

```

10 PRINT "i"; TAB 8; "1*RND"; TAB 20; "6*RND"
20 PRINT "----"; TAB 8; "-----"; TAB 20; "-----"
30 FOR I=1 TO 5
40   LET A=RND
50   LET S=S*A
60   PRINT i; TAB 8; A; TAB 20; S
70 NEXT I
80 STOP
RUN

```

i	1 * R N D	6 * R N D
1	0.62315369	3.7389221
2	0.73695374	4.4217224
3	0.27182007	1.6309204
4	0.3873291	2.3239746
5	0.87016296	5.2209778

Executați programul înlocuind linia 50 cu:

```

50 LET S=52*A
RUN

```

i	1 * R N D	52 * R N D
1	0.65000916	33.800476
2	0.75108337	39.056335
3	0.33152771	17.239441
4	0.86534119	44.997742
5	0.90074158	46.838562

TEST

Scrieți un program care să afișeze șase numere aleatoare în intervalul $0 \div 5.999999$.

```

10 FOR A=1 TO 6
20   LET X=6*RND
30   PRINT X
40 NEXT A
50 STOP

```

Aplicație. Introduceți și executați următoarele programe:

a)

```

10 DIM A(4)
20 FOR I=1 TO 4
30   LET A(I)=INT (RND*8)+1
40 NEXT I
50 FOR I=1 TO 4
60   PRINT TAB 6; "A("; I;")="
70 NEXT I

```

b)

```

10 DIM A(4,4)
20 FOR I=1 TO 4
30   FOR J=1 TO 4
40     LET A (I, J)=INT (RND * 8+1)
50     PRINT "A("; I; ", "; J; ")="; A (I, J)
60   NEXT J
70 NEXT I
80 PRINT AT 16, 15; "1 2 3 4"
90 PRINT
100 FOR I=1 TO 4
110   PRINT TAB 13; I; TAB 15; A (I, 1); " ";
      A (I, 2); " "; A (I, 3); " "; A (I, 4)
120 NEXT I

```

- ```

c) 10 DIM A (3, 3, 3)
 20 FOR I=1 TO 3
 30 FOR J=1 TO 3
 40 FOR K=1 TO 3
 50 LET A (I, J, K)=INT (RND*4)+1
 60 PRINT "A("; I; ", "; J; ", "; K; ")="; A (I, J, K)
 70 NEXT K
 80 NEXT J
 90 NEXT I
 100 STOP

d) 10 DIM A (2, 2, 2, 2, 2)
 20 FOR I=1 TO 2
 30 FOR J=1 TO 2
 40 FOR K=1 TO 2
 50 FOR L=1 TO 2
 60 PRINT "A ("; I; ", "; J; ", "; K; ", "; L; ", "; M; ")="; A (I, J, K, L, M)
 70 FOR M=1 TO 2
 80 LET A (I, J, K, L, M)=INT (RND*9)+1
 90 NEXT M
 100 NEXT L
 110 NEXT K
 120 NEXT J
 130 NEXT I
 140 END

e) 10 DIM A$ (4, 10)
 20 FOR I=1 TO 4
 30 READ B$ (I)
 40 NEXT I
 50 FOR I=1 TO 4
 60 PRINT "A$ ("; I; ")="; A$ (I)
 70 NEXT I
 80 DATA "ION", "PAUL", "AGA", "ADA"
 90 STOP

f) 10 DIM B$ (21, 5)
 20 FOR I=1 TO 21
 30 READ A$ (I)
 40 IF 3*INT (I/3)=I THEN RESTORE
 50 NEXT I
 60 FOR I=21 TO 1 STEP-1
 70 PRINT B$ (I); " ";
 80 NEXT I
 90 DATA "PAUL", "DANI", "ADA"

g) 10 DATA "PI", 7
 20 DIM B$ (21, 4)
 30 DIM Z (21)
 40 FOR I=1 TO 21
 50 READ B$ (I), Z (I)
 60 IF 3*INT (I/3)=I THEN RESTORE
 65 DATA "DANI"
 70 NEXT I
 80 DATA 23, "DAN"
 90 FOR I=1 TO 21
 100 PRINT B$ (I), Z (I)
 110 DATA 11
 120 NEXT I

```

*Observație.* Instrucțiunea **RESTORE** (v. linia 60) face ca instrucțiunea **READ** următoare să citească datele de la o instrucțiune **DATA** aflată la linia (n) sau după aceasta. Dacă (n) lipsește (cazul nostru) se ia ca valoare implicită 1.

### Joc pe calculator

Scrieți un program care simulează aruncarea unui zar de 25 ori.

|                       |            |
|-----------------------|------------|
| 10 FOR I=1 TO 5       | <b>RUN</b> |
| 20 FOR J=1 TO 5       | 2 5 3 1 3  |
| 30 LET K=INT(6*RND+1) | 3 4 5 5 2  |
| 40 PRINT K; " "       | 4 3 5 4 1  |
| 50 NEXT J             | 4 5 4 4 6  |
| 60 PRINT              | 1 6 3 4 2  |
| 70 PRINT              |            |
| 80 NEXT I             |            |
| 90 STOP               |            |

În linia 30 a programului am folosit instrucțiunea

```
30 LET K=INT (6*RND+1)
```

care are ca efect generarea numerelor întregi de la 1 la 6 inclusiv.

## □ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 220

Diferențele de scriere (v. specificațiile de programare și documentația de proiectare) apar în:

### a) Modul de introducere a matricei de date A:

```
50 FOR Z=1 TO 6
51 PRINT C$ (Z); " : "
55 S=0
60 FOR I=1 TO 3
61 PRINT "R"; : PRINT USING "#"; I; : PRINT " "; : INPUT A(Z,I)
65 S=S+A (Z, I)
70 NEXT I
75 B(Z)=S/3
80 NEXT Z
```

Remarcați că spre deosebire de programele precedente citirea datelor (v. formatul de editare "#")

```
61 PRINT "R"; : PRINT USING "#"; I; : PRINT " "; : INPUT A (Z,I)
```

se realizează în aceeași buclă **FOR-NEXT** (v. liniile 60–70) pe care le și prelucrează.

### b) Modul de afișare a rezultatelor

Sub capul de tabel tipărit cu instrucțiunea:

```
120 PRINT "ZIUA "; " R1", " R2", " R3", "MEDIA"
```

s-au afișat (v. instrucțiunile 140, 159, 160) livrările zilnice și media livrărilor calculată în variabila indexată B(Z).

```
140 FOR Z=1 TO 6
159 PRINT C$(Z), A(Z,1), A(Z,2), A(Z,3), B(Z)
160 NEXT Z
```



Pentru tipărirea mediei livrărilor pe rezervor, calculată în variabila indexată D(I) s-au scris instrucțiunile:

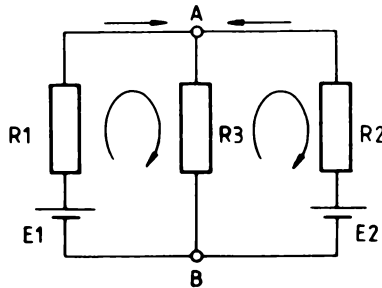
```
164 PRINT "MEDIA ↑ ";
165 FOR I=1 TO 3
167 PRINT D(I),
170 NEXT I
171 PRINT
```

c) Modul de dimensionare a vectorului de date alfanumerice (nu este necesară declararea lungimii șirului de caractere).

```
5 DIM C$(6), B(6), A(6,3), D(3)
```

**Aplicații**

1. Fie rețeaua electrică



în care se cunosc:  $E_1=48\text{ V}$ ;  $E_2=8\text{ V}$ ;  $R_1=2\Omega$ ;  $R_2=3\Omega$ ;  $R_3=2\Omega$ .

Să se determine intensitatea curentului prin fiecare ramură a rețelei.

Aplicind legile lui Kirchoff, rezultă ecuațiile:

$$I_1 + I_2 - I_3 = 0; \tag{5}$$

$$I_1 R_1 + I_3 R_3 = E_1; \tag{6}$$

$$I_3 R_3 + I_2 R_2 = E_2 \tag{7}$$

Pentru rezolvarea sistemului de ecuații (5), (6), (7) aplicăm formulele lui Cramer.

$$I_1 = \frac{\begin{vmatrix} 0 & 1 & -1 \\ 48 & 0 & 2 \\ 8 & 3 & 2 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -1 \\ 2 & 0 & 2 \\ 0 & 3 & 2 \end{vmatrix}} = 14\text{ A}; \tag{8}$$

$$I_2 = \frac{\begin{vmatrix} 1 & 0 & -1 \\ 2 & 48 & 2 \\ 0 & 8 & 2 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -1 \\ 2 & 0 & 2 \\ 0 & 3 & 2 \end{vmatrix}} = -4\text{ A}; \tag{9}$$

$$I_3 = \frac{\begin{vmatrix} 1 & 1 & 0 \\ 2 & 0 & 48 \\ 0 & 3 & 8 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & -1 \\ 2 & 0 & 2 \\ 0 & 3 & 2 \end{vmatrix}} = 10\text{ A}. \tag{10}$$



```

310 PRINT
320 N1=3
330 N2=N1
340 E1=0

```

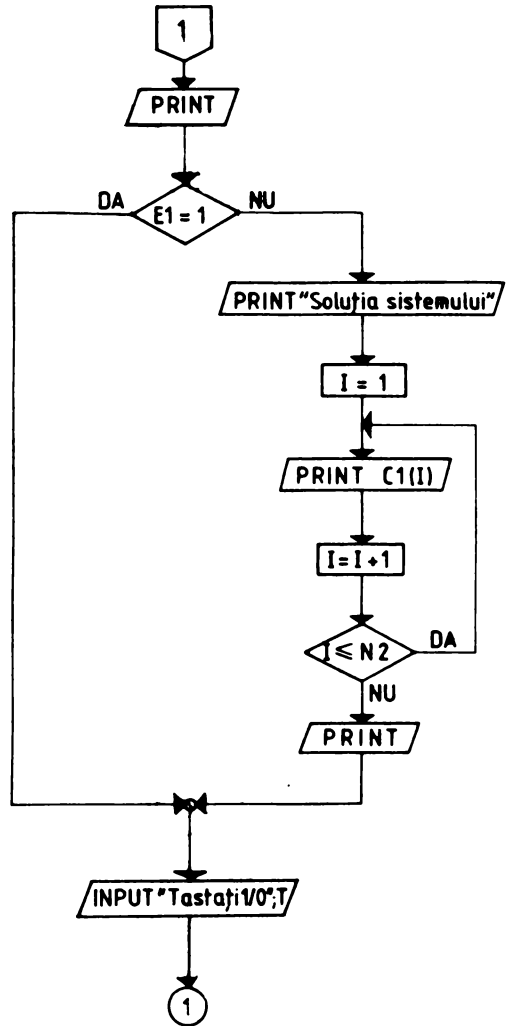
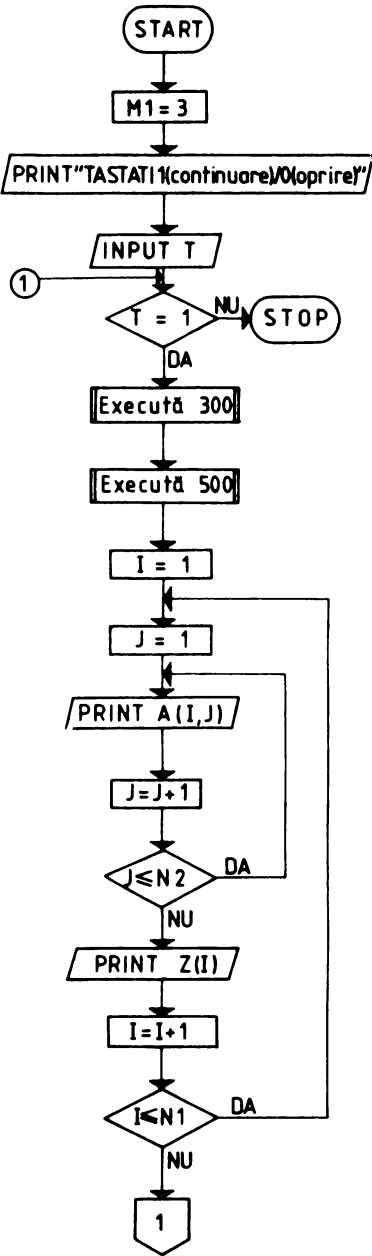


Fig. 6.9

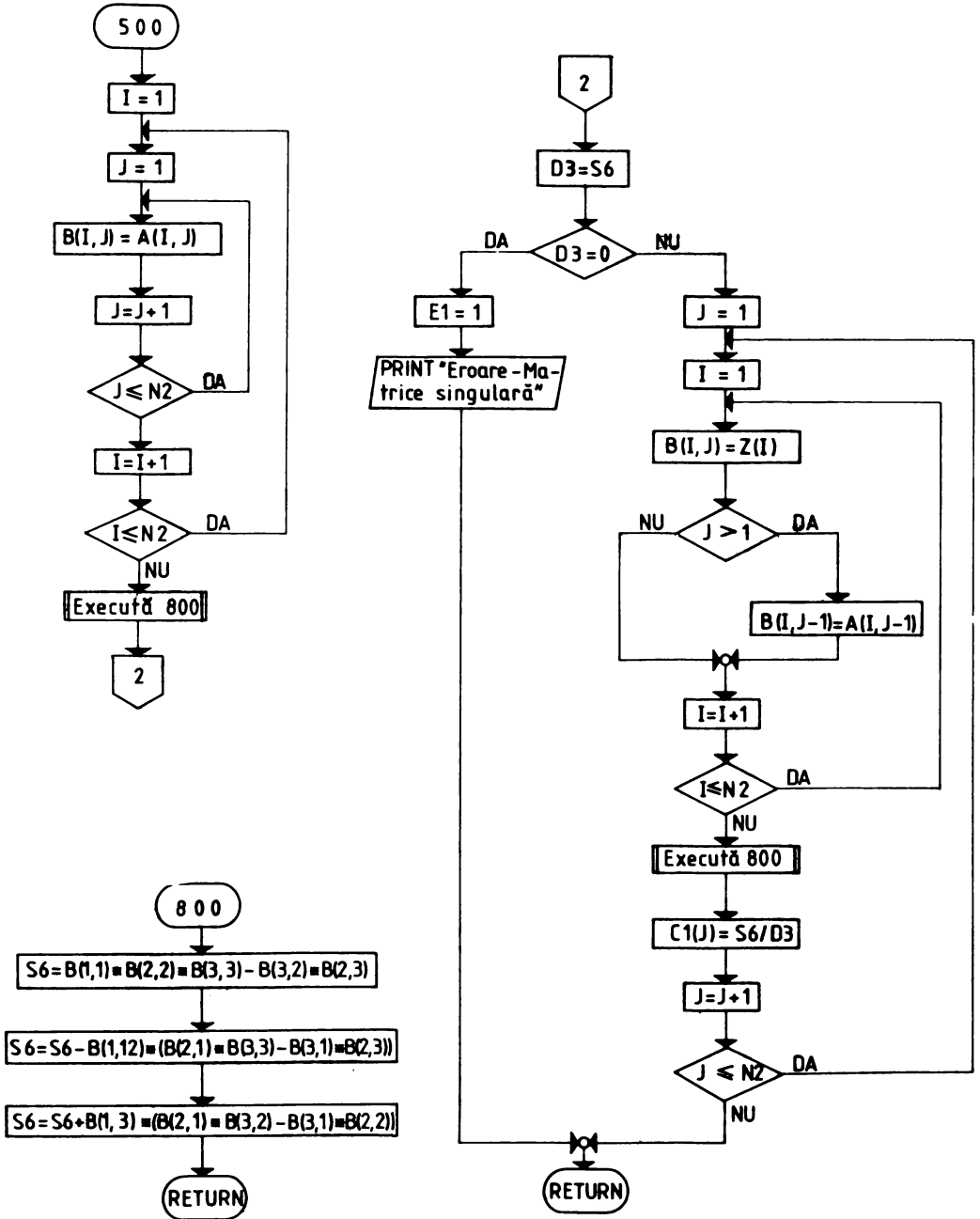


Fig. 6.9

```

350 FOR I=1 TO N1
360 PRINT "Ecuația"; I
370 FOR J=1 TO N1
380 PRINT J; " ";
390 INPUT A (I, J)
400 NEXT J
410 INPUT "C"; Z (I)
420 NEXT I
430 RETURN
500 REM MATRICE 3×3
510 REM A matricea coeficienților
520 REM B matrice de lucru
530 REM C1 vector rezultate
540 REM D3 determinant
550 REM E1 stare eroare
560 REM N2 număr coloane
570 REM S6 sumă
580 REM Z vector termeni liberi
590 REM
600 FOR I=1 TO N2
610 FOR J=1 TO N2
620 B (I, J)=A (I, J)
630 NEXT J
640 NEXT I
650 GOSUB 800
660 D3=S6
670 IF (D3=0) THEN 770
680 FOR J=1 TO N2
690 FOR I=1 TO N2
700 B(I, J)=Z (I)
710 IF (J>1) THEN B (I, J-1)=A(I, J-1)
720 NEXT I
730 GOSUB 800
740 C1(J)=S6/D3
750 NEXT J
760 RETURN
770 E1=1
780 PRINT "EROARE – matrice singulară"
790 RETURN
800 REM Calcul determinant 3×3
810 S6=B(1,1)*(B(2,2)*B(3,3)-B(3,2)*B(2,3))
820 S6=S6-B(1,2)*(B(2,1)*B(3,3)-B(3,1)*B(2,3))
830 S6=S6+B(1,3)*(B(2,1)*B(3,2)-B(3,1)*B(2,2))
840 RETURN
850 END

```

### Remarci

- Simulați cu creionul și hirtia modul de funcționare a programului.
- Introduceți în biblioteca dvs. de programe științifice acest program.

2. Introduceți și executați următoarele programe:

```

a) 10 MODE 1
20 RĂSPUNS=0
30 N1=INT (RND*8+1)
40 N2=INT (RND*8+1)
50 WHILE RĂSPUNS<>N1*N2
60 PRINT N1; "x"; N2; "=";
70 INPUT "RĂSPUNS="; RĂSPUNS
80 WEND
90 END

```

- b) 10 **MODE 1**  
 20 **READ N\$, T\$**  
 30 **WHILE N\$ < > "AAA"**  
 40 **PRINT N\$, T\$**  
 50 **READ N\$, T\$**  
 60 **WEND**  
 70 **DATA IONESCU ION, 812330**  
 80 **DATA ANDREI ILIE, 202015**  
 90 **DATA ADAM VASILE, 803020, AAA, AAA**  
 100 **END**
- c) 10 **MODE 1**  
 20 **WHILE N\$ < > "AAA"**  
 30 **READ N\$, T\$**  
 40 **PRINT N\$, T\$**  
 50 **WEND**  
 60 **DATA IONESCU ION, 812330, ANDREI ILIE, 202015**  
 70 **DATA ADAM VASILE, 803020, AAA, AAA**  
 80 **END**
- d) 10 **A=0**  
 20 **RĂSPUNS=0**  
 30 **N1=INT (RND\*8+1)**  
 40 **N2=INT (RND\*8+1)**  
 50 **WHILE RĂSPUNS < > N1\*N2 AND A < 4**  
 60 **PRINT N1; "x"; N2; "=";**  
 70 **INPUT "RĂSPUNS="; RĂSPUNS**  
 75 **A=A+1**  
 80 **WEND**
- e) 10 **MODE 1**  
 20 **G=0**  
 30 **FOR I=1 TO 12**  
 40 **READ A\$, B\$**  
 50 **PRINT "CÎTE ZILE ARE LUNA"; A\$**  
 60 **INPUT "RĂSPUNS" RĂSPUNS**  
 70 **IF RĂSPUNS < > B\$ THEN G=G+1; PRINT "RĂSPUNS GREȘIT!" ;**  
**PRINT A\$; "ARE"; B\$; "ZILE";**  
 80 **NEXT**  
 90 **PRINT**  
 100 **PRINT "AȚI GREȘIT"; G; "RĂSPUNSURI"**  
 110 **DATA IANUARIE, 31, FEBRUARIE, 28**  
 120 **DATA MARTIE, 31, APRILIE, 30**  
 130 **DATA MAI, 31, IUNIE, 30**  
 140 **DATA IULIE, 31, AUGUST, 31**  
 150 **DATA SEPTEMBRIE, 30, OCTOMBRIE, 31**  
 160 **DATA NOIEMBRIE, 30, DECEMBRIE, 31**  
 170 **END**
- f) 10 **MODE 1**  
 20 **READ întrebare\$, răspuns\$**  
 30 **WHILE întrebare\$ < > "AAA"**  
 40 **PRINT ; PRINT întrebare\$**  
 50 **r\$= ""**  
 60 **incercări=1**  
 70 **WHILE r\$ < > răspuns\$ AND incercări < 4**  
 80 **LINE INPUT r\$ ,**  
 90 **incercări=incercări+1**  
 100 **IF r\$ < > răspuns\$ AND incercări < 4**  
**THEN PRINT "GREȘIT!"**  
 110 **WEND**  
 120 **IF r\$ < > răspuns\$ THEN PRINT "Răspunsul este"; răspuns\$**  
 130 **READ întrebare\$, răspuns\$**

```

140 WEND
150 DATA Ce echipă românească a câștigat CUPA CAMPIONILOR EUROPENI?,
 Steaua
160 DATA Care este antrenorul echipei RAPID, Schimbător
170 AAA, AAA
180 END

```

```

g) 5 WHILE Y$ <> "NU"
 10 DATA 2, 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A
 20 CLS
 30 FOR NUMĂRCĂRȚI=1 TO 5
 40 FOR I=1 TO RND(13) : READ P$: NEXT
 50 PRINT TAB (15); P$; " ";
 60 RESTORE : NEXT
 70 FOR L=1 TO 15 : PRINT : NEXT
 80 INPUT "DORIȚI SĂ JUCAȚI DA/NU"; Y$
 90 WEND

```

## □ Codificarea în limbajul BASIC-COMMODORE

vol. 2, pag. 221

De notat, asemănarea programului cu cel executat pe calculatorul PRAE. Singura deosebire apare la scrierea instrucțiunii din linia 135 (BASIC-COMMODORE nu acceptă **PRINT USING**). Vă recomandăm ca în locul liniilor 130, 135, 138 să se introducă următoarea secvență de instrucțiuni BASIC-COMMODORE:

```

130 PRINT "A (" ; Z; ", " ; ")=" ;
140 INPUT A (Z, I)
142 PRINT

```

De notat că limbajul BASIC-COMMODORE nu acceptă nici instrucțiunile matriciale și nici instrucțiunea **PRINT AT**.

**Aplicație.** Modificați programele precedente astfel încât să poată fi rulate pe un calculator COMMODORE.

### Jocuri pe calculator

Introduceți și executați următoarele programe:

- a) 

```

10 FOR I=1 TO 10
20 N1=INT (RND * 8+1)
30 N2=INT (RND * 8+1)
40 PRINT N1 ; "X" ; N2 ; "=" ;
50 INPUT RĂSPUNS
60 NEXT I
70 END

```
- b) 

```

10 DIM A$(4)
20 FOR I=1 TO 4
30 READ A$ (I)
40 NEXT I
50 FOR I=1 TO 4
60 PRINT "A$ (" ; I ; ")=" ; A$(I)
70 NEXT I
80 DATA UNU, DOI, TREI, PATRU
90 END

```
- c) 

```

10 INPUT "INTRODUCEȚI PRIMUL NUMĂR : " ; X
20 INPUT "INTRODUCEȚI AL DOILEA NUMĂR : " ; Y

```

```

30 PRINT "PRIMUL NUMĂR=" ; X
40 PRINT "AL DOILEA NUMĂR=" ; Y
50 PRINT "ALEGEȚI UNUL DIN OPERATORII: +, -, * /"
60 INPUT "OPȚIUNE: " ; R$
70 PRINT "OPȚIUNE: " ; R$
80 IF R$="+" THEN 140
90 IF R$="-" THEN 160
100 IF R$="*" THEN 180
110 IF R$="/" THEN 200
120 PRINT "OPERATOR INCORECT"
130 GOTO 60
140 R1=X+Y
150 GO TO 250
160 R1=X-Y
170 GOTO 250
180 R1=A * B
190 GOTO 250
200 IF Y=0 THEN 230
210 R1=X/Y
220 GO TO 220
230 PRINT "LA ÎMPĂRȚIRE, Y TREBUIE SĂ FIE DIFERIT DE 0"
240 GO TO 10
250 INPUT "RĂSPUNSUL DVS= " ; R
260 PRINT "RĂSPUNSUL DVS= " ; R
270 IF R1 < > R THEN 300
280 PRINT "FELICITĂRI!"
290 STOP
300 PRINT "RĂSPUNS GREȘIT"
310 GOTO 250
320 END

```

*Observație.* Programul vă ajută să vă dezvoltați calculul aritmetic (adunări, scăderi, înmulțiri, împărțiri) fără creion, gumă sau calculator.

## □ Particularități ale programării în limbajul BASIC-80

vol. 2, pag. 223

De notat (v. vol. 2, pag. 223) asemănarea programului cu cel executat pe calculatorul AMSTRAD. Remarcați și în cadrul acestui program executat pe microcalculatoarele M 118, TPD, JUNIOR: a) modul de dimensionare a vectorului de date alfanumerice C\$ (fără a se fi indicat lungimea elementelor sale); b) utilizarea instrucțiunii **PRINT USING** (v. linia 61).

**Aplicație.** Scrieți un program BASIC-80 care:

a) înmulțește o matrice cu un scalar

```

100 FOR I=1 TO N
110 FOR J=1 TO M
120 Y(I, J)=X(I, J) * S
130 NEXT J
140 NEXT I

```

b) adună două matrice

```

100 FOR I=1 TO N
110 FOR J=1 TO M
120 Z(I, J)=X(I, J)+Y(I, J)
130 NEXT J
140 NEXT I

```



c) înmulțește două matrice:

$$X = \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ x_{31} & x_{32} \end{bmatrix}$$

și

$$Y = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \end{bmatrix}$$

$$Z = \begin{bmatrix} x_{11}y_{11} + x_{12}y_{21} & x_{11}y_{12} + x_{12}y_{22} & x_{11}y_{13} + x_{12}y_{23} \\ x_{21}y_{11} + x_{22}y_{21} & x_{21}y_{12} + x_{22}y_{22} & x_{21}y_{13} + x_{22}y_{23} \\ x_{31}y_{11} + x_{32}y_{21} & x_{31}y_{12} + x_{32}y_{22} & x_{31}y_{13} + x_{32}y_{23} \end{bmatrix}$$

Fiecare element  $j_k$  al lui  $Z$  se formează din linia  $j$  a matricei  $X$  și coloana  $k$  a matricei  $Y$ , conform relației

$$Z_{jk} = X_{j1} Y_{1k} + X_{j2} Y_{2k} + \dots + X_{jn} Y_{nk} \quad (11)$$

Pentru înmulțirea a două matrice vom avea nevoie să înmulțim transpusa unei matrici  $U$  cu matricea originală  $U$ .

De asemenea, va trebui să înmulțim vectorul  $Y$  cu matricea  $U$ .

Programul care urmează efectuează ambele operații. Secvența de instrucțiuni 70÷150 generează matricea  $U$  și vectorul  $Z$ . Secvența de instrucțiuni 160–290 calculează matricea  $A$  și vectorul  $Z$  de care avem nevoie, conform ecuației

$$U^T U = A \quad (12)$$

și

$$Y U = Z \quad (13)$$

În acest caz, matricea  $U$  conține 5 linii și 3 coloane. Vectorul  $Y$  conține cinci elemente.

Programul atribuie lui  $N1$  și  $N2$  valoarea 5 respectiv 3, dimensionează pe  $Z$ ,  $A$ ,  $Y$ ,  $U$  (vezi linia 50) și definește șabloanele de editare  $A\$$  și  $B\$$  din liniile 10 și 20.

Secvența de instrucțiuni 160÷240 merită să fie studiată cu atenție. Ea conține trei bucle nested. Bucila din centru (190÷220) realizează înmulțirea propriu-zisă a matricelor, sau mai exact înmulțirea unei matrici cu transpusa ei.

Buclele 170÷230 și 250÷270 sînt de nivelul al doilea. Linia 170 ține cont că matricea  $A$  va fi simetrică. În acest caz trebuie calculate numai elementele diagonalei de deasupra. Elementele de sub diagonală sînt atribuite de către linia 210. Vor fi termeni ca:

$$A(1, 3) = A(3, 1)$$

În sfîrșit, bucla din liniile 250÷270 realizează înmulțirea  $Y U$ .

Remarcați linia 180:

$$180 \quad A(K, L) = 0$$

care inițializează cu zero fiecare element al lui  $A$  înainte de a începe operația de adunare din linia 200. Matricea  $A$  va avea trei linii (la fel ca transpusa matricii  $U$ ) și trei coloane (la fel ca matricea  $U$ ). Vectorul  $Z$  va avea lungimea 3. De fapt, atît  $Y$  cît și  $Z$  trebuie considerați ca vectori linie. În mod alternativ, dacă vrem să considerăm vectorii  $Y$  și  $Z$  ca vectori coloană, atunci va trebui să scriem ecuația de înmulțire ca:

$$Y^T U = Z^T$$

unde vectorii-coloană transpuși devin vectori linie.

```

10 A$="#####"
20 B$="#####"
30 N1=5
40 N2=3
50 DIM Z(3), A(3, 3), Y(5), U(5, 3)
60 REM Citire date
70 PRINT
80 FOR I=1 TO N1
90 U(I, 1)=1

```

```

100 FOR J=2 TO N2
110 U(I,J)=I * U(I,J-1)
120 NEXT J
130 Y(I)=2 * I
140 NEXT I
150 REM
160 FOR K=1 TO N2
170 FOR L=1 TO K
180 A(K, L)=0
190 FOR I=1 TO N1
200 A(K, L)=A(K, L)+U(I, L) * U(I, K)
210 IF (K <> L) THEN A(L, K)=A(K, L)
220 NEXT I
230 NEXT L
240 Z(K)=0
250 FOR I=1 TO N1
260 Z(K)=Z(K)+Y(I) * U(I, K)
270 NEXT I
280 NEXT K
290 PRINT
300 PRINT " U Y"
310 FOR I=1 TO N1
320 FOR J=1 TO N2
330 PRINT USING A$; U(I%0, J%0);
340 NEXT J
350 PRINT USING B$; Y(I)
360 NEXT I
370 PRINT
380 PRINT " A Z"
390 FOR I=1 TO N2
400 FOR J=1 TO N2
410 PRINT USING A$; A(I,J)
420 NEXT J
430 PRINT USING B$; Z(I)
440 NEXT I
450 PRINT
460 END

```

d) calculează determinantul unei matrice 3 pe 3.

```

10 REM DETERMINANTUL UNEI MATRICE 3X3
20 REM N1 număr linii
30 REM N2 număr coloane
40 REM
50 A$="###.###"
60 N1=3
70 N2=3
80 DIM B(3,3)
90 REM Introducere date
100 PRINT
110 FOR I=1 TO N1
120 PRINT "LINIA" ; I
130 FOR J=1 TO N2
140 PRINT J ; " " ;
150 INPUT B(I, J)
160 NEXT J
170 NEXT I
180 REM CALCUL DETERMINANT - REGULA LUI CRAMER
190 S=B(1,1) * (B(2,2) * B(3,3)-B(3,2) * B(2,3))
200 S=S-B(1,2) * (B(2,1) * B(3,3)-B(3,1) * B(2,3))
210 S=S+B(1,3) * (B(2,1) * B(3,2)-B(3,1) * B(2,2))
220 REM
230 PRINT
240 FOR I=1 TO N1

```

```

250 FOR J=1 TO N2
260 PRINT USING A$; B(I,J) ;
270 NEXT J
275 PRINT
280 NEXT I
290 PRINT
300 PRINT "DETERMINANTUL = " ; S
310 PRINT
320 END

```

### Definirea unei funcții scrise de utilizator

În foarte multe aplicații intervin funcții cu unul sau mai mulți parametri. Limbajul BASIC-80 oferă utilizatorului facilitatea definirii acestor funcții în cadrul programului cu instrucțiunea **DEF FN**. Formatul general al instrucțiunii este:

---

Format general

---

**DEF FN** <nume> [(listă-parametri)] = <def-funcție>

---

unde:

<nume> este numele unei variabile BASIC-80;  
 <listă-parametri> conține variabilele utilizate la definirea funcției (parametrii trebuie separați prin virgule);  
 <def-funcție> este o expresie.

**Aplicație.** Să se calculeze

$$\int_1^8 \frac{dx}{x}$$

prin metoda trapezelor. Se va utiliza subrutina TRAPEZ de integrare prin metoda trapezelor.

La proiectarea și realizarea programului se vor avea în vedere următoarele funcțiuni:

1. Citirea/afișarea datelor de intrare
2. Apelarea subrutinei TRAPEZ (500)
3. Calculul integralei
4. Imprimarea rezultatului

În tabelul 6.5 și tabelul 6.6 sînt ilustrate tabela de variabile și programul sursă al subrutinei.

*Tabelul 6.5*

---

TABELA DE VARIABILE

---

| Variabile de intrare    | Variabile de stare    | Variabile de ieșire     |
|-------------------------|-----------------------|-------------------------|
| A: limita inferioară    | H: pasul de integrare | TI: valoarea integralei |
| B: limita superioară    |                       |                         |
| N: numărul de intervale |                       |                         |

---

Tabelul 6.6

| PROGRAM SURSĂ |                  |
|---------------|------------------|
| 500           | H=(B-A)/N        |
| 510           | Tl=(F(A)+F(B))/2 |
| 520           | FOR K=1 TO N-1   |
| 530           | Tl=Tl+F(A+K * H) |
| 540           | NEXT K           |
| 550           | Tl=Tl * H        |
| 560           | RETURN           |

**Remarcă.** Introduceți în biblioteca dvs. de programe științifice subrutina TRAPEZ.

```

5 DEF FN F(X)=1/X
10 INPUT "Număr intervale" ; N
20 INPUT "Limite integrare" ; A, B
30 GOSUB 500
40 PRINT "Aria="; Tl
50 STOP
500 (v. subrutina TRAPEZ)
.
.
.
560
570 END

```

#### Aplicații

1. Introduceți și executați următorul program:

|    |                     |            |
|----|---------------------|------------|
| 10 | <b>RANDOMIZE</b>    | <b>RUN</b> |
| 20 | <b>FOR I=1 TO 4</b> | 0. 5199228 |
| 30 | <b>PRINT RND</b>    | 0. 3434087 |
| 40 | <b>NEXT I</b>       | 0. 2987809 |
|    |                     | 0. 7020067 |

**Observație.** Funcția **RANDOMIZE** inițializează generatorul de numere aleatoare. Ea are formatul:

| Format general              |
|-----------------------------|
| <b>RANDOMIZE</b> (expresie) |

Dacă **RANDOMIZE** lipsește, funcția **RND** va da aceeași secvență de numere aleatoare la fiecare execuție a programului.

2. Scrieți un program care să simuleze aruncarea unei monezi de 30 de ori. Programul va contoriza și afișa de câte ori "cade" leul sau stema.

```

5 REM L : LEUL ; S : STEMA 50 IF A=2 THEN L=L+1 ELSE S=S+1
10 L=0 60 NEXT I
20 S=0 70 PRINT TAB 8 ; " L=" ; L; TAB 20 ; "S=" ; S
30 FOR I=1 TO 30 80 END
40 A=INT (2 * RND+1)

```

3. Scrieți un program care să simuleze aruncarea a două zaruri de 80 de ori. Programul va afișa numărul și frecvența de apariție a unei fețe a zarului sub forma unei diagrame.

Exemplu:

| NUMĂR                             | FRECVENȚA | DIAGRAMA |
|-----------------------------------|-----------|----------|
| 2                                 | 2         | **       |
| 12                                | 4         | ****     |
| .                                 | .         | .        |
| .                                 | .         | .        |
| 10 DIM S(15)                      |           |          |
| 20 FOR I=2 TO 12                  |           |          |
| 30 S(I)=0                         |           |          |
| 40 NEXT I                         |           |          |
| 50 FOR I=1 TO 80                  |           |          |
| 60 S1=INT(6 * RND+1)              |           |          |
| 70 S2=INT(6 * RND+1)              |           |          |
| 80 S=S1+S2                        |           |          |
| 90 S(S)=S(S)+1                    |           |          |
| 100 NEXT I                        |           |          |
| 110 PRINT                         |           |          |
| 120 FOR I=2 TO 12                 |           |          |
| 130 PRINT I; TAB 5; S(I); TAB 10; |           |          |
| 140 IF S(I)=0 THEN 190            |           |          |
| 150 FOR J=1 TO S(I)               |           |          |
| 160 PRINT " * " ;                 |           |          |
| 170 NEXT J                        |           |          |
| 180 PRINT                         |           |          |
| 190 NEXT I                        |           |          |
| 200 STOP                          |           |          |

4. Să se rezolve sistemul de ecuații (v. aplicație, pag. 319) prin metoda eliminării a lui Gauss.

Tabela de variabile și schema logică ale programului (subrutinele 500; 5000 sint prezentate în tabelul 6.7 și figura 6.10).

Tabelul 6.7

| TABELA DE VARIABILE        |                        |                           |
|----------------------------|------------------------|---------------------------|
| Variabile de intrare       | Variabile de stare     | Variabile de ieșire       |
| N1: număr linii (ecuații)  | E1: stare              | C1: vectorul rezultatelor |
| N2: număr coloane          | M1: lungimea maximă    |                           |
| A: matricea coeficienților | B: matrice de lucru    |                           |
| Z: vector termeni liberi   | B1: matrice de lucru   |                           |
|                            | H1: variabila de lucru |                           |
|                            | W1: vector de lucru    |                           |
|                            | S6: suma               |                           |

Programul debutează prin a vă întreba dacă doriți sau nu continuarea. Răspundeți la această întrebare prin a tasta 1 pentru continuare și 0 pentru oprire. Următoarea întrebare se referă la numărul de ecuații ale sistemului. Răspundeți cu un număr între 2 și 8. Coeficienții necunoscutelor și termenii liberi sint introduși pe rînd pentru fiecare ecuație în parte. Introduceți valorile pentru sistemul de ecuații studiat (v. aplicația de la pag. 319) și verificați dacă vectorul soluție este 14, -4, 10.

Programul imprimă un mesaj de eroare atunci cînd matricea coeficienților este singulară:

EROARE – matrice singulară!

Programul poate fi întrerupt în două rînduri: a) tastînd 0 pentru variabila T; b) introducînd o valoare sub 2 pentru numărul de ecuații (N1).

În sfîrșit, programul apelează două subrutine: 500 (introducere date) și 5000 (rezolvarea propriu-zisă a sistemului).

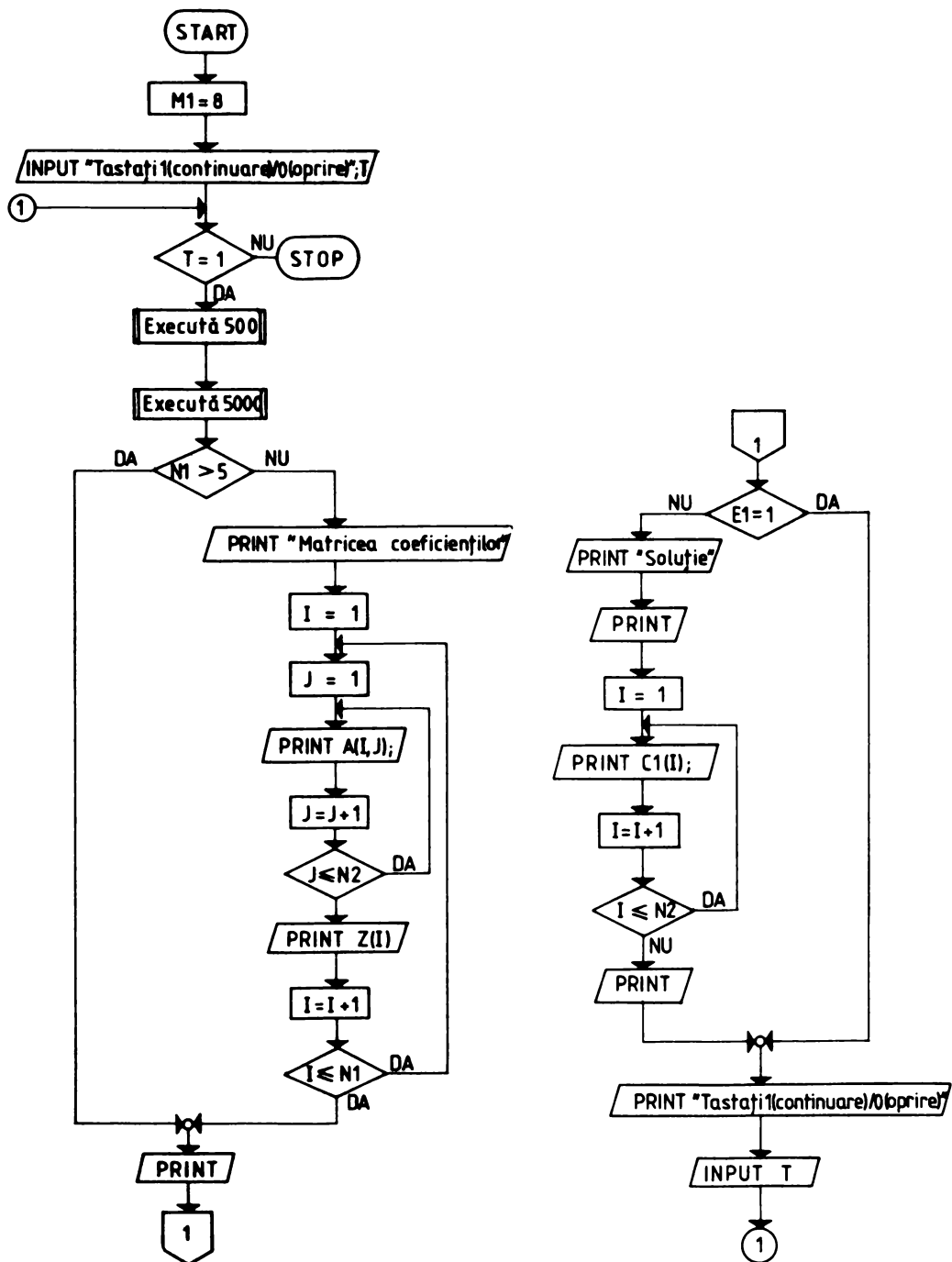


Fig. 6.10.

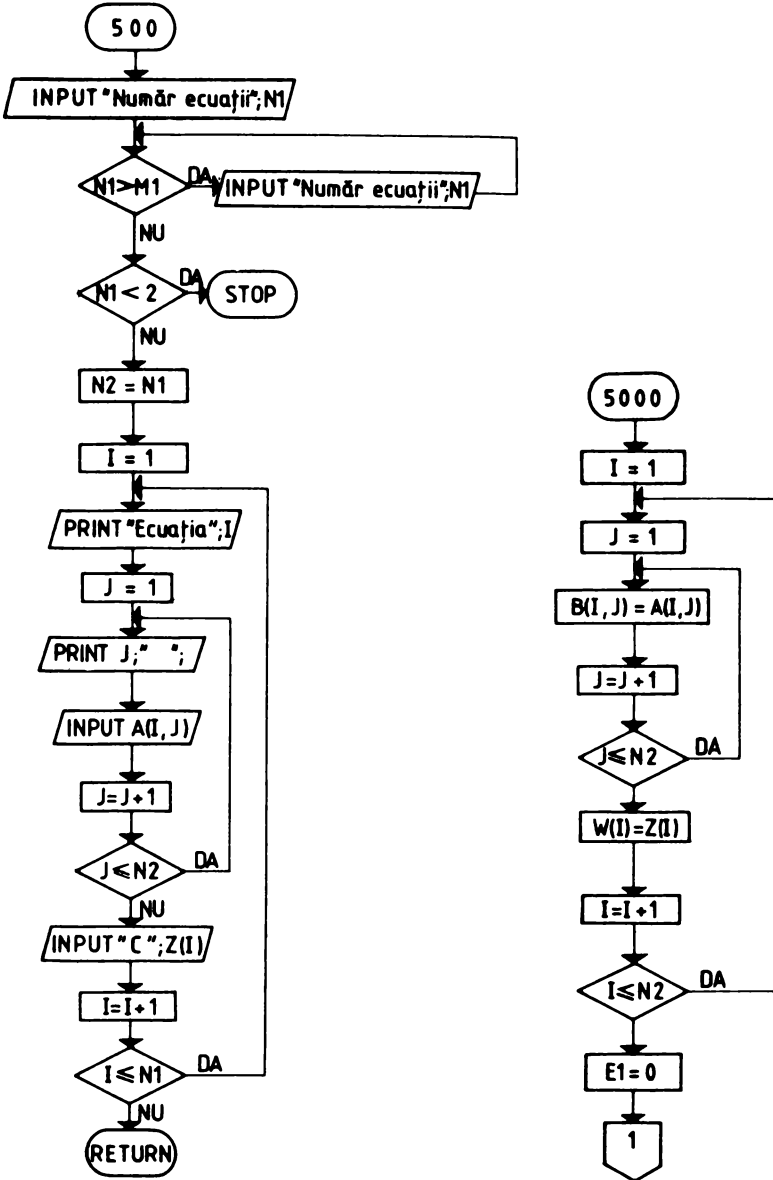


Fig. 6.10

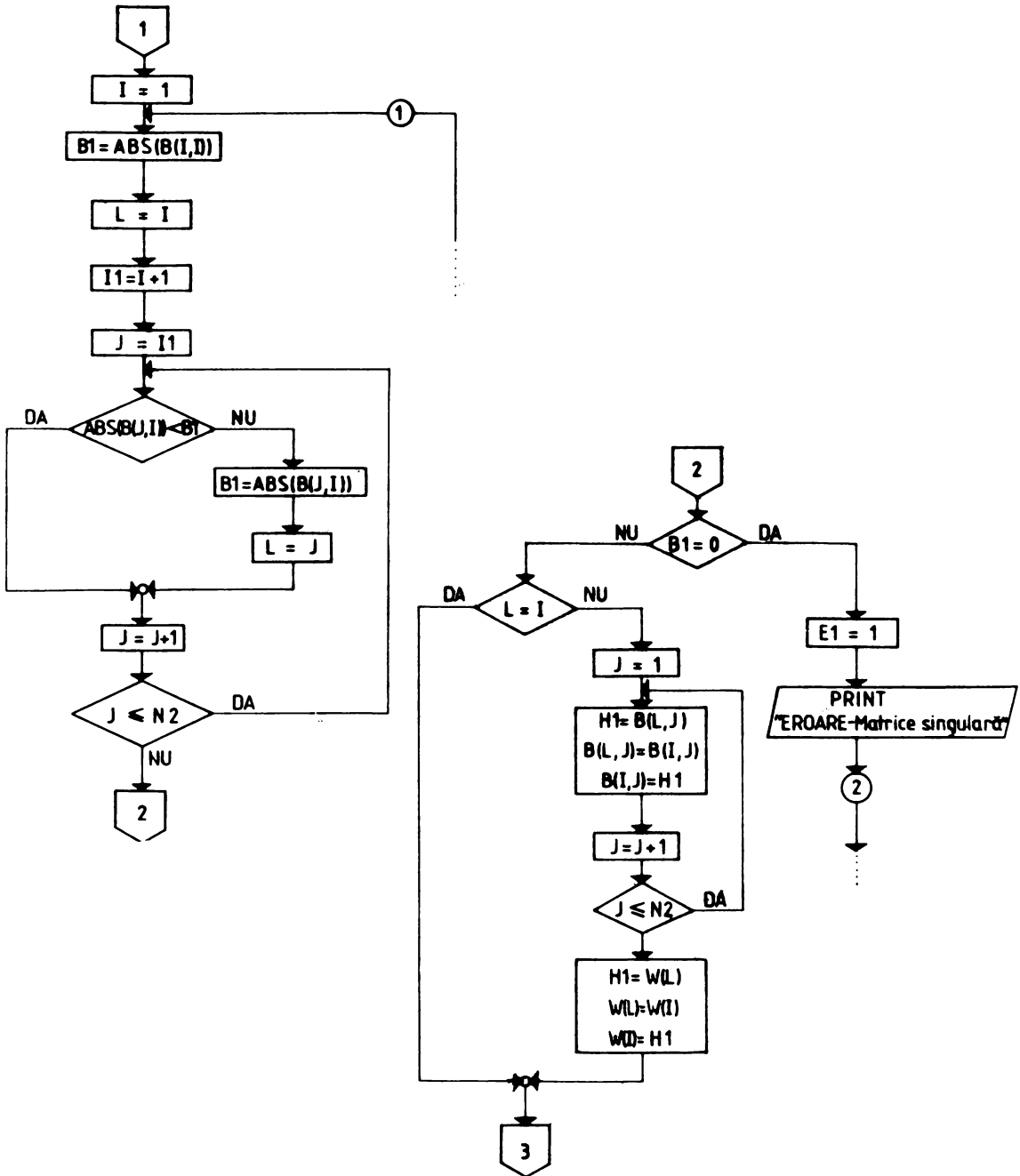


Fig. 6.10



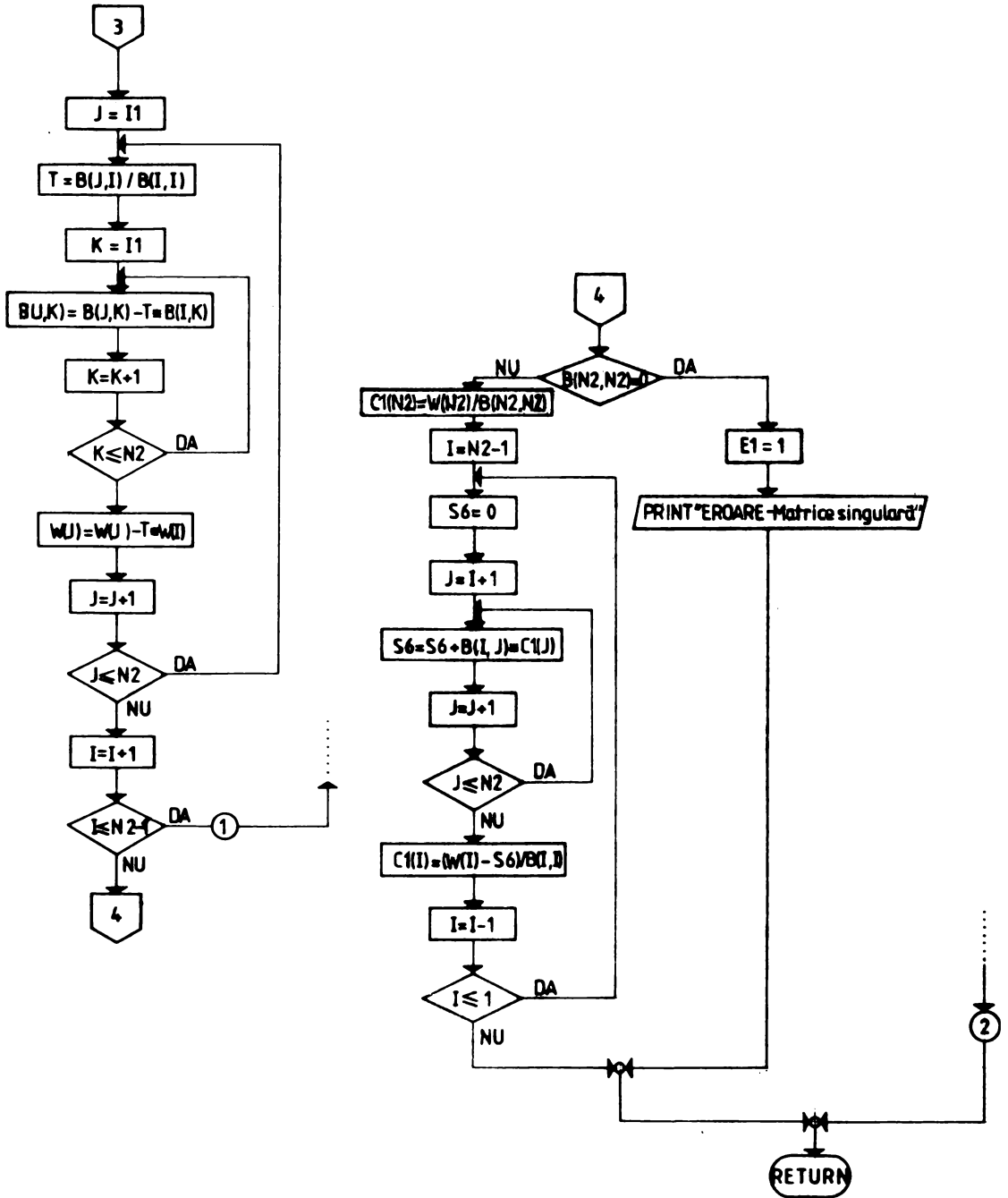


Fig. 6.10

## Remarci

- Simulați cu creionul și hirtia modul de funcționare a acestui program.
- Introduceți în biblioteca de programe științifice a dvs. acest program.

```

3 M1=8
5 INPUT "Tastați 1 (continuare)/0 (oprire)"; T
10 WHILE T=1
15 REM C1 vectorul rezultatelor
16 REM E1 stare eroare
17 REM M1 număr ecuații
18 REM N1 număr linii
19 REM N2 număr coloane
20 A$="# #.# #.# #.# ^^^^"
30 B$="#=# #.# #.# #.# ^^^^"
40 C$="# #.# #.# #.#"
60 DIM Z(8), A(8,8), C1(8), W(8), B(8,8)
100 GOSUB 500
110 GOSUB 5000
120 REM
130 IF (N1>5) THEN 210
140 PRINT "Matricea coeficienților"
150 FOR I=1 TO N1
160 FOR J=1 TO N2
170 PRINT USING A$; A(I, J);
180 NEXT J
190 PRINT USING B$; Z(I)
200 NEXT I
210 PRINT
220 IF (E1=1) THEN 290
230 PRINT "Soluție"
240 PRINT
250 FOR I=1 TO N2
260 PRINT USING C$; C1(I);
270 NEXT I
280 PRINT
290 INPUT "Tastați 1 (continuare)/0 (oprire)"; T
300 WEND
310 STOP
500 REM
530 PRINT
540 INPUT "Număr ecuații"; N1
550 WHILE N1>M1
555 INPUT "Număr ecuații" ; N1
558 WEND
560 IF (N1<2) THEN 5580
570 N2=N1
580 FOR I=1 TO N1
590 PRINT "Ecuația" ; I
600 FOR J=1 TO N2
610 PRINT J ; " " ;
620 INPUT A(I, J)
630 NEXT J
640 INPUT "C"; Z(I)
650 NEXT I
660 RETURN
5000 REM
5010 REM A matricea coeficienților
5020 REM B matricea de lucru
5030 REM B1
5040 REM C1 vector rezultate
5050 REM E1 stare sistem
5060 REM H1
5070 REM N2 număr coloane
5080 REM S6

```

```

5090 REM Z vector termeni liberi
5100 FOR I=1 TO N2
5110 FOR J=1 TO N2
5120 B(I, J)=A(I, J)
5130 NEXT J
5140 W(I)=Z(I)
5150 NEXT I
5160 E1=0
5170 FOR I=1 TO N2-1
5180 B1=ABS (B(I, I))
5190 L=I
5200 I1=I+1
5210 FOR J=I1 TO N2
5220 IF (ABS (B(J, I)) < B1) THEN 5250
5230 B1=ABS (B(J, I))
5240 L=J
5250 NEXT J
5260 IF (B1=0) THEN 5550
5270 IF (L=I) THEN 5360
5280 FOR J=1 TO N2
5290 H1=B(L, J)
5300 B(L, J)=B(I, J)
5310 B(I, J)=H1
5320 NEXT J
5330 H1=W(L)
5340 W(L)=W(I)
5350 W(I)=H1
5360 FOR J=I1 TO N2
5370 T=B(J, I)/B(I, I)
5380 FOR K=I1 TO N2
5390 B(J, K)=B(J, K)-T * B(I, K)
5400 NEXT K
5410 W(J)=W(J)-T * W(I)
5420 NEXT J
5430 NEXT I
5440 IF (B (N2, N2)=0) THEN 5550
5450 C1 (N2)=W(N2)/B(N2, N2)
5460 REM substituție înapoi
5470 FOR I=N2-1 TO 1 STEP-1
5480 S6=0
5490 FOR J=I+1 TO N2
5500 S6=S6+B(I, J) * C1(J)
5510 NEXT J
5520 C1(I)=(W(I)-S6)/B(I, I)
5530 NEXT I
5540 RETURN
5550 E1=1
5560 PRINT "EROARE - matrice singulară!"
5570 RETURN
5580 END

```

**Particularități ale programării  
in limbajul BASIC-PLUS**

vol. 2, pag. 224

În comparație cu programul anterior diferențele de scriere a programului BASIC-PLUS apar numai în citirea matricei de date a livrărilor (A):

```

180 FOR Z=1 TO 6
190 PRINT C$(Z) ; " : "

```

```

200 S=0
210 INPUT "REZERVORUL 1"; A(Z, 1)
220 S=S+A(Z,1)
230 INPUT "REZERVORUL 2"; A(Z, 2)
235 S=S+A(Z, 2)
240 INPUT "REZERVORUL 3"; A(Z, 3)
260 S=S+A(Z, 3)
270 PRINT
280 B(Z)=S/3
290 NEXT Z

```

Spre deosebire de programul rulat pe calculatoarele M 118, TPD, JÚNIOR, în programul BASIC-PLUS se folosesc trei instrucțiuni:

```

210 INPUT "REZERVORUL 1"; A(Z, 1)
230 INPUT "REZERVORUL 2"; A(Z, 2)
240 INPUT "REZERVORUL 3"; A(Z, 3)

```

pentru citirea (pe linie) a valorilor matricei A. Remarcați, de asemenea, și maniera de calcul a totalului general livrări:

```

220 S=S+A(Z, 1)
235 S=S+A(Z, 2)
260 S=S+A(Z, 3)

```

În tabelul 6.8 puteți urmări modul de calcul al lui S și al lui B(Z).

Tabelul 6.8

| Nr. linie | Instrucțiune                 | Z | A            | S                                    | B  |
|-----------|------------------------------|---|--------------|--------------------------------------|----|
| 0         | 1                            | 2 | 3            | 4                                    | 5  |
| 200       | S=0                          | 1 | -            | 0                                    | -  |
| 210       | INPUT "REZERVORUL 1"; A(Z,1) | 1 | A(1,1)<br>15 | 0                                    | -  |
| 220       | S=S+A(Z,1)                   | 1 | A(1,1)<br>15 | S+A(1,1)<br>0+15                     | -  |
| 230       | INPUT "REZERVORUL 2"; A(Z,2) | 1 | A(1,2)<br>18 | S+A(1,1)<br>0+15                     | -  |
| 235       | S=S+A(Z,2)                   | 1 | A(1,2)<br>18 | S+A(1,1)+A(1,2)<br>0+15+18           | -  |
| 240       | INPUT "REZERVORUL 3"; A(Z,3) | 1 | A(1,3)<br>30 | S+A(1,1)+A(1,2)<br>0+15+18           | -  |
| 260       | S=S+A(Z,3)                   | 1 | A(1,3)<br>30 | S+A(1,1)+A(1,2)+A(1,3)<br>0+15+18+30 | -  |
| 280       | B(Z)=S/3                     | 1 | A(1,3)<br>30 | S+A(1,1)+A(1,2)+A(1,3)<br>0+15+18+30 | 21 |
| .         | .                            | . | .            | .                                    | .  |
| .         | .                            | . | .            | .                                    | .  |
| .         | .                            | . | .            | .                                    | .  |

| 0   | 1                             | 2 | 3            | 4                                    | 5  |
|-----|-------------------------------|---|--------------|--------------------------------------|----|
| 200 | S=0                           | 6 | 0            | 0                                    | 44 |
| 210 | INPUT "REZERVORUL 1"; A(Z,1)  | 6 | A(6,1)<br>60 | 0                                    | 44 |
| 220 | S=S+A(Z,1)                    | 6 | A(6,1)<br>60 | S+A(6,1)<br>0+60                     | 44 |
| 230 | INPUT "REZERVORUL 2"; A(Z,2)  | 6 | A(6,2)<br>60 | S+A(6,1)<br>60                       | 44 |
| 235 | S=S+A(Z,2)                    | 6 | A(6,2)<br>60 | S+A(6,1)+A(6,2)<br>0+60+60           | 44 |
| 240 | INPUT "REZERVORUL 3"; A(Z, 3) | 6 | A(6,3)<br>60 | S+A(6,1)+A(6,2)<br>0+60+60           | 44 |
| 260 | S=S+A(Z,3)                    | 6 | A(6,3)<br>60 | S+A(6,1)+A(6,2)+A(6,3)<br>0+60+60    | 44 |
| 280 | B(Z)=S/3                      | 6 | A(6,3)<br>60 | S+A(6,1)+A(6,2)+A(6,3)<br>0+60+60+60 | 60 |

Lista cu rezultatele execuției este prezentată în figura 6.11.

```

:RUN
 INTRODUCETI LIVRARILE
 DE BENZINA PE ZILE IN ORDINEA
 R1 R2 R3
LUNI :
REZERVORUL 1?15 1
REZERVORUL 2?18
REZERVORUL 3?30

REZERVORUL 1?23
REZERVORUL 2?40
REZERVORUL 3?0

MIERCURI :
REZERVORUL 1?55
REZERVORUL 2?26
REZERVORUL 3?1

JOI :
REZERVORUL 1?0
REZERVORUL 2?0
REZERVORUL 3?0

VINERI :
REZERVORUL 1?12
REZERVORUL 2?60
REZERVORUL 3?60

SIMBATA :
REZERVORUL 1?60
REZERVORUL 2?60
REZERVORUL 3?60

```

Fig. 6.11.

| ZIUA     | R1   | R2 | R3      | MEDIA   |
|----------|------|----|---------|---------|
| LUNI     | 15   | 18 | 30      | 21      |
| MARTI    | 23   | 40 | 0       | 21      |
| MIERCURI | 55   | 26 | 1       | 27.3333 |
| JOI      | 0    | 0  | 25      | 8.33333 |
| VINERI   | 12   | 60 | 60      | 44      |
| SIMBATA  | 60   | 60 | 60      | 60      |
| MEDIA    | 27.5 | 34 | 29.3333 |         |

MAXIMUL LIVRARILOR 60 TONE  
 LA REZERVORUL = 2  
 IN ZIUA DE = VINERI

STOP AT LINE 710

Fig. 6.11. continuare.

**Observație.** Pentru citirea dinamică a matricei A se poate utiliza și instrucțiunea **MAT INPUT**.

**Aplicație.** Să se determine prin metoda Newton-Raphson soluția ecuației:

$$x^2 - 3 = 0$$

cunoscând: soluția inițială  $x=3$ , precizia  $=0,000001$ .

Programul va apela subrutina NERA (5000) a cărei tabelă de variabile este ilustrată în tabelul 6.9. Programul sursă al subrutinei este prezentat în tabelul 6.10.

Tabelul 6.9

| TABELA DE VARIABILE  |                       |                      |
|----------------------|-----------------------|----------------------|
| Variabile de intrare | Variabile de stare    | Variabile de ieșire  |
| T1: precizia         | D6: delta X           | X1: soluția ecuației |
| X: soluția inițială  | F: funcția            |                      |
|                      | F1: derivata funcției |                      |

Tabelul 6.10

| PROGRAM SURSĂ   |                                               |
|-----------------|-----------------------------------------------|
| 5000 REM        | 5060 D6=F/F1                                  |
| 5010 X1=X       | 5070 X=X1-D6                                  |
| 5020 GOSUB 5200 | 5080 PRINT "X="; X1; ", fX="; F; ", dfx="; F1 |
| 5030 REM        | 5090 IF (ABS(D6)) >= ABS(T1 * X) THEN 5010    |
| 5040 REM        | 5100 RETURN                                   |
| 5050 REM        |                                               |

**Notă.** Subrutina 5200 conține funcția (F) și derivata funcției. De exemplu, pentru ecuația  $x^2 - 3 = 0$ , corespund instrucțiunile:

5200 F=X \* X-3  
 5210 F1=2 \* X  
 5220 RETURN

```

10 REM Metoda Newton–Raphson
15 REM X1
20 REM D6
30 REM F
40 REM F1
50 REM T1
55 REM X
60 REM
70 T1=.000001
80 X=3
90 GOSUB 5000
100 PRINT
110 PRINT "Soluția este"; X
120 STOP
5000 REM
5010 X1=X
5020 GOSUB 5200
5030 REM
5040 REM
5050 REM
5060 D6=F/F1
5070 X=X1-D6
5080 PRINT "X="; X1; ", f x="; F; ", d fx="; F1
5090 IF (ABS(D6) >= ABS(T1 * X)) THEN 5010
5100 RETURN
5200 F=X * X-3
5210 F1=2 * X
5220 RETURN
5230 END

```

Soluția inițială  
Delta x  
Funcția  
Derivata funcției  
Precizia  
Soluția ecuației

## TEST

Scrieți un program BASIC–PLUS care să editeze matricea:

```

1 2 3
4 5 6
7 8 9
10 11 12
13 14 15
R.
10 DIM X(5, 3)
20 MAT READ X
30 MAT PRINT X ;
40 DATA 1, 2, 3, 4, 5, 6, 7
50 DATA 8, 9, 10, 11, 12, 13, 14, 15
60 END

```

Precizați care va fi rezultatul execuției programului, dacă linia 10 se modifică în:  
10 DIM X(3, 5)

## Jocuri pe calculator

1. Scrieți un program care să simuleze aruncarea a două zaruri de 50 de ori. Programul va afișa numărul și frecvența de apariție a unei fețe a celor două zaruri.

```

10 DIM S(15)
20 FOR I=2 TO 12
30 LET S(I)=0
40 NEXT I
50 FOR I=1 TO 50
60 LET S1=INT (6 * RND+1)
70 LET S2=INT (6 * RND+1)
80 LET S=S1+S2
90 LET S(S)=S(S)+1
100 NEXT I
110 PRINT
120 PRINT "NUMĂR"; "FRECVENȚA"

```

```

130 FOR I=2 TO 12
140 PRINT I, S(I)
150 NEXT I
160 STOP

```

2. Scrieți un program care să simuleze aruncarea unui zar de 50 de ori. Programul va afișa numărul și frecvența de apariție a unei fețe a zarului.

```

10 DIM S(6)
20 FOR I=1 TO 6
30 S(I)=0
40 NEXT I
50 FOR I=1 TO 50
60 S=INT (6 * RND+1)
70 S(S)=S(S)+1
30 NEXT I
90 PRINT
100 PRINT "NUMĂR"; "FRECVENȚA"
110 FOR I=1 TO 6
120 PRINT I, S(I)
130 NEXT I
140 STOP

```

### □ Particularități ale programării în limbajul ABASIC

vol. 2, pag. 226

De notat (v. vol. 2, pag. 226) asemănarea programului cu cele executate pe calculatoarele AMSTRAD, M118, TPD și JUNIOR.

La o analiză mai puțin atentă a programului grăbindu-vă poate să treceți mai repede la tema acestei conversații, s-ar putea să rămână neobservat un amănunt important legat de instrucțiunea **DATA**.

Comparînd-o cu instrucțiunea omoloagă se constată că în ABASIC constantele alfanumerice (șir) trebuie să fie cuprinse între ghilimele.

### TEST

Să se determine prin metoda Newton-Raphson soluția ecuației

$$x^2 - 3 = 0$$

cunoscînd: soluția inițială  $x=3$ , precizia= $0,000001$  și numărul de iterații 20. Programul va apela subrutina NERA1. Programul sursă al subrutinei este prezentat în tabelul 6.11 iar tabela de variabile a aceleași subrutine este ilustrată în tabelul 6.12.

Tabelul 6.11

#### PROGRAM SURSA

```

5000 REM
5010 E1% = FO%
5020 FOR I% = 1 TO M5%
5030 X1 = X
5040 GOSUB 5200
5050 IF (ABS (FA) > H2) THEN 5090
5060 PRINT "EROARE"
5070 E1% = T0%
5080 GOTO 5160
5090 D6 = F / F1
5100 X = X1 - D6
5110 PRINT "X="; X1; ", fx="; ", dfx="; F1
5120 IF (ABS (D6) <= ABS (T1 * X)) THEN 5160
5130 NEXT I%
5140 PRINT "LIPSA CONVERGENȚĂ ÎN"; M5%; "ITERAȚII"
5150 E1% = T0%
5160 RETURN

5200 F = X * X - 3
5210 F1 = 2 * X
5220 RETURN

```



Tabelul 6.12

| TABELA DE VARIABILE                                |                                               |                      |
|----------------------------------------------------|-----------------------------------------------|----------------------|
| Variabile de intrare                               | Variabile de stare                            | Variabile de ieșire  |
| X: soluție inițială                                | H2: număr foarte mic (1E-15)                  | X1: soluția ecuației |
| T1: precizia                                       | F: funcția                                    |                      |
| M5 <sup>0</sup> / <sub>0</sub> : număr de iterații | F1: derivata funcției                         |                      |
|                                                    | D6: delta X                                   |                      |
|                                                    | FO <sup>0</sup> / <sub>0</sub> : FALSE        |                      |
|                                                    | T0 <sup>0</sup> / <sub>0</sub> : TRUE         |                      |
|                                                    | I: variabila de control a buclei              |                      |
|                                                    | E1 <sup>0</sup> / <sub>0</sub> : stare eroare |                      |

**Remarcă.** Introduceți în biblioteca dvs. de programe științifice subrutina NERA1.

```

10 REM D6 Delta X
15 REM E1 stare eroare
16 REM F funcția
17 REM F1 derivația funcției
18 REM F0 FALS (zero)
19 REM H2 număr mic
20 REM M5 număr iterații
21 REM T0 ADEVĂRAT (unu)
22 REM T1 precizia
23 T1=0.000001
30 H2=1E-15
40 M50/0=20
50 FO0/0=0
60 T00/0=NOT FO0/0
70 INPUT "Soluția inițială"; X
80 IF (X<-19) THEN 9999
90 GOSUB 5000
100 PRINT
110 IF (E10/0=FO0/0) THEN PRINT "Soluția este"; X
120 GO TO 70
5000
 (v. subrutina NERA1)
5200 F=X * X
5210 F1=2 * X
5220 RETURN
9999 END

```

### Aplicații

1. Scrieți un program care să furnizeze, la cerere, capitala unei țări:

```

10 INPUT "ȚARA"; P$
30 WHILE P$ <> A$ AND P$ <> "XXX"
40 READ A$, B$
50 IF P$=A$ THEN PRINT P$, A$
60 WEND
70
75 IF P$="XXX" THEN PRINT "ȚARA NECUNOSCUTĂ PROGRAMULUI"
80 DATA FRANȚA, PARIS
90 DATA ROMÂNIA, BUCUREȘTI
100 DATA BULGARIA, SOFIA
110 DATA SPANIA, MADRID
120 DATA ITALIA, ROMA
130 END

```

2. Să se determine lucrul mecanic total al cuplului motor de moment, pentru un arbore dat:

$$M_m = 50\sqrt{2\theta^2 + 3\theta + 4} \quad [\text{N}\cdot\text{m}/\text{rad}] \quad (14)$$

la o rotație  $\theta \in [0, 2\pi]$ .

**Indicație.** Pentru rezolvarea problemei se utilizează relațiile:

$$dL = M \cdot \omega dt = M d\theta \quad (15)$$

Integrând (14) obținem:

$$L = \int_{\theta_1}^{\theta_2} M d\theta \quad [\text{N}\cdot\text{m}] \quad (16)$$

Așadar, problema se reduce la calculul integralei

$$L_m = \int_{\theta_1}^{\theta_2} M_m d\theta \quad (17)$$

Pentru calculul integralei (17) se va utiliza subrutina TRAPEZ. Se cunosc:  $\theta_1 = 30^\circ$ ;  $\theta_2 = 75^\circ$ ; număr intervale = 10.

3. Introduceți și executați următorul program:

```

10 INPUT "NUMELE PRIETENULUI"; P$
20 WHILE N$ <> "AAA" AND N$ <> P$
30 READ N$, A$
40 IF N$ = P$ THEN PRINT P$, A$
50 WEND
60 IF N$ = "xxx" THEN PRINT "NUME NECUNOSCU PROGRAMULUI"
70 DATA IONESCU ION, 10 SEPTEMBRIE
80 DATA AGA ION, 5 IULIE
90 DATA LEU VASILE, 3 MAI
95 DATA xxx, xxx
100 END

```

4. Să se rezolve sistemul de ecuații (v. aplicație, pag. 319) prin metoda eliminării a lui GAUSS-JORDAN.

Tabela de variabile și schema logică a programului (subrutinele 500; 5000) sint prezentate în tabelul 6.13 respectiv figura 6.12.

Tabelul 6.13

| TABELA DE VARIABILE           |                        |                      |
|-------------------------------|------------------------|----------------------|
| Variabile de intrare          | Variabile de stare     | Variabile de ieșire  |
| N1: număr de linii            | E1: stare eroare       | C1: vector rezultate |
| N2: număr de coloane          | I2: matrice de lucru   |                      |
| A: matricea coeficienților    | M1: lungime maximă     |                      |
| Z: vectorul termenilor liberi | B: matrice de lucru    |                      |
|                               | B1: variabilă de lucru |                      |
|                               | D3: determinant        |                      |
|                               | H1: variabilă de lucru |                      |
|                               | I3: index linie        |                      |
|                               | I4: index coloană      |                      |
|                               | I5: variabilă de lucru |                      |
|                               | N3: număr vectori      |                      |
|                               | coeficienți            |                      |
|                               | P1: index pivot        |                      |
|                               | W: matricea soluțiilor |                      |

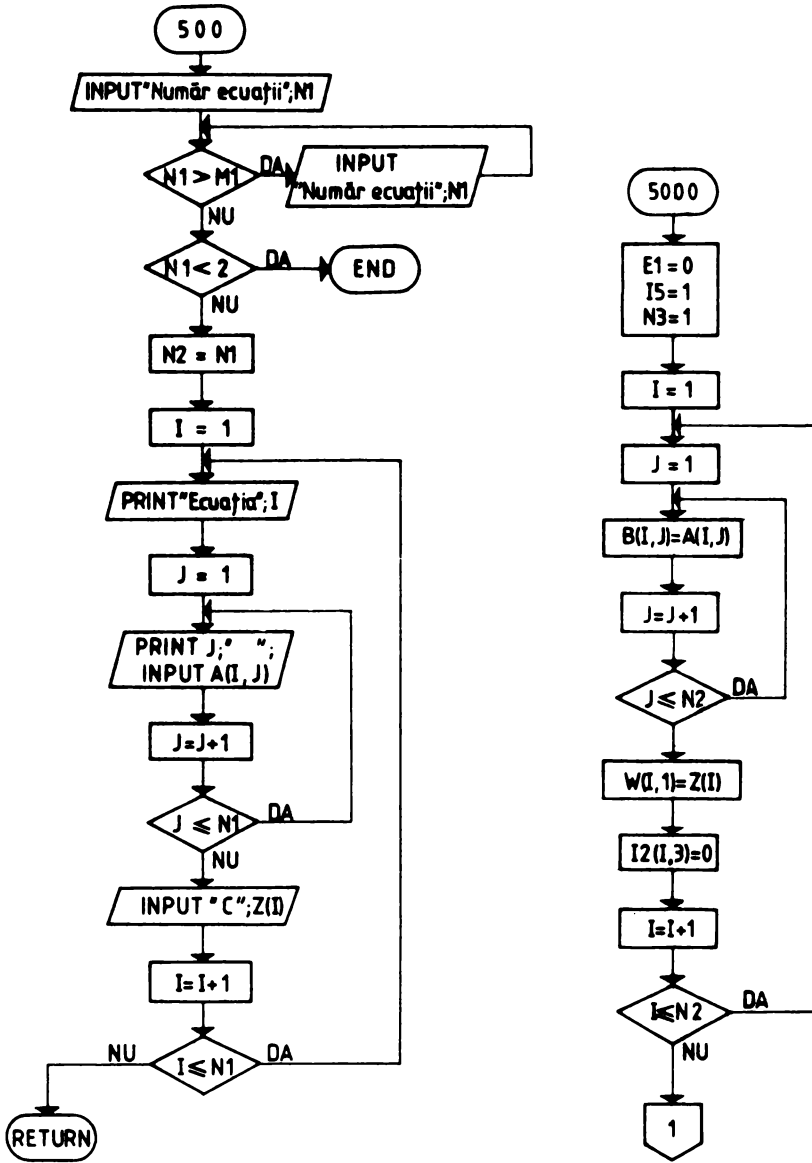


Fig. 6.12.

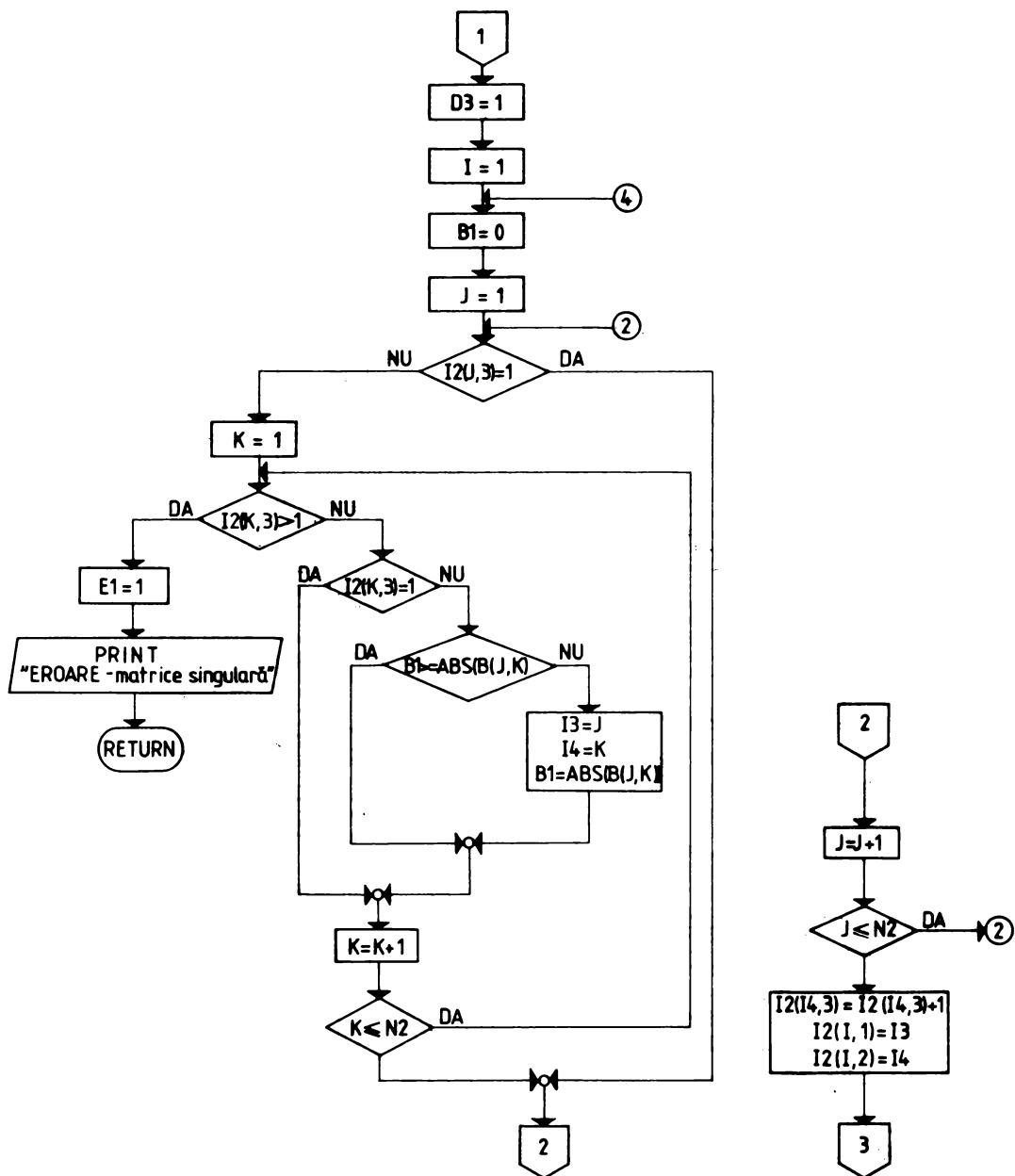


Fig. 6.12.

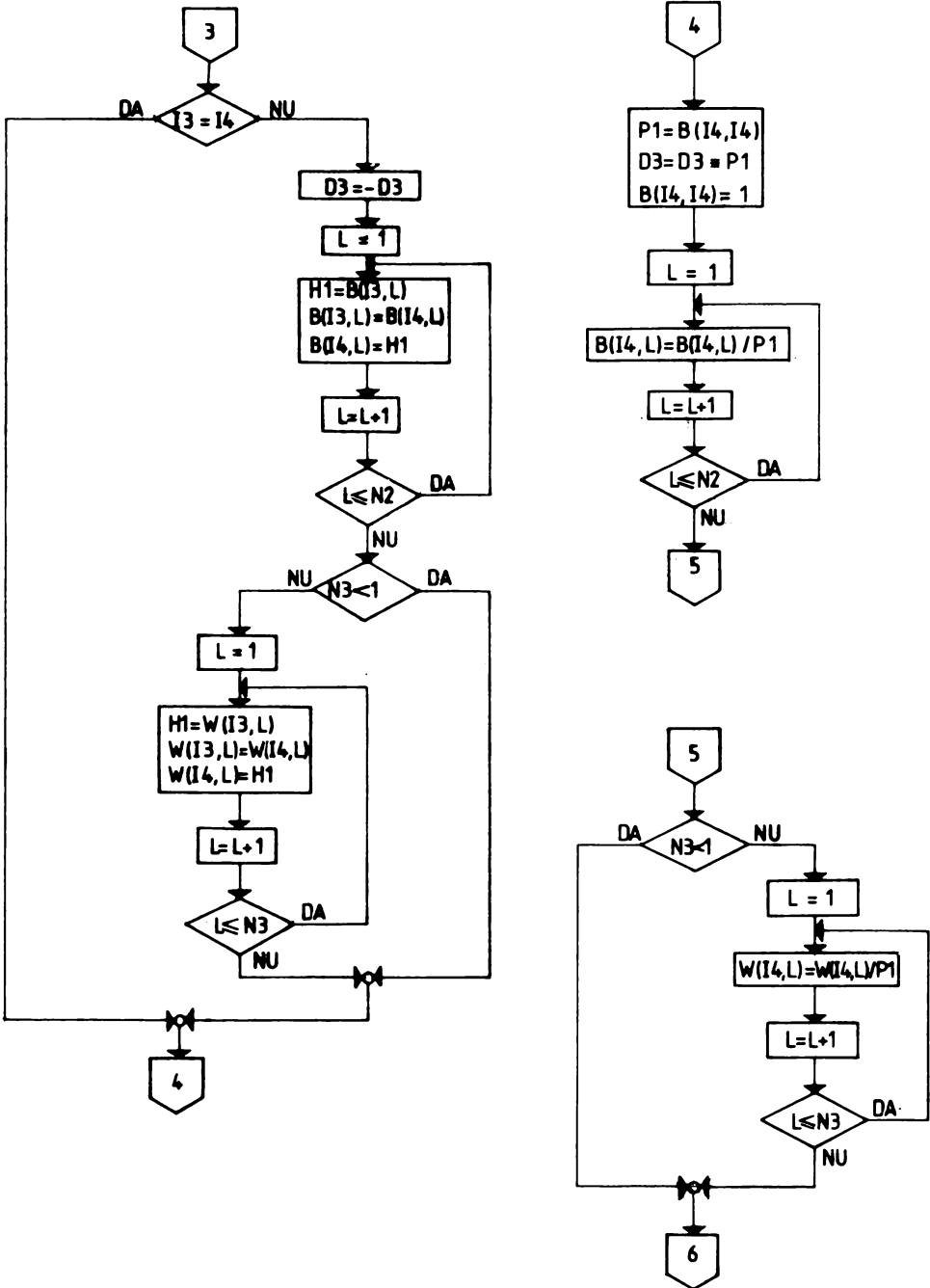


Fig. 6.12.

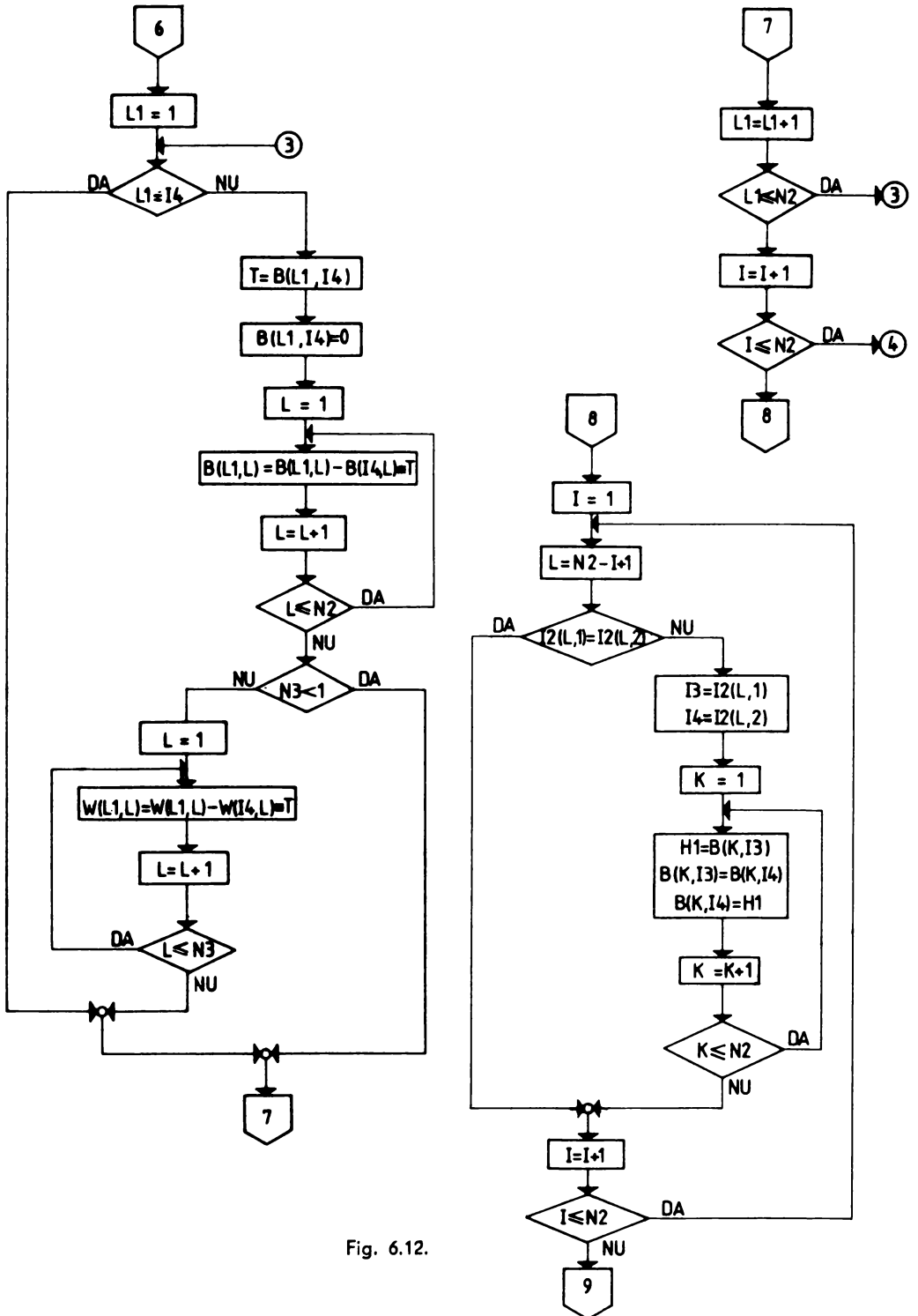


Fig. 6.12.

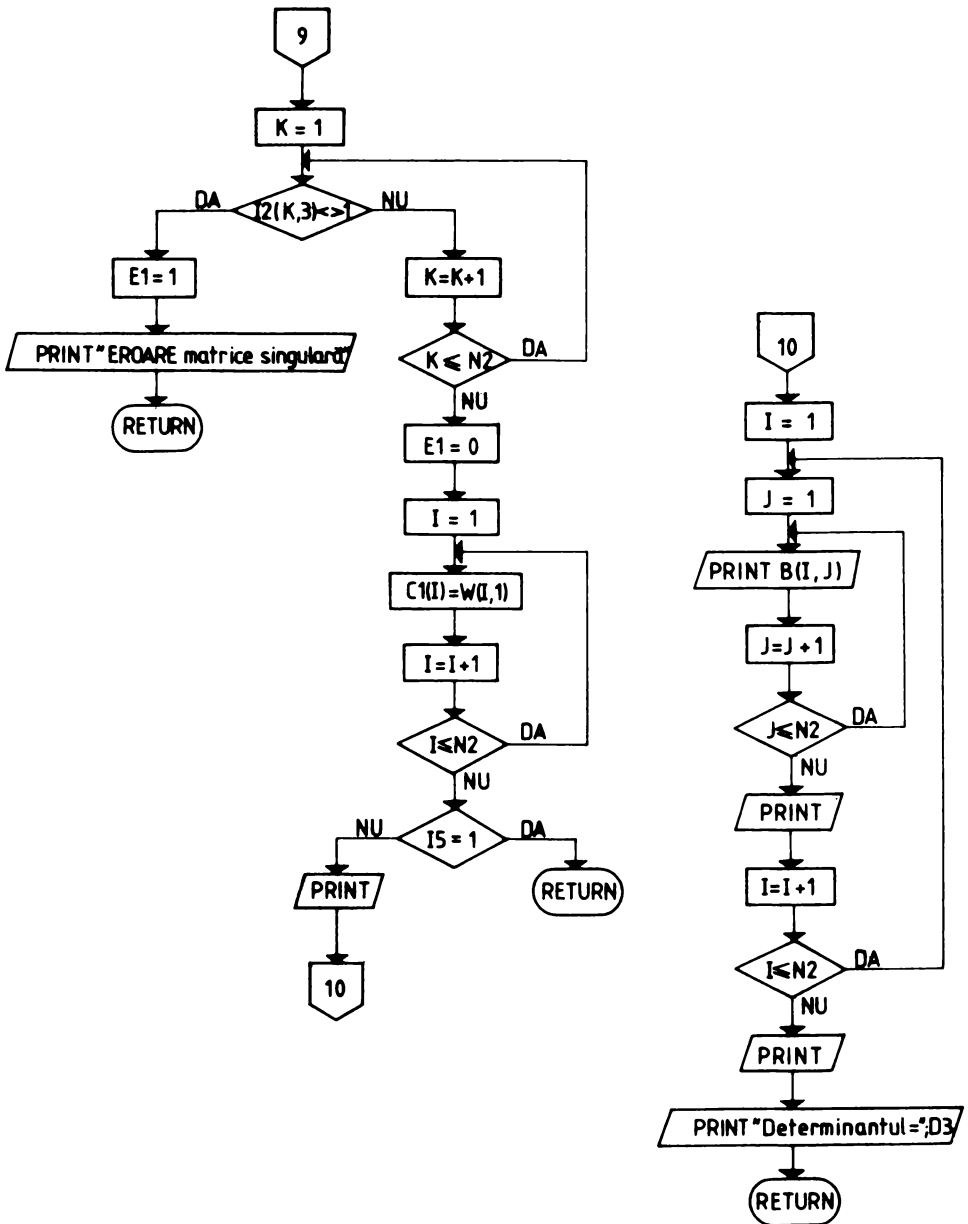


Fig. 6.12.

Programul principal apelează subrutina GAUSS-JORDAN prin instrucțiunea

100 GOSUB 5000

De notat, că în această versiune matricea originală A (matricea coeficienților) este duplicată, de la început în matricea B (v. liniile 5200–5230). Inversa matricei coeficienților este plasată în matricea B în locul datelor inițiale. Vectorul termenilor liberi Z, este de asemenea izolat de vectorul rezultatelor – C1. Se pot afișa, de asemenea inversa matricei și determinantul matricei coeficienților.

Introduceți valorile pentru sistemul de ecuații studiat (v. aplicația de la pag. 319) și verificați dacă vectorul soluție obținut este [14,-4,10].

### Remarci

- Simulați cu creionul și hirtia funcționarea programului.
- Introduceți în biblioteca dvs. de programe științifice acest program.

```

5 INPUT "Tastați 1 (continuare)/0 (oprire)"; T
10 WHILE T=1
15 REM C1 vector rezultate
16 REM E1 stare eroare
17 REM I2 matrice de lucru
18 REM M1 lungime maximă
19 REM N1 număr de linii
20 REM N2 număr de coloane
25 REM
30 A$="# #.# #.# ^^^^"
40 B$="# #.# #.# ^^^^"
50 C$="# #.# #.#" "# #"
60 M1=8
70 DIM Z(8), A(8,8), C1(8), W(8,1), B(8,8), I2(8,3)
80 REM
90 GOSUB 500
100 GOSUB 5000
110 REM
120 IF (N1>5) THEN 200
130 PRINT "Matricea coeficienților"
140 FOR I=1 TO N1
150 FOR J=1 TO N2
160 PRINT USING A$; A(I, J);
170 NEXT J
180 PRINT USING B$; Z(I)
190 NEXT I
200 PRINT
210 IF (E1=1) THEN 280
120 IF (N1>5) THEN 200
230 PRINT
240 FOR I=1 TO N2
250 PRINT USING C$; C1(I);
260 NEXT I
270 PRINT
280 INPUT "Tastați 1 (continuare)/0 (oprire)"; T
290 WEND
300 STOP
500 REM
510 PRINT
520 INPUT "Număr ecuații"; N1
530 WHILE N1>M1
535 INPUT "Număr ecuații"; N1
538 WEND
540 IF (N1<2) THEN 6200
550 N2=N1
560 FOR I=1 TO N1
570 PRINT "Ecuația"; I
580 FOR J=1 TO N1

```



```

590 PRINT J; " ";
600 INPUT A(I,J)
610 NEXT J
620 INPUT "C"; Z(I)
630 NEXT I
640 RETURN
5000 REM A matricea coeficienților
5010 REM B matricea de lucru
5020 REM B1 variabilă de lucru
5030 REM C1 vectorul rezultatelor
5040 REM D3 determinant
5050 REM E1 stare eroare
5060 REM H1 variabila de lucru
5070 REM I2 matrice de lucru
5080 REM I3 index linie
5090 REM I4 index coloană
5100 REM I5 variabilă de lucru
5110 REM N2 număr de coloane
5120 REM N3 număr vectori coeficienți
5130 REM P1 index pivot
5140 REM W matricea soluției
5150 REM Z vectorul coeficienților (termenilor liberi)
5160 REM
5170 E1=0
5180 I5=1
5190 N3=1
5200 FOR I=1 TO N2
5210 FOR J=1 TO N2
5220 B(I,J)=A(I,J)
5230 NEXT J
5240 W(I, 1)=Z(I)
5250 I2(I,3)=0
5260 NEXT I
5270 D3=1
5280 FOR I=1 TO N2
5290 REM
5300 REM căutare pivot
5310 REM
5320 B1=0
5330 FOR J=1 TO N2
5340 IF (I2(J,3)=1) THEN 5430
5350 FOR K=1 TO N2
5360 IF (I2(K,3) > 1) THEN 6170
5370 IF (I2(K,3)=1) THEN 5420
5380 IF (B1 >= ABS (B(J,K))) THEN 5420
5390 I3=J
5400 I4=K
5410 B1=ABS (B(J, K))
5420 NEXT K
5430 NEXT J
5440 I2(I4,3)=I2(I4,3)+1
5450 I2(I,1)=I3
5460 I2(I,2)=I4
5470 REM punere pivot pe diagonală
5480 IF (I3=I4) THEN 5620
5490 D3=-D3
5500 FOR L=1 TO N2
5510 H1=B(I3, L)
5520 B(I3,L)=B(I4,L)
5530 B(I4, L)=H1
5540 NEXT L
5550 IF (N3 < 1) THEN 5620
5560 FOR L=1 TO N3
5570 H1=W(I3, L)

```

```

5580 W(I3, L)=W(I4, L)
5590 W (I4, L)=H1
5600 NEXT L
5610 REM
5620 P1=B(I4, I4)
5630 D3=D3*P1
5640 B(I4, I4)=1
5650 FOR L=1 TO N2
5660 B(I4, L)=B(I4, L)/P1
5670 NEXT L
5680 IF (N3 < 1) THEN 5730
5690 FOR L=1 TO N3
5700 W(I4,L)=W(I4, L)/P1
5710 NEXT L
5720 REM
5730 FOR L1=1 TO N2
5740 IF (L1=I4) THEN 5835
5750 T=B(L1, I4)
5760 B(L1, I4)=0
5770 FOR L=1 TO N2
5780 B(L1, L)=B(L1, L)-B(I4, L) * T
5790 NEXT L
5800 IF (N3 < 1) THEN 5835
5810 FOR L=1 TO N3
5820 W(L1, L)=W(L1, L)-W(I4, L) * T
5830 NEXT L
5835 NEXT L1
5840 NEXT I
5850 REM
5860 REM interschimbare coloane
5870 FOR I=1 TO N2
5880 L=N2-I+1
5890 IF (I2 (L,1)=I2 (L,2)) THEN 5970
5900 I3=I2 (L,1)
5910 I4=I2 (L,2)
5920 FOR K=1 TO N2
5930 H1=B(K, I3)
5940 B(K,I3)=B(K,I4)
5950 B(K,I4)=H1
5960 NEXT K
5970 NEXT I
5980 FOR K=1 TO N2
5990 IF (I2(K, 3) < > 1) THEN 6170
6000 NEXT K
6010 E1=0
6020 FOR I=1 TO N2
6030 C1(I)=W(I,1)
6040 NEXT I
6050 IF (I5=1) THEN 6190
6060 PRINT
6070 PRINT "Inversa matricei"
6080 FOR I=1 TO N2
6090 FOR J=1 TO N2
6100 PRINT USING A$; B(I,J);
6110 NEXT J
6120 PRINT
6130 NEXT I
6140 PRINT
6150 PRINT "Determinantul="; D3
6160 RETURN
6170 E1=1
6180 PRINT "EROARE matrice singulară"
6190 RETURN
6200 END

```

## TEMA 6

 **Răspundeți prin DA sau NU la următoarele întrebări:**

Instrucțiunile matriciale sînt precedate de cuvîntul cheie **MAT**. Limbajul BASIC-aMIC și limbajul BASIC-PLUS permit efectuarea următoarelor operații cu masive, luate ca un tot:

- adunarea,
- scăderea,
- înmulțirea cu un scalar,
- împărțirea a două masive,
- înmulțirea a două masive,
- atribuirea.

- Instrucțiunile matriciale **ZER**, **COD**, **IDN** se folosesc la inițializarea unor masive numerice.

- Atunci cînd se folosește instrucțiunea **MAT INPUT** elementele unui masiv se introduc pe linie.

- Unei instrucțiuni **MAT READ** i se asociază instrucțiunea **DATA** (corespunzător elementelor din liniile masivului).

- În cazul în care într-un program nu se lucrează cu masive ale căror dimensiuni sînt mai mari ca 10 instrucțiunea **DIM** poate fi omisă.

- Secvența de instrucțiuni:

```
50 DIM A(11, 11), B(11, 11), C(11, 11), D(11, 11)
60 MAT INPUT A, B, C, D
```

este corectă.

- Secvența de instrucțiuni BASIC-aMIC:

```
10 DIM A$(20, 5)
20 INPUT A$
```

nu este corectă.

- Secvența de instrucțiuni BASIC-PLUS

```
10 DIM A(6,9)
15 MAT=INV (A)
```

nu este corectă.

- Secvența de instrucțiuni BASIC-PLUS

```
10 DIM A(6,6)
15 MAT=INV (A)
```

este corectă.

- Instrucțiunea BASIC-COMMODEORE

```
10 A=A+B(I, -2)
```

este incorectă.

– Secvența de instrucțiuni BASIC-PLUS

```
10 DIM A(30)
20 MAT A=IDN
```

este incorectă.

– Instrucțiunea

```
10 DIM A(20), B(10,15), X, Y, Z, U(15)
```

este eronată.

– Instrucțiunea ABASIC

```
10 X(A(I))=Y(A(I+1))+3
```

este corectă.

– Următorul program:

```
10 DIM B(4,5)
20 FOR L=1 TO 4
30 FOR C=1 TO 5
40 B (L, C)=8
50 NEXT C
60 NEXT L
70 FOR L=1 TO 4
80 FOR C=1 TO 5
90 PRINT B (L,C);
100 NEXT C
110 PRINT
120 PRINT
130 NEXT L
140 END
```

afișează cifra 8 de 20 de ori, într-o matrice cu 4 linii și 5 coloane.

– Funcția **RANDOMIZE** inițializează generatorul de numere aleatoare.

– În urma execuției programului

```
10 FOR I=1 TO 3
20 PRINT RND
30 NEXT I
```

se va genera aceeași secvență de numere aleatoare.

– În BASIC-aMIC punctul de coordonate (1,1) se află în colțul din stînga sus al ecranului iar punctul de coordonate (32, 32) în dreapta jos.

– În BASIC HC-85 variabilele șir și variabilele indexate pot avea același nume.

– Variabilele șir trebuie declarate într-o instrucțiune **DIM**.

**Inlocuiți cuvintele care lipsesc din următoarele propoziții:**

a) Operația de adunare a masivului A cu masivul X, cu plasarea rezultatului în masivul Y, se realizează cu instrucțiunea matricială .....

b) Operația de scădere a două masive X și Y, cu plasarea rezultatului în masivul Z se realizează cu instrucțiunea matricială .....

c) Operația de înmulțire a matricii X cu scalarul S se realizează cu instrucțiunea matricială .....

d) Operația de înmulțire a două masive X și Y, cu plasarea rezultatului în masivul Z se realizează cu instrucțiunea matricială .....

e) Instrucțiunile matriciale prezintă următoarele avantaje.....

f) Deosebirea între o instrucțiune **MAT READ** și o instrucțiune **MAT INPUT** constă în .....

g) Secvența de instrucțiuni:

```
10 INPUT C, L
20 DIM A(C,L), B(C), X(L)
```

este corectă numai dacă .....

h) Pentru rezolvarea unei ecuații prin metoda Newton-Raphson se introduc următoarele date .....

i) Pentru calculul integralelor prin metoda trapezelor trebuie cunoscute .....

j) În urma execuției programului:

```
10 DIM X(6,6)
15 MAT READ X (4,5)
20 MAT PRINT X;
30 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
40 DATA 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
50 DATA 21, 22, 23, 24, 25, 26, 27, 8, 29, 30
60 DATA 31, 32, 33, 34, 35, 36
70 END
```

rezultă .....

k) În urma execuției programului:

```
10 DIM A(1,4)
20 MAT READ A
30 MAT PRINT A
40 DATA 1, 2, 3, 4
50 END
```

rezultă .....

l) În urma execuției programului:

```
10 DIM Y(4,1)
20 MAT READ A
30 MAT PRINT A
40 DATA 1, 2, 3, 4
50 END
```

rezultă .....

m) în urma execuției programului:

```
10 DIM B(4,5)
20 MAT READ B
30 MAT PRINT B;
40 DATA 8, 8, 8, 8, 8, 8, 8, 8, 8, 8
50 DATA 8, 8, 8, 8, 8, 8, 8, 8, 8, 8
60 END
```

se afișează 4 linii cu ..... și ..... coloane cu .....

n) În BASIC-aMIC, într-o instrucțiune **PRINT AT** coordonatele pentru <nr. linie> și <nr. coloane> se transmit..... iar în BASIC-PRAE .....

o) În BASIC-PRAE simbolurile folosite pentru editarea valorilor numerice sînt .....

p) În instrucțiunea **PRINT** din limbajul BASIC TIM S se pot specifica linia ..... și coloana .....

r) Limbajul BASIC implementat pe calculatoarele ..... nu acceptă instrucțiunile matriciale.

s) Instrucțiunea **PRINT AT** este acceptată în BASIC .....

t) În BASIC ..... este posibilă utilizarea unor subșiruri dintr-un șir.

u) Ecranul TV, din punctul de vedere al PRAE-ului este împărțit în ..... puncte pe orizontală și ..... puncte pe verticală. Punctul de coordonare (0, 0) este stabilit în .....

Să se scrie câte un program BASIC pentru fiecare din problemele de mai jos:

a) Se consideră matricile

$$A = \begin{bmatrix} 1 & -2 & 3 \\ 5 & -1 & -2 \\ 0 & 3 & 4 \end{bmatrix}; \quad B = \begin{bmatrix} 2 & -4 & 0 \\ -3 & 1 & 2 \\ 5 & 2 & -5 \end{bmatrix}$$

Să se calculeze:

a1)  $(A*B)^{-1}$

a2)  $B^{-1}*A^{-1}$

b) Se consideră matricea:

$$Y = \begin{bmatrix} 4 & -4 & 4 \\ 1 & 1 & 7 \\ -3 & 9 & -8 \end{bmatrix}$$

Să se calculeze:

b1)  $Y^{-1}$

b2)  $Y*Y^{-1}$

b3)  $Y^{-1}*Y$

c) Să se calculeze:

$$\left[ [1 \quad 1 \quad 1] \cdot \begin{bmatrix} 20 & 700 & 30 \\ 40 & 21 & 10 \\ 0 & 400 & 15 \end{bmatrix} \right] \cdot \begin{bmatrix} 2.05 \\ .23 \\ .30 \\ .25 \end{bmatrix}$$

d) Să se citească matricea

$$\begin{bmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{bmatrix}$$

și să se tipărească sub forma:

$$\begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \end{array}$$

e) Să se rezolve următoarele sisteme de ecuații:

e1)  $-3w+6x-5y-z=-32$   
 $w+9x-5y-2z=9$   
 $w+6y+5z=2$   
 $-7w+4x-y+5z=-56$

e2)  $7v+6w-3x-y+9z=26.3$   
 $-9v+2w+9x+5y+z=91.1$   
 $-3v+4w+5x+5z=62.9$   
 $6v-8x-2y-6z=-55.6$   
 $-3v-9w-5x+7y+3z=-25.9$

e3)  $-2x-5y=-16$   
 $-x+4y=31$

f) Pentru termocuplul fier-constantan, dependențele dintre tensiunea electromotoare și agitația termică măsurată prin temperatură pot fi aproximare prin drepte ai căror coeficienți se pot determina printr-o regresie liniară.

În tabelul 6.14 sînt ilustrate rezultatele măsurătorilor de laborator.

Tabelul 6.14

| T°C | (mv) | T°C | (mv) |
|-----|------|-----|------|
| 24  | 0,11 | 40  | 1,06 |
| 26  | 0,22 | 41  | 1,1  |
| 28  | 0,38 | 44  | 1,24 |
| 29  | 0,46 | 48  | 1,48 |
| 32  | 0,58 | 50  | 1,56 |
| 33  | 0,69 | 53  | 1,79 |
| 35  | 0,80 | 58  | 2,02 |

La proiectarea și elaborarea programului se vor avea în vedere următoarele funcțiuni:

1. Citirea datelor de intrare
2. Afișarea datelor de intrare
3. Apelarea subrutinei de regresie liniară REGLIN (500).
4. Imprimarea coeficienților (a, b) dreptei de regresie.

În tabelul 6.15 și tabelul 6.16 se prezintă tabela de variabile a subrutinei respectiv programul sursă al aceleași subrutine.

Tabelul 6.15

| TABELA DE VARIABILE                           |                                    |                              |
|-----------------------------------------------|------------------------------------|------------------------------|
| Variabile de intrare                          | Variabile de stare                 | Variabile de ieșire          |
| N: numărul valorilor experimentale            | S1, S2, S3, S4: variabile de lucru | A: termenul liber            |
| X: vectorul valorilor variabilei independente |                                    | B: panta dreptei de regresie |
| Y: vectorul valorilor variabilei dependente   |                                    |                              |

Tabelul 6.16

---

PROGRAM SURSA

---

```

500 S1=0.0
510 S2=0.0
520 S3=0.0
530 S4=0.0
550 FOR I=1 TO N
560 S1=S1+X(I)
570 S2=S2+Y(I)
580 S3=S3+X(I) * X(I)
590 S4=S4+X(I) * Y(I)
600 NEXT I
610 A=(S2 * S3-S4 * S1)/(N * S3-S1 * S1)
620 B=(N * S4-S1 * S2)/(N * S3-S1 * S1)
630 RETURN

```

---

*Observație.* Pentru calculul coeficienților (a, b) ecuației de regresie

$$y=ax+b \quad (18)$$

s-au folosit formulele:

$$a = \frac{\sum y_i \sum x_i^2 - \sum x_i \sum x_i y_i}{n \sum x_i^2 - (\sum x_i)^2} \quad (19)$$

$$b = \frac{n \sum x_i y_i - \sum x_i \sum y_i}{n \sum x_i^2 - (\sum x_i)^2} \quad (20)$$

**g)** Pentru calculul geometric al profilului flancurilor dintelui, în cazul angrenajelor cu dinți drepți și profil în evolventă se utilizează funcția involută

$$\text{inv}(\alpha) = \text{tg} \alpha - \alpha \quad (21)$$

Utilizând metoda Newton-Raphson, să se determine valoarea unghiului  $\alpha$  din ecuația:

$$\text{tg} \alpha - \alpha = \gamma \quad (22)$$

pentru:  $\alpha_1 = 20^\circ$ , precizia = 0,000001, numărul de iterații = 10,  $\gamma = 15^\circ$ . Se va apela subrutina NERA1 (v. pag. 342)

**h)** Să se determine prin metoda Newton-Raphson soluția ecuației

$$e^x = 4x$$

cunoscând: soluția inițială  $x = 0,1$ , numărul de iterații = 20 și precizia = 0,000001. Se va apela subrutina NERA1.

*Indicație.* Pentru evaluarea lui  $e^x$  folosiți secvența de instrucțiuni:

```

5200 E=EXP (X)
5210 F=E-4*X
5220 F1=E-4
5230 RETURN

```

**i)** În tabelul 6.17 se prezintă datele experimentale privind variația unei rezistențe cu temperatura în domeniul  $T \in [50, 100^\circ\text{C}]$ .



Tabelul 6.17

|        |    |       |       |       |       |       |      |       |       |       |       |
|--------|----|-------|-------|-------|-------|-------|------|-------|-------|-------|-------|
| T [°C] | 50 | 55    | 60    | 65    | 70    | 75    | 80   | 85    | 90    | 95    | 100   |
| R [Ω]  | 76 | 76,11 | 76,32 | 76,47 | 76,50 | 76,90 | 77,1 | 77,32 | 77,31 | 77,42 | 77,56 |

Să se determine valorile intermediare ale rezistenței conductorului prin **interpolare liniară**.

La proiectarea și elaborarea programului se vor avea în vedere următoarele funcțiuni:

1. Citirea/scrierea datelor de intrare
2. Apelarea subrutinei INTERLIN
3. Tipărire rezultate.

În tabelul 6.18, tabelul 6.19 și tabelul 6.20 se prezintă formatul datelor de ieșire, tabela de variabile a subrutinei INTERLIN respectiv programul sursă al aceleiași subrutine.

Tabelul 6.18

|                             |  |
|-----------------------------|--|
| FORMATUL DATELOR DE IEȘIRE  |  |
| DATELE DE INTRARE           |  |
| T (GR.C)                    |  |
| * * *                       |  |
| * * *                       |  |
| .                           |  |
| .                           |  |
| * * *                       |  |
| R (OHMI)                    |  |
| * * *                       |  |
| * * *                       |  |
| .                           |  |
| .                           |  |
| * * *                       |  |
| DATELE DE IEȘIRE            |  |
| T= * * * GR.C R= * * * OHMI |  |

Tabelul 6.19

|                                                                 |                                                             |                                                              |
|-----------------------------------------------------------------|-------------------------------------------------------------|--------------------------------------------------------------|
| TABELA DE VARIABILE                                             |                                                             |                                                              |
| Variabile de intrare                                            | Variabile de stare                                          | Variabile de ieșire                                          |
| N: Numărul perechilor de valori cunoscute din graficul funcției | KOD: cod de eroare (0 – interpolare reușită;<br>1 – eroare) | Y0: valoarea aproximativă a funcției pentru valoarea X0 dată |
| X: vectorul valorilor argumentului                              | U, V: variabile de lucru                                    | KOD: cod eroare                                              |
| Y: vectorul valorilor funcției                                  |                                                             |                                                              |
| X0: valoarea inițială a argumentului                            |                                                             |                                                              |

Tabelul 6.20

## PROGRAM SURSĂ

```

500 KOD=0
510 N1=N-1
520 FOR I=1 TO N1
530 IF (X0 >= X(I)) AND (X0 <= X(I+1)) THEN 570
540 NEXT I
550 KOD=1
560 RETURN
570 U=Y(I)+(Y(I+1)-Y(I))
580 V=X(I+1)-X(I) * (X0-X(I))
590 Y0=U/V
600 RETURN

```

j) Să se determine intensitățile curenților din laturile circuitului din figura 6.13 prin metoda teoremelor lui Kirchoff. Se dau:

$$E1=8V; E2=48V; r1=3\Omega; r2=2\Omega; R=2\Omega.$$

Pentru rezolvarea sistemului de ecuații se vor utiliza:

- formulele lui Cramer (v. pag. 319),
- metoda eliminării a lui Gauss (v. subrutinele 500, 5000, pag. 331),
- metoda eliminării a lui Gauss-Jordan (v. subrutinele 500, 5000, pag. 344).

k) Scrieți un program care să simuleze aruncarea a două monezi de 30 de ori. Programul va contoriza și afișa numărul pentru care rezultatul acestui experiment imaginar este: leu-leu (LL), stemă-stemă (SS) și leu-stemă (LS) sau stemă-leu (SL).

l) Scrieți un program care să furnizeze, la cerere, informații privind autorii cărților din biblioteca dvs. personală (nume, pronume, lucrare, editură, anul apariției).

m) Se consideră trei rezervoare cilindrice echilaterale care conțin benzină de trei calități. Să se editeze situația livrărilor de benzină pentru un număr oarecare (N) de beneficiari. Programul va afișa în final "TOTAL LIVRĂRI" și "TOTAL VALOARE". Prețurile de livrare pe tona de benzină sînt respectiv 8 750 lei, 7 500 lei și 5 250 lei. Formatul datelor de ieșire, tabela de variabile, pseudocodul și textul conversației sînt prezentate în modulele de analiză și proiectare structurată figura 6.14.

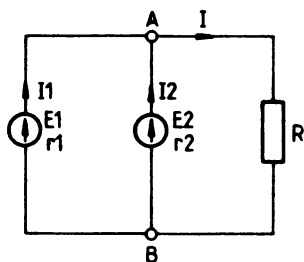


Fig. 6.13.

Se reia problema din conversația precedentă, complicînd-o după cum urmează.

- Se citește de la tastatură numărul de articole (maximum 8);
- Se memorează într-o matrice cu 8 linii (reprezentînd numărul maxim de articole – tipuri de jucării) și 12 coloane (reprezentînd lunile anului) cantitățile lunare de jucării fabricate.

FORMATUL DATELOR DE IEȘIRE

\*\* SITUAȚIA LIVRĂRILOR \*\*

| NR.                    | DENUMIRE | LIVRĂRI | VALOARE |
|------------------------|----------|---------|---------|
| *                      | *****    | ***     | ***     |
| .                      | .        | .       | .       |
| .                      | .        | .       | .       |
| .                      | .        | .       | .       |
| TOTAL LIVRĂRI: ** TONE |          |         |         |
| TOTAL VALOARE: ** LEI  |          |         |         |

a)

TABELA DE VARIABILE

| Variabile de intrare                 | Variabile de stare                     | Variabile de ieșire                    |
|--------------------------------------|----------------------------------------|----------------------------------------|
| B\$: vector nume beneficiari         | S: total livrări pe beneficiar         | S1: total livrări                      |
| L: matrice livrări benzină           | T: total valoare livrări pe beneficiar | T1: total valoare livrări              |
| N: număr beneficiari                 | I: index                               | S: total livrări pe beneficiar         |
| P: vector prețuri pe tona de benzină | J: index                               | T: total valoare livrări pe beneficiar |

b)

PSEUDOCODUL

```

TEMA SEQ
 INPUT "INTRODUCEȚI NUMĂR BENEFICIARI"; N
 PRINT
CIT-PRET PENTRU I DE LA 1 LA 3
 READ P(I)
CIT-PRET SFIRȘIT
 PRINT
CIT-NUM-LIV PENTRU I DE LA 1 LA N
 INPUT "NUME BENEFICIAR"; B$(I)
CIT-LIV PENTRU J DE LA 1 LA 3
 PRINT "LIVRĂRI"; B$(I); "DIN REZERVORUL"; J
 INPUT L (I, J)
CIT-LIV SFIRȘIT
CIT-NUM-LIV SFIRȘIT
A SEQ
 PRINT "SITUAȚIA LIVRĂRILOR"
 PRINT
 PRINT "NR. DENUMIRE LIVRĂRI VALOARE"
 PRINT

```

c)

Fig. 6.14. Modulele de analiză și proiectare structurată: a) formatul datelor de ieșire; b) tabela de variabile; c) pseudocodul;

```

PRINT
PRINT
PRINT "-----"
T1=0
S1=0
A END
TOTAL PENTRU I DE LA 1 LA N
 S=0
 T=0
T1 PENTRU J DE LA 1 LA 3
T2 DACĂ L (I, J)<>0
 T=T+L(I, J) * P(J)
 S=S+L (I, J)
T2 SFIRȘIT
T1 SFIRȘIT
C SEQ
 S1=S1+S
 T1=T1+T
 PRINT
C END
TOTAL PRINT I; B$(I); S; T
F SFIRȘIT
 SEQ
 PRINT "-----"
 PRINT "TOTAL LIVRĂRI : " ; S1 ; "TONE"
 PRINT "-----"
 PRINT "TOTAL VALOARE : " ; T1 ; "LEI"
F END
TEMA END

```

c)

## TEXTUL CONVERSAȚIEI

```

RUN
INTRODUCEȚI NUMĂR BENEFICIARI ? 5
NUME BENEFICIAR? PECO-11
LIVRĂRI PECO-11 DIN REZERVORUL 1 ? 1
LIVRĂRI PECO-11 DIN REZERVORUL 2 ? 12
LIVRĂRI PECO-11 DIN REZERVORUL 3 ? 0
NUME BENEFICIAR? PECO-12
LIVRĂRI PECO-12 DIN REZERVORUL 1 ? 5
LIVRĂRI PECO-12 DIN REZERVORUL 2 ? 3
LIVRĂRI PECO-12 DIN REZERVORUL 3 ? 12
NUME BENEFICIAR? PECO-13
LIVRĂRI PECO-13 DIN REZERVORUL 1 ? 4
LIVRĂRI PECO-13 DIN REZERVORUL 2 ? 5
LIVRĂRI PECO-13 DIN REZERVORUL 3 ? 0
NUME BENEFICIAR? PECO-14
LIVRĂRI PECO-14 DIN REZERVORUL 1 ? 2
LIVRĂRI PECO-14 DIN REZERVORUL 2 ? 3
LIVRĂRI PECO-14 DIN REZERVORUL 3 ? 5
NUME BENEFICIAR? PECO-15
LIVRĂRI PECO-15 DIN REZERVORUL 1 ? 2
LIVRĂRI PECO-15 DIN REZERVORUL 2 ? 0
LIVRĂRI PECO-15 DIN REZERVORUL 3 ? 5

```

d)

Fig. 6.14. d) textul conversației.

● Pentru fiecare tip de jucărie fabricată, în cadrul fiecărui trimestru se calculează:

– procentul cantității totale trimestriale față de cantitatea anuală;  
 – abaterea cantității totale trimestriale (în procente) față de cantitatea medie trimestrială.

● Pentru fiecare articol se afișează:

– număr trimestru (I, II, III, IV);  
 – cantitatea totală trimestrială;  
 – procentul cantității trimestriale față de cantitatea anuală;  
 – abaterea cantității totale trimestriale (în procente) față de cantitatea medie trimestrială.

**Se reia problema din conversația precedentă complicind-o sub următoarele aspecte:**

● Se citește de la tastatură numărul de articole.  
 ● Se memorează codurile tuturor articolelor.  
 ● Pentru fiecare articol se memorează consumurile trimestriale normale.

● Într-o matrice cu patru linii (reprezentind trimestrele) și trei coloane (reprezentind lunile unui trimestru) se memorează consumurile trimestriale raportate ale unui articol identificat prin codul său.

● Pentru fiecare articol se va calcula abaterea față de consumurile trimestriale normale, rezultatele urmind a se memora într-o matrice cu 4 coloane reprezentind abaterile trimestriale și n linii reprezentind numărul articolelor (maximum 20).

Datele vor fi introduse dinamic iar rezultatele vor fi tipărite în formatul: COD, TRIMESTRUL 1, TRIMESTRUL 2, TRIMESTRUL 3, TRIMESTRUL 4.

## SOLUȚIA TEMEI 6

Nu răspundem.

Nu răspundem.

**Programele BASIC sînt:**

```
a) 10 DIM A(3,3), B(3,3), C(3,3), D(3,3), E(3,3)
 20 MAT READ A, B
 30 MAT C=A*B
 40 MAT D=INV (C)
 50 PRINT
 60 PRINT "INV (A*B)"
 70 MAT PRINT D
 80 MAT C=INV (A)
 90 MAT D=INV (B)
 100 MAT E=D*C
 110 PRINT
 120 PRINT "INV (B)*INV (A)"
 130 MAT PRINT E
```

```

140 DATA 1, -2, 3, 5, -1, -2, 0, 3, 4
150 DATA 2, -4, 0, -3, 1, 2, 5, 2, -5
160 END

b) 10 DIM A(3,3), B(3,3), C(3,3)
 20 MAT READ A
 30 MAT B=INV (A)
 40 PRINT "INV (A)"
 50 MAT PRINT B
 60 MAT C=A*B
 70 PRINT "A*INV (A)"
 90 MAT PRINT C
 100 MAT C=B*A
 110 PRINT
 120 PRINT "INV (A)*A"
 130 MAT PRINT C
 140 DATA 4, -4, 4, 1, 1, 7, -3, 9, -8
 150 END

c) 10 DIM A(1,3), B(3,4), D(1,4), E(1,4), C(4,1)
 20 MAT READ A, B, C
 30 MAT D=A*B
 40 MAT E=D*C
 50 MAT PRINT E
 60 DATA 1, 1, 1
 70 DATA 20, 700, 30, 40, 21, 10, 0, 400, 15
 80 DATA 2.05, .23, .30, .25
 90 END

e) 10 DIM C(10,10), K(10,1), I(10,10), S(10,1)
 20 READ N
 30 IF N=0 THEN 420
 40 MAT READ C(N,N), K(N,1)
 50 PRINT "MATRICEA COEFICIENTILOR"
 60 MAT PRINT C
 70 PRINT
 80 PRINT "TERMENII LIBERI"
 90 MAT PRINT K
 100 MAT I=INV (C)
 110 MAT S=I*K
 120 PRINT
 130 PRINT "SOLUȚIE"
 140 MAT PRINT S
 240 PRINT
 250 GO TO 20
 260 DATA 4
 270 DATA -3, 6, -5, -1
 280 DATA 1, 9, -5, -2
 290 DATA 1, 0, 6, 5, -7, 4, -1, 5, -32, 9, 2, -56
 310 DATA 5
 320 DATA 7, 6, -3, -1, 9
 330 DATA -9, 2, 9, 5, 1, -3, 4, 5, 5
 340 DATA 6, 0, -8, -2, -6
 350 DATA -3, -9, 5, 7, 3
 360 DATA 26.3, 91.1, 62.9, -55.6, -25.9
 370 DATA 2
 380 DATA -2, -5
 390 DATA -1, 4
 400 DATA -16, 31, 0
 420 END

k) 5 REM LL : LEU-LEU; SS : STEMA-STEMA; LS : LEU-STEMA SAU
 6 REM STEMA-LEU
 10 LL=0
 20 SS=0

```

```

30 LS=0
40 FOR I=1 TO 30
50 F1=INT (2*RND+1)
60 F2=INT (2*RND+1)
70 S=F1+F2
80 IF S=4 THEN
90 LL=LL+1
100 ELSE
110 IF S=2 THEN
120 SS=SS+1
130 ELSE
140 LS=LS+1
150 ENDIF
160 ENDIF
170 PRINT "SS"; TAB 10; "LT"; TAB(20); "SS"
180 PRINT SS; TAB 10; LT; TAB(20); SS
190 END

```

```

l) 10 INPUT "TITLUL LUCRĂRII"; N$
 40 WHILE N$ < > A$ AND N$ < > "xxx"
 50 READ A$, B$, C$, D$, E$
 60 IF N$=A$ THEN PRINT A$, B$, C$, D$, E$
 70 WEND
 80 DATA REBREANU, LIVIU, ION, EMINESCU, 1980
 90 DATA REBREANU, LIVIU, RĂSCOALA, EMINESCU, 1980
 100 DATA REBREANU, LIVIU, CIULEANDRA, EMINESCU 1981
 110 DATA xxx, xxx, xxx, xxx, xxx
 120 IF N$="xxx" THEN PRINT "AUTOR NECUNOSCUT"
 130 END

```

m) .

```

100 REM PROGRAM DE EDITARE A SITUĂȚILOR LIVRĂRILOR
110 REM DE BENZINĂ PRIVIND UN NUMĂR OARECARE (N)
120 REM DE BENEFICIARI
130 PRINT
140 DIM P(3), B$(20), L(20, 3)
150 REM P – PREȚURI PE TONĂ DE BENZINĂ
160 REM B$ – NUME BENEFICIARI
170 REM L – LIVRĂRI PENTRU FIECARE BENEFICIAR
180 REM N – NUMĂR DE BENEFICIARI
190 REM CITIRE NUMĂR BENEFICIARI
200 INPUT "INTRODUCEȚI NUMĂR BENEFICIARI"; N
210 PRINT
230 DATA 8750, 7500, 5250
240 FOR I=1 TO 3
250 READ P(I)
260 NEXT I
270 PRINT
280 REM CITIRE NUME ȘI LIVRĂRI BENEFICIAR
290 FOR I=1 TO N
300 INPUT "NUME BENEFICIAR"; B$(I)
310 FOR J=1 TO 3
320 PRINT "LIVRĂRI"; B$(I); "DIN REZERVORUL"; J
330 INPUT L(I, J)
340 NEXT J
350 NEXT I
360 REM TIPĂRIRE RAPORT LIVRĂRI
370 PRINT TAB(6), "SITUAȚIA LIVRĂRILOR"
380 PRINT
390 PRINT TAB(2); "NR"; TAB(5); "DENUMIRE"; TAB(21); "LIVRĂRI"; TAB(30);
400 PRINT "VALOARE"
410 PRINT

```

```

420 GOSUB 620
430 T1=C : S1=0
440 FOR I=1 TO N
450 S=0 : T=0
460 FOR J=1 TO 3
470 IF L(I, J)=0 THEN 500
480 T=T+L(I, J)*P(J)
490 S=S+L(I, J)
500 NEXT J
510 S1=S1+S
520 T1=T1+T
530 PRINT TAB(1); I; TAB(6); B$(I); TAB(24); S; TAB(30); T
540 NEXT I
550 GOSUB 620
560 PRINT " TOTAL LIVRARI : "; S1; "TONE"
570 GOSUB 620
580 PRINT " TOTAL VALOARE : "; T1; "LEI"
590 PRINT FOR I=1 TO 3
600 STOP
610 END
620 PRINT "-----"
630 RETURN

```

:RUN

INTRODUCETI NUMAR BENEFICIARI ?5

```

NUME BENEFICIAR ?PECO-1
LIVRARI PECO-1 DIN REZERVORUL 1
?100
LIVRARI PECO-1 DIN REZERVORUL 2
?20
LIVRARI PECO-1 DIN REZERVORUL 3
?30
NUME BENEFICIAR ?PECO-2
LIVRARI PECO-2 DIN REZERVORUL 1
?10
LIVRARI PECO-2 DIN REZERVORUL 2
?20
LIVRARI PECO-2 DIN REZERVORUL 3
?30
NUME BENEFICIAR ?PECO-3
LIVRARI PECO-3 DIN REZERVORUL 1
?10
LIVRARI PECO-3 DIN REZERVORUL 2
?11
LIVRARI PECO-3 DIN REZERVORUL 3
?12
NUME BENEFICIAR ?PECO-4
LIVRARI PECO-4 DIN REZERVORUL 1
?13
LIVRARI PECO-4 DIN REZERVORUL 2
?14
LIVRARI PECO-4 DIN REZERVORUL 3
?15
NUME BENEFICIAR ?PECO-5
LIVRARI PECO-5 DIN REZERVORUL 1
?23

```



LIVRARI PECO-5 DIN REZERVORUL 2  
 ?7  
 LIVRARI PECO-5 DIN REZERVORUL 3  
 ?0

\*\* SITUAȚIA LIVRARILOR \*\*

| NR | DENUMIRE | LIVRARI | VALOARE    |
|----|----------|---------|------------|
| 1  | PECO-1   | 150     | 1.1825E+06 |
| 2  | PECO-2   | 60      | 395000     |
| 3  | PECO-3   | 33      | 233000     |
| 4  | PECO-4   | 42      | 297500     |
| 5  | PECO-5   | 30      | 253750     |

TOTAL LIVRARI : 315 TONE

TOTAL VALOARE : 2.36175E+06 LEI

STOP AT LINE 600

- Programul pentru calculul abaterii cantității totale trimestriale de jucării, în procente față de cantitatea medie trimestrială este:

```

5 REM PROGRAMUL CITEȘTE DE LA TASTATURA NUMĂRUL DE
10 REM ARTICOLE (Z) ȘI CĂNTIȚĂȚILE LUNARE DE JUCĂRII
15 REM FABRICATE (C(K, I)).
20 REM PROGRAMUL CALCULEAZĂ ȘI AFIȘEAZĂ PROCENTUL CĂNTIȚĂȚII
25 REM TRIMESTRIALE (R1) FAȚĂ DE CĂNTIȚĂȚEA
30 REM ANUALĂ (T)
35 REM PROGRAMUL CALCULEAZĂ ȘI AFIȘEAZĂ ABATEREA CĂNTIȚĂȚII
40 REM TOTALE TRIMESTRIALE (R2) ÎN PROCENTE
45 REM FAȚĂ DE CĂNTIȚĂȚEA MEDIE TRI-
50 REM TRIMESTRIALĂ (M)
55 DIM C(8,12)
60 PRINT "INDICAȚI NUMĂR ARTICOLE (MAXIMUM 8)";
65 INPUT Z
70 FOR K=1 TO Z
72 PRINT "INDICAȚI CĂNTIȚĂȚILE LUNARE DE JUCĂRII"
75 FOR I=1 TO 12
78 PRINT "C("; K; ", "; I; ")=";
80 INPUT C(K, I)
82 NEXT I
85 NEXT K
90 PRINT
95 PRINT
100 PRINT
105 FOR K=1 TO Z
110 REM CALCUL CĂNTIȚĂȚEA ANUALĂ JUCĂRII (T)
115 T=0
120 FOR I=1 TO 12
125 T=T+C(K, I)
130 NEXT I
135 REM CALCUL CĂNTIȚĂȚEA MEDIE TRIMESTRIALĂ (M)
140 M=T/4
145 REM CALCULEAZĂ ȘI AFIȘEAZĂ RIND PENTRU
150 REM TRIMESTRUL 1
155 S=C(K,1)+C(K,2)+C(K,3)
160 R1=S*100/T
165 R2=(S-M)*100/M

```

```

170 PRINT "I", S, R1, R2
175 REM CALCULEAZĂ ȘI AFIȘEAZĂ RIND PENTRU
180 REM TRIMESTRUL 2
185 S=C(K,4)+C(K,5)+C(K,6)
190 R1=S*100/T
195 R2=(S-M)*100/M
200 PRINT "II", S, R1, R2
205 REM CALCULEAZĂ ȘI AFIȘEAZĂ RIND PENTRU
210 REM TRIMESTRUL 3
215 S=C(K,7)+C(K,8)+C(K,9)
220 R1=S*100/T
225 R2=(S-M)*100/M
230 PRINT "III", S, R1, R2
235 REM CALCULEAZĂ ȘI AFIȘEAZĂ RIND PENTRU
240 REM TRIMESTRUL 4
245 S=C(K,10)+C(K,11)+C(K,12)
250 R1=S*100/T
255 R2=(S-M)*100/M
260 PRINT "IV", S, R1, R2
265 PRINT
270 PRINT
275 NEXT K
280 END
RUN

```

INDICAȚI NUMĂR ARTICOLE (MAXIMUM 8) ? 2

INDICAȚI CANTITĂȚILE LUNARE DE JUCĂRII

|             |              |
|-------------|--------------|
| C(1,1)=? 5  | C(2,1)=? 50  |
| C(1,2)=?10  | C(2,2)=?100  |
| C(1,3)=?15  | C(2,3)=?150  |
| C(1,4)=?20  | C(2,4)=?200  |
| C(1,5)=?25  | C(2,5)=?250  |
| C(1,6)=?30  | C(2,6)=?300  |
| C(1,7)=?35  | C(2,7)=?350  |
| C(1,8)=?40  | C(2,8)=?400  |
| C(1,9)=?45  | C(2,9)=?450  |
| C(1,10)=?50 | C(2,10)=?500 |
| C(1,11)=?55 | C(2,11)=?550 |
| C(1,12)=?60 | C(2,12)=?600 |

|     |      |         |          |
|-----|------|---------|----------|
| I   | 30   | 7.6923  | -69.2307 |
| II  | 75   | 19.2307 | -23.0769 |
| III | 120  | 30.7692 | 23.0769  |
| IV  | 165  | 42.3076 | 69.2307  |
| I   | 300  | 7.62923 | -69.2307 |
| II  | 750  | 19.2307 | -23.0769 |
| III | 1200 | 30.7692 | 23.0769  |
| IV  | 1650 | 42.3076 | 69.2307  |

#### Remarci

- În linia 55 s-a definit matricea C (cu 8 linii și 12 coloane)
- Cantitățile lunare de jucării se introduc dinamic în matricea C, cu instrucțiunea INPUT din cadrul ciclurilor 70-90 și 75-90 (două bucle FOR-NEXT).
- Cantitatea medie trimestrială se calculează cu instrucțiunea

140 M=T/4

în care T (cantitatea anuală de jucării) a fost calculat cu secvența de instrucțiuni 105-130.

- Valorile lui R1 și R2 se calculează cu relațiile:

$$R1 = S * 100 / T,$$

$$R2 = (S - M) * 100 / M$$

în care S se determină pentru fiecare trimestru în parte cu instrucțiunile de atribuire corespunzătoare (v. liniile 155, 185, 215, 245).

- Deoarece instrucțiunile 160, 165; 190, 195; 220, 225 și 250, 255 sint identice se recomandă gruparea acestora într-o subrutină ce urmează a fi apelată din punctele respective.

- Programul pentru calculul abaterii consumurilor trimestriale raportate față de consumurile trimestriale normate ale mai multor articole este:

```

10 PRINT ' PROGRAMUL CITEȘTE CONSUMURILE TRIMESTRIALE NORMATE (N)'
15 PRINT ' ȘI CONSUMURILE TRIMESTRIALE RAPORTATE (L) ALE MAI '
20 PRINT ' MULTOR ARTICOLE (Z), CALCULEAZĂ ȘI TIPĂREȘTE ABATEREA (A) '
25 PRINT ' FAȚĂ DE CONSUMURILE TRIMESTRIALE NORMATE '
30 PRINT
35 PRINT
40 PRINT
45 DIM C(8), L(4,3), A(8,4), N(4)
50 PRINT ' INDICAȚI NUMĂRUL DE ARTICOLE '
55 INPUT Z
60 PRINT 'INDICAȚI CODURILE ARTICOLELOR'
65 FOR I=1 TO Z
70 INPUT C(I)
75 NEXT I
80 FOR K=1 TO Z
85 PRINT "INDICAȚI CONSUMURILE TRIMESTRIALE NORMATE PENTRU"
90 PRINT "ARTICOLUL CU CODUL:", C(K)
95 FOR I=1 TO 4
100 INPUT N(I)
105 NEXT I
110 PRINT "INDICAȚI CONSUMURILE TRIMESTRIALE RAPORTATE PENTRU"
115 PRINT "ARTICOLUL CU CODUL" C(K)
120 FOR I=1 TO 4
125 FOR J=1 TO 3
130 INPUT L (I, J)
135 NEXT J
140 NEXT I
145 REM 'CALCUL ABATERE'
150 FOR I=1 TO 4
155 FOR J=1 TO 3
160 IF J = < > 1 THEN 170
165 T=0
170 T=T+L(I, J)
175 IF J < > 3 THEN 185
180 A(K, I)=(T-N(I)) * 100/N(I)
185 NEXT J
190 NEXT I
195 NEXT K
200 REM 'TIPĂRIREA CAP TABEL'
205 PRINT ' ABATERE % '
210 PRINT
215 PRINT ' COD TRIMESTRUL 1 TRIMESTRUL 2 TRIMESTRUL 3 TRIMESTRUL 4'
220 PRINT
225 PRINT
230 PRINT
235 REM ' TIPĂRIRE REZULTATE '
240 FOR I=1 TO Z

```

```

245 PRINT C(I),
250 FOR J=1 TO 4
255 PRINT A (I, J),
260 NEXT J
265 NEXT I
270 STOP
275 END

```

**RUN**

INDICAȚI NUMĂRUL DE ARTICOLE ? 8

INDICAȚI CODURILE ARTICOLELOR ? 1, 2, 3, 4, 5, 6, 7, 8

INDICAȚI CONSUMURILE TRIMESTRIALE NORMATE PENTRU  
ARTICOLUL CU CODUL 1

5  
10  
15  
20

INDICAȚI CONSUMURILE TRIMESTRIALE RAPORTATE PENTRU  
ARTICOLUL CU CODUL 1

3  
8  
.  
.  
.

ABATERE %

| COD | TRIMESTRUL 1 | TRIMESTRUL 2 | TRIMESTRUL 3 | TRIMESTRUL 4 |
|-----|--------------|--------------|--------------|--------------|
| .   | .            | .            | .            | .            |
| .   | .            | .            | .            | .            |
| .   | .            | .            | .            | .            |
| 3   | 885          | 60           | 58.4615      | 16.1538      |
| 4   | 1912         | 1155.25      | 170.769      | 433.338      |
| .   | .            | .            | .            | .            |
| .   | .            | .            | .            | .            |
| .   | .            | .            | .            | .            |

#### Remarci

- În linia 45 s-au definit vectorii C, N și matricile L (4 linii și 3 coloane), A (8 linii și 4 coloane).
- De reținut procedura de calcul a abaterii pe care o vom decupa din programul principal.

```

150 FOR I=1 TO 4
155 FOR J=1 TO 3
160 IF J < > 1 THEN 170
165 T=0
170 T=T+L(I, J)
175 IF J < > 3 THEN 185
180 A(K, I)=(T-N (I)) * 100/N (I)
185 NEXT J
190 NEXT I

```

- Pentru înțelegerea programului recomandăm **simularea execuției acestuia cu creionul și hirtia**. Cel care vă va verifica va fi evident; calculatorul cu care lucrați.

## CONVERSAȚIA 7

Proiectarea și realizarea unui program BASIC pentru crearea unui fișier secvențial de livrări. Instrucțiuni BASIC-AMSTRAD, COMMODORE, BASIC-80, BASIC-PLUS, ABASIC pentru crearea fișierelor de organizare secvențială. Programe utile pentru gestionarea fondurilor de date în fișiere. JOCURI, APLICAȚII și TESTE pentru cititor



## EXEMPLELE 7-PC, m, M, F

### □ De la problemă la program

Pină acum, toate datele folosite de către programele noastre au fost introduse prin instrucțiunile **INPUT**, **READ-DATA** sau **LET**. Cu consecvență, am depozitat datele ca parte integrantă a programului sau le-am introdus direct de la tastatură. Această manieră de lucru se pretează foarte bine pentru un volum mic de date pe care dorim să-l prelucrăm o singură dată.

În cazul în care se dorește prelucrarea unui volum mare de date sau este necesară întreținerea acestui fond de date se impune ca datele să fie separate de program structurându-le în așa-numitele **fișiere de date**.

Ce este un fișier? **Fișierele** reprezintă un mijloc simplu de a manevra structuri de date, ce pot fi citite și întreținute prin programe (**BASIC**). În general, un fișier poate fi asociat cu o mulțime organizată de informații.

Utilizarea fișierelor permite modificarea, sortarea, interclasarea datelor în concordanță cu specificațiile de programare. Posibilitățile sînt limitate numai prin capacitatea noastră de a gândi modul de rezolvare informatică a problemelor.

Fișierele de date sînt folosite pentru gestiunea stocurilor, sistemele de producție, sistemul de personal, proiectarea asistată de calculator, într-un cuvînt în majoritatea activităților profesionale.

Una din trăsăturile acestor fișiere care le fac „misterioase” este aceea că sînt invizibile. Prelucrarea datelor structurate în fișiere nu presupune și listarea de fiecare dată a conținutului acestora, deși este bine ca programele să furnizeze și anumite liste de control pentru a putea urmări ceea ce prelucreză calculatorul. După ce veți vedea cîteva exemple, vă veți putea convinge că într-adevăr calculatorul efectuează operațiile dorite.

Ceea ce trebuie reținut este faptul că putem introduce/extrage sau modifica datele organizate în fișiere sub controlul programelor. Fișierele **BASIC** pot fi **secvențiale** (un șir continuu de articole, memorate fizic pe suport în ordinea în care au fost scrise de programator – v. **BASIC-AMSTRAD**, **COMMODORE**, **BASIC-80**, **BASIC-PLUS**, **ABASIC**), **masive virtuale** (masive de date rezidente pe un fișier disc – v. **BASIC-PLUS**), **relative** (v. **BASIC-COMMODORE**) și **selective** (în acces direct, v. **BASIC-AMSTRAD**, **COMMODORE**, **BASIC-80**, **BASIC-PLUS** – fișiere articol).

Pentru a ne familiariza cu modul în care datele pot fi introduse într-un fișier vă propunem să proiectăm și să realizăm un program scurt, folositor, prin care să creăm pe un suport magnetic (dischetă, disc) fișierul privind livrările de benzină ale unei stații de benzină și uleiuri efectuate pentru diverși beneficiari.

Se vor introduce în fișier datele privind: codul beneficiarului, cantitatea de benzină livrată și prețul în lei pe tona de benzină. Numărul beneficiarilor nu este dinainte cunoscut. Convenim ca introducerea datelor în fișier să fie oprită în momentul cînd tastăm pentru codul beneficiarului valoarea 99 (un cod artificial).

Limbajul pe care îl vom utiliza este analog celui folosit pentru un fișier manual: **deschide** fișierul, **scrie** un element, **închide** fișierul.

De notat că toate elementele unei înregistrări a fișierului privind: codul, cantitatea livrată și prețul se scriu succesiv pe dischetă separate prin virgulă.

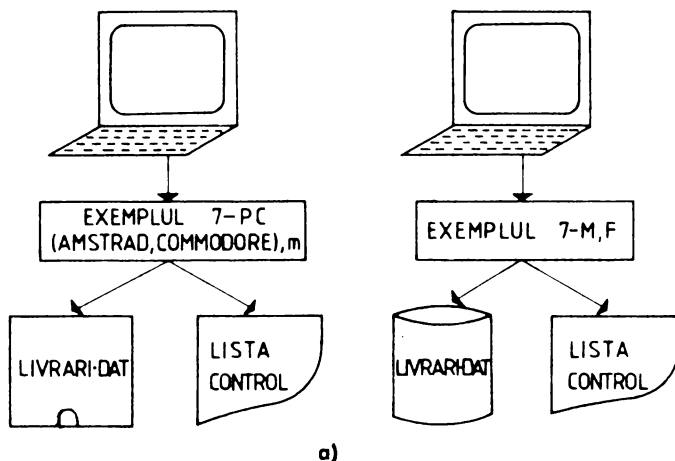
### □ Specificații de programare

**Schema de sistem a EXEMPLELOR 7-PC (AMSTRAD, COMMODORE), m, M, F** este ilustrată în modulele de analiză și proiectare structurată, figura 7.1. Programul citește de la tastatură codul beneficiarului, cantitatea livrată, prețul tonei de benzină și le introduce în fișierul LIVRARI. DAT de pe dischetă/disc. De remarcat prezența LISTEI DE CONTROL ce conține înregistrările fișierului creat.

**Structura înregistrării de ieșire.** O înregistrare a fișierului LIVRARI. DAT este structurată logic în trei cimpuri: cod-beneficiar, cantitate-livrată, preț-tonă-benzină. Datele sînt separate fizic prin delimitatorul " , ".

**Datele de intrare** se transmit de la tastatură.

Tabela de variabile, specificațiile de programare și alocarea funcțiilor de prelucrare sînt definite în modulele de analiză și proiectare structurată, figura 7.1.




---

FORMATUL DATELOR DE IEȘIRE

---

Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F

---

COD

CANT

PREȚ

---

b)

Fig. 7.1. Modulele de analiză și proiectare structurată: a) schema de sistem; b) formatul datelor de ieșire;

## DATE INTRARE

Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE) m, M, F

| COD BENEFICIAR | CANTITATE BENZINĂ<br>LIVRATĂ (tone) | PREȚ TONĂ<br>(lei) |
|----------------|-------------------------------------|--------------------|
| 10             | 20                                  | 8.000              |
| 30             | 30                                  | 7.500              |
| 20             | 40                                  | 8.000              |
| 99             | 99                                  | 99                 |

c)

## TABELA DE VARIABLE

Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F

| Variabile de intrare    | Variabile de stare | Variabile de ieșire      |
|-------------------------|--------------------|--------------------------|
| COD: cod beneficiar     |                    | COD: cod beneficiar      |
| CANT: cantitate livrată |                    | CANT: cantitate livrată  |
| PREȚ: preț tonă benzină |                    | PREȚ: preț, tonă benzină |

d)

## SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F

## Descrierea programului

Programul creează un fișier secvențial pe dischetă (disc). Fișierul conține date referitoare la livrările de benzină efectuate pentru mai mulți beneficiari.

## Intrări

Date privind codul beneficiarului, cantitatea livrată, preț.

## Ieșiri

Fișier pe dischetă (disc).

## Lista de funcțiuni ale programului:

1. Deschidere fișier
2. Afișare mesaj introducere date
3. Citire date
4. Înscrisoare în dischetă (disc) a datelor citite
5. Închidere fișier
6. Afișare mesaj sfârșit creare fișier
7. Stop

e)

## ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

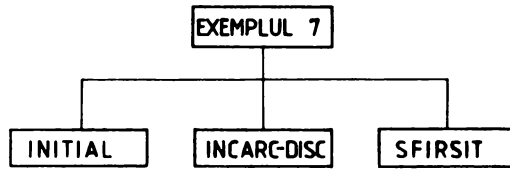
Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F

| Modul       | Funcțiuni |
|-------------|-----------|
| INIȚIAL     | 1, 2, 3   |
| INCARC-DISC | 4, 3      |
| SFIRȘIT     | 5, 6, 7   |

f)

Fig. 7.1. c) date intrare; d) tabela de variabile; e) specificații de programare; f) alocarea funcțiilor de prelucrare;





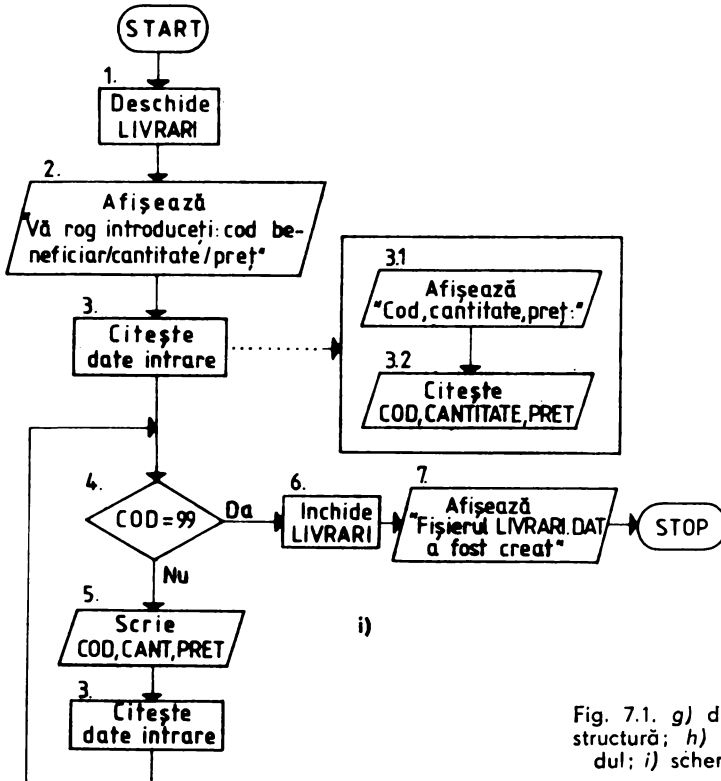
g)

PSEUDOCODUL

Nume program: EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F

|             |                                                              |
|-------------|--------------------------------------------------------------|
| EXEMPLUL7   | SEQ                                                          |
| INIȚIAL     | SEQ                                                          |
|             | Deschide LIVRARI                                             |
|             | Afișează "Vă rog introduceți: cod beneficiar/cantitate/preț" |
|             | Afișează un rind de spații                                   |
|             | Citește date intrare                                         |
| INIȚIAL     | END                                                          |
| INCARC-DISC | CIT TIMP COD < > 99                                          |
|             | Scrie COD, CANT, PREȚ                                        |
|             | Citește date intrare                                         |
| INCARC-DISC | SFIRȘIT                                                      |
|             | Inchide LIVRARI                                              |
|             | Afișează "Fișierul LIVRARI. DAT a fost creat"                |
| EXEMPLUL7   | END                                                          |

h)



i)

Fig. 7.1. g) diagrama de structură; h) pseudocodul; i) schema logică.

## □ Documentația de proiectare

**Diagrama de structură.** Schema din figura 7.1 (g) completată cu tabelul pentru alocarea funcțiilor de prelucrare redă structura logică a EXEMPLELOR 7-PC (AMSTRAD, COMMODORE), m, M, F. De notat prezența unor funcțiuni noi: deschidere fișiere, închidere fișiere.

**Pseudocodul și schema logică.** Pseudocodul pentru EXEMPLELE 7-PC (AMSTRAD, COMMODORE), m, M, F și schema logică sînt prezentate în modulele de analiză și proiectare structurată, fig. 7.1.

### Observații

- Orice fișier BASIC, înainte de a fi prelucrat, trebuie deschis iar după aceea închis.
- Pentru crearea unui fișier BASIC se recomandă utilizarea structurii de iterație **CIT TIMP** (v. cazul fișierelor vide).

## □ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 227

Programul BASIC este prezentat în volumul 2, pagina 227. În cele ce urmează vom analiza numai instrucțiunile din liniile 30, 80 și 120.

### OPEN – Deschideți fișierele!

Înainte de a scrie într-un caiet, registru, carnet, agendă etc. nu-i așa că acestea trebuie mai întii deschise? După ce ați terminat de scris ceea ce v-ați propus urmează operația de închidere a „memoriilor externe” folosite. În mod similar se petrec lucrurile și cu fișierele pe suport magnetic.

### Instrucțiunea OPENOUT

30 OPENOUT "LIVRARI . DAT"

Instrucțiunea **OPENOUT** are drept scop deschiderea canalului de transfer asociat unui fișier ASCII secvențial de ieșire – LIVRARI . DAT de pe dischetă/disc. Ea trebuie declarată înaintea oricărei alte instrucțiuni care se referă la înregistrările unui fișier. Cu alte cuvinte, fiecare fișier folosit de un program BASIC trebuie mai întii „deschis”.

Formatul instrucțiunii **OPENOUT** este:

|                                |
|--------------------------------|
| Format general                 |
| <b>OPENOUT</b> <nume – fișier> |

### Scriere pe dischetă

○ dată citite informațiile privind: cod-beneficiar, cantitate, preț-tonă benzină urmează înscrierea acestora pe dischetă în fișierul secvențial, cu organizare secvențială – LIVRARI . DAT. Pentru aceasta se folosește instrucțiunea **WRITE**.

### Instrucțiunea WRITE #

80 WRITE # 9, COD, CANT, PREȚ

De notat că, după cuvântul rezervat **WRITE**, se specifică numărul canalului pentru floppy disc – 9, urmat de variabilele COD, CANT și PREȚ separate prin virgule.

Formatul instrucțiunii **WRITE** este:

|                                               |
|-----------------------------------------------|
| Format general                                |
| <b>WRITE</b> [# (nr. canal),][listă expresii] |

### Tehnica plasării instrucțiunii INPUT

Când întâlnești un program structurat este bine să studiezi cu atenție plasarea și manevrarea instrucțiunii **INPUT**. Întrebările obișnuite privind plasarea instrucțiunii **INPUT** pentru programul BASIC-AMSTRAD pot fi: „De ce s-a plasat o instrucțiune **INPUT** o dată înainte de bucla **WHILE-WEND** și apoi la sfârșitul acesteia?” „De ce nu plasăm instrucțiunea **INPUT** doar la începutul buclei **WHILE-WEND**?”

În figura 7.2 este prezentată imaginea unui astfel de program.

Iată răspunsul: un program scris în această manieră va lucra foarte bine pînă ce ultima înregistrare va fi citită. Apoi, o dublură sau o înregistrare-rest vor fi scrise în fișierul de pe dischetă.

Principiul cheie care trebuie reținut este acela că variabila care indică sfârșitul fișierului trebuie testată imediat după instrucțiunea **INPUT**

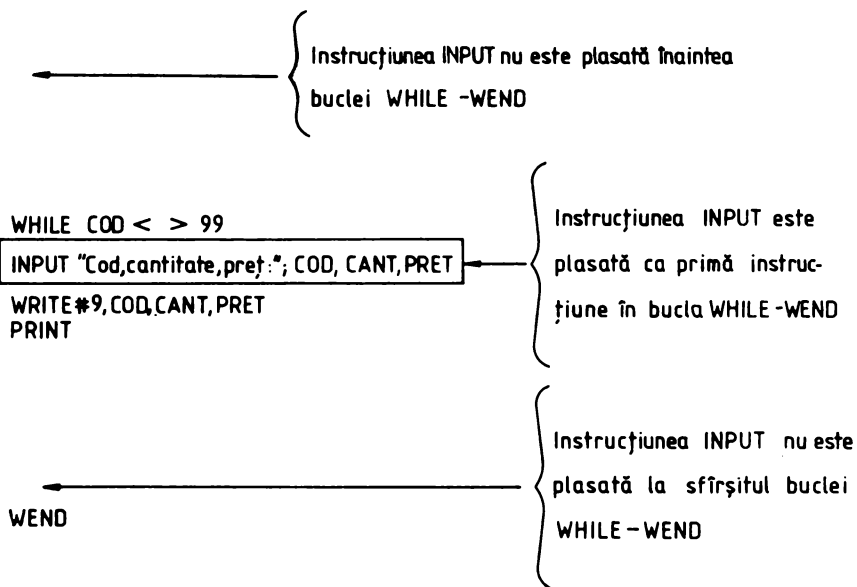


Fig. 7.2. Plasarea eronată a instrucțiunii **INPUT**.

(operație de citire). În cadrul programului, variabila COD este testată de instrucțiunea **WHILE COD < > 99** (linia 70).

Prin urmare, înainte a primei prelucrări, o înregistrare trebuie deja să fie citită. Acest **INPUT** inițial a fost plasat apoi la sfârșitul buclei **WHILE-WEND**.

*Observație.* Acesta nu este singurul mod de plasare a instrucțiunii **INPUT**, dar el este relativ ușor de înțeles și utilizat pentru programele BASIC simple. Într-o asemenea manieră vom întrebuința și noi tehnica plasării instrucțiunii **INPUT**.

### **CLOSE – închideți fișierele!**

În momentul în care s-a citit sfârșitul fișierului de date (COD=99), controlul programului este trecut la instrucțiunea:

120 **CLOSEOUT**

care realizează închiderea fișierului de ieșire **LIVRĂRI . DAT**.

### **Instrucțiunea CLOSEOUT**

Instrucțiunea închide canalul de transfer asociat fișierului de ieșire (**LIVRĂRI . DAT**). Formatul instrucțiunii este:

|                 |
|-----------------|
| Format general  |
| <b>CLOSEOUT</b> |

*Observație.* Instrucțiunea **CLOSEOUT** se execută numai pentru fișierele care au fost deschise în cadrul programului.

#### **Aplicații**

1) Simulați funcționarea programului precizând totodată și rezultatele execuției programului.

2) Plasați instrucțiunea **INPUT** din linia 60 în linia 75 a programului și eliminați din program linia 100. Ce-ați constatat?

3) Modificați programul **BASIC-AMSTRAD** astfel încât pe dischetă să se scrie pe lângă: **COD**, **CANT**, **PREȚ** și valoarea livrărilor (**CANT \* PREȚ**).

## **Codificarea în limbajul BASIC-COMMODORE**

vol. 2, pag. 227

Diferențele de scriere a programului **BASIC-COMMODORE** apar în toate cele trei instrucțiuni **OPEN**, **PRINT**, **CLOSE**.

### **Instrucțiunea OPEN**

Spre deosebire de **BASIC-AMSTRAD**, în **BASIC-COMMODORE** pentru instrucțiunea **OPEN** se specifică numărul fișierului (poate avea valori

între 1 și 255), numărul dispozitivului (8 pentru floppy disc), numărul canalului de transfer (de la 2 la 14 inclusiv) și numele fișierului.

15 **OPEN** 15, 8, 15

20 **OPEN** 2, 8, 2, "LIVRĂRI.DAT, S, W"

Formatul instrucțiunii este:

---

Format general

---

**OPEN** <nr. fișier>, <dispozitiv>, <nr. canal>, "<nume-fișier, tip, R/W>"

---

De notat că la deschiderea unui fișier secvențial, cel de-al patrulea parametru are în structura sa nume fișier, tip fișier: **P** pentru program (PRG); **S** pentru fișier secvențial (SEQ); **R** pentru fișier relativ (REL), operație: **R** (READ) sau **W** (WRITE).

În tabelul 7.1 sînt prezentate dispozitivele periferice și codurile asociate acestora iar în tabelul 7.2 sînt ilustrate operațiile I/O și codurile corespunzătoare pentru fiecare dispozitiv în parte.

Tabelul 7.1

| Dispozitiv  | Cod dispozitiv |
|-------------|----------------|
| Tastatură   | 0              |
| Casetofon   | 1              |
| RS-232      | 2              |
| Monitor     | 3              |
| Imprimantă  | 4              |
| Imprimantă  | 5              |
| Floppy disc | 8              |

Tabelul 7.2

| Dispozitiv  | Cod operație | Operație       |
|-------------|--------------|----------------|
| Casetofon   | 0            | Citare         |
|             | 1            | Scriere        |
|             | 2            | Scriere*       |
| Floppy disc | 1-14         | Canal transfer |
|             | 15           | Comandă canal  |
| Imprimantă  | 0            | sus/grafice    |
|             | 7            | sus/jos        |

**Notă:** \* scriere END OF TAPE

În programul BASIC-COMMODORE instrucțiunea **OPEN** din linia 15 deschide pentru fișierul cu numărul 15, de pe dischetă (8), canalul 15 (instrucțiunea este folosită pentru testarea erorilor de canal). Cit privește următoarea instrucțiune **OPEN**, aceasta deschide pentru fișierul cu nr. 2 ce se va crea în ieșire (output) pe dischetă (8), canalul nr. 2 asociat fișierului LIVRĂRI.DAT, de tip secvențial (**S**) pentru care se realizează operația de scriere **W** (WRITE).

## Instrucțiunea PRINT #

În BASIC-COMMODORE, scrierea datelor într-un fișier secvențial se realizează cu instrucțiunea **PRINT #**.

```
100 PRINT # 2, K%/10; A$; C2; A$; C3
```

**PRINT #** nu compactează datele, ci le scrie așa cum ar fi afișate de **PRINT**. Pentru a obține un cîmp compact de date în fișier se utilizează ca delimitator în (lista) caracterul "; ". Dacă se utilizează ", " se scriu pe disc și spațiile care ar apare la afișare.

Pentru a forma explicit șirurile pe disc, acestea trebuie explicit separate prin delimitatori (" ; " ; " , " ; "[CR]").

Formatul instrucțiunii este:

|                                        |
|----------------------------------------|
| Format general                         |
| <b>PRINT #</b> <nr. fișier>[, <listă>] |

## Instrucțiunea CLOSE

În BASIC-COMMODORE închiderea unui fișier se realizează cu instrucțiunea **CLOSE** urmată de numărul fișierului declarat în instrucțiunea **OPEN**.

```
120 CLOSE 2 : CLOSE 15
```

Formatul instrucțiunii este:

|                           |
|---------------------------|
| Format general            |
| <b>CLOSE</b> <nr. fișier> |

**Aplicație.** Introduceți și executați următoarele programe:

- |                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                              |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>1. 5 OPEN 15, 8, 15    10 OPEN 5, 8, 5, "STUDENT, S, W"    20 A\$="DANIEL" : B\$="ANDREI"    30 PRINT # 5, A\$    40 PRINT # 5, B\$    50 CLOSE 5, 15    60 END</pre> | <pre>30 READ CANT (I) 40 READ PREȚ (I) 50 V(I)=CANT(I) * PREȚ(I) 60 NEXT I 70 DATA P-1, 30, 20, P-2, 50, 60,    P-3, 20, 10 80 DATA P-4, 15, 20, P-5, 16, 10,    R-6, 3, 9 85 X\$=CHR\$(13) 90 OPEN 15, 8, 15 100 OPEN 2, 8, 2, "AX1, S, W" 110 FOR I=1 TO 6 120 PRINT # 2, B\$(I); X\$; CANT(I);    X\$; PREȚ(I); X\$; V(I) 130 NEXT I 140 CLOSE 2 : CLOSE 15 150 END</pre> |
| <pre>2. 5 OPEN 15, 8, 15    10 OPEN 5, 8, 5, "STUDENT", S, W"    20 A\$="DANIEL" : B\$="ANDREI"    30 X\$=CHR\$(13)    40 PRINT # 5, A\$, X\$, B\$    50 CLOSE 5, 15</pre> |                                                                                                                                                                                                                                                                                                                                                                              |
| <pre>3. 10 FOR I=1 TO 6    20 READ B\$(I)</pre>                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                              |

## □ Particularități ale programării in limbajul BASIC-80

vol. 2, pag. 228

Față de programele din exemplele precedente apar diferențe la scrierea instrucțiunilor **OPEN**, **PRINT** și **CLOSE**.

### Instrucțiunea **OPEN**

30 **OPEN** "O", # 1, "LIVRARI.DAT"

Instrucțiunea **OPEN** alocă un buffer pentru operațiile de I/O și determină modul de acces. Formatul instrucțiunii este:

---

Format general

---

**OPEN** "<MOD>", [#]<nr. fișier>, "<nume-fișier>", [<lung-inreg>]

---

unde, <MOD> este un șir de expresii în care primul caracter poate fi: O (mod de ieșire secvențial); I (mod de intrare secvențial); R (mod I/O în acces direct);

<nr. fișier> este o expresie întregă cu valori de la 1 la 15. Numărul este asociat fișierului pe toată perioada cât acesta este deschis și folosit în instrucțiunile de I/O;

<lung-inreg> este o expresie care, atunci când este precizată, stabilește lungimea înregistrării pentru operații de I/O în acces direct (implicit, lungimea unei înregistrări se consideră 128 octeți).

*Observație.* Într-un program BASIC-80 același fișier poate fi deschis sub mai multe numere pentru citire secvențială sau acces direct, dar numai cu un singur număr pentru scriere secvențială.

### Instrucțiunea **PRINT #**

80 **PRINT #**1, COD; " , " ; CANT; " , " ; PREȚ

**PRINT #** scrie date într-un fișier secvențial BASIC-80.

Formatul instrucțiunii este:

---

Format general

---

**PRINT #**<nr. fișier>[**USING**(expresie-șir);]<listă-expresii>

---

în care:

<nr. fișier> este numărul folosit la deschiderea fișierului;

<expresie-șir> (v. instrucțiunea **PRINT USING**);

#### Remarci

- **PRINT #** nu compactează datele;
- Pentru a forma explicit șirurile pe disc, acestea trebuie explicit separate prin delimitatori.

**TEST**

Precizați rezultatul execuției următoarelor programe:

- |                                                                                                                                                           |                                                                                      |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| <p>a) 10 OPEN "O", # 1, "xx"<br/>         20 READ A\$, B\$<br/>         30 DATA CELEBRUL, 702<br/>         40 PRINT # 1, A\$; B\$<br/>         50 END</p> | <p>30 READ A\$, B\$<br/>         40 PRINT #1, A\$; ", "; B\$<br/>         50 END</p> |
| R.                                                                                                                                                        |                                                                                      |
| <p>b) 10 OPEN "O", # 1, "yy"<br/>         20 DATA CELEBRUL, 702</p>                                                                                       | <p>a) CELEBRUL702<br/>         b) CELEBRUL,702</p>                                   |

**Instrucțiunea CLOSE**

120 CLOSE # 1

Instrucțiunea închide fișierul #1 de pe dischetă.  
 Formatul instrucțiunii este:

|                                                     |
|-----------------------------------------------------|
| Format general                                      |
| <b>CLOSE [#](nr. fișier)[,[#](nr. fișier) ...]]</b> |

**Remarci**

- (nr. fișier) este numărul sub care fișierul a fost deschis.
- **CLOSE** fără argumente închide toate fișierele deschise.
- Fișierul poate fi deschis sub același număr sau sub alt număr.
- Instrucțiunile **END** și **NEW** închid automat toate fișierele (instrucțiunea **STOP** face excepție).

**Aplicație.** Introduceți și executați următorul program:

|                                                                                                                 |                                                                                                        |
|-----------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|
| <p>10 OPEN "O", # 1, "AGENDA"<br/>         20 INPUT "NUME"; N\$<br/>         30 IF N\$=" " THEN CLOSE : END</p> | <p>40 INPUT "TELEFON" : " ; T\$<br/>         50 PRINT #1, N\$; ", " ; T\$<br/>         60 GO TO 20</p> |
|-----------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------|

**Observație.** După cum remarcați, programul nu este scris structurat. Varianta structurală a acestuia este:

|                                                                                                                                                            |                                                                                                                     |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| <p>10 OPEN "O", # 1, "AGENDA"<br/>         20 INPUT "NUME" : " ; N\$<br/>         30 WHILE N\$ &lt; &gt; " "<br/>         40 INPUT "TELEFON" : " ; T\$</p> | <p>50 PRINT #1, N\$; ", " ; T\$<br/>         60 INPUT "NUME" : " ; N\$<br/>         70 WEND<br/>         80 END</p> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------|

**Particularități ale programării  
 în limbajul BASIC-PLUS**

vol. 2, pag. 228

**Fișiere BASIC-PLUS**

Din punct de vedere al structurii și metodei de acces fișierele BASIC-PLUS pot fi: fișiere secvențiale ASCII, fișiere masive virtuale și fișiere articol. Pentru a înțelege mai bine clasificarea de mai sus va trebui să definim noțiunea de *articol*. Prin *articol* înțelegem elementul unui fișier ce se



poate trata și identifica (logic/fizic) ca o entitate. De notat că suportul fizic extern al fișierelor BASIC-PLUS poate fi: banda perforată, cartela perforată, hîrtia de imprimantă, ecranul monitoarelor, banda magnetică, discul magnetic. Suporturile de informație pot fi orientate fie pe articol (ex. linia de imprimantă), fie pe fișier (disc, bandă magnetică).

**Fișierele secvențiale ASCII** reprezintă cea mai simplă metodă de structurare a datelor pe un suport extern. Datele din structura acestor fișiere sînt reprezentate în format ASCII fiind separate între ele (în interiorul unui articol) prin virgulă. De notat că un articol dintr-un fișier ASCII secvențial este identic cu articolul creat de utilizator în mod conversațional, printr-o instrucțiune **INPUT**.

Pentru exploatarea fișierelor ASCII secvențiale puteți folosi următorul set de instrucțiuni BASIC-PLUS: **OPEN**, **CLOSE**, **INPUT#**, **INPUT LINIE#** (instrucțiune de citire a unui articol ca o singură dată), **PRINT#**, **PRINT# . . . USING**, **MAT INPUT#** (instrucțiune matricială de citire a unui masiv dintr-un fișier), **MAT PRINT** (instrucțiune matricială de scriere a unui masiv pe un fișier), **KILL** (instrucțiune de eliminare a unui fișier ASCII secvențial).

**Fișierele masive virtuale** sînt masive rezidente pe un fișier disc, ca extensie a memoriei interne, ale căror dimensiuni [26] sînt limitate numai de mărimea spațiului disponibil pe discul respectiv.

Pentru exploatarea fișierelor masive virtuale puteți folosi următorul set de instrucțiuni BASIC-PLUS: **DIM#** (instrucțiune pentru declararea masivelor virtuale), **OPEN . . . AS VIRTUAL** (instrucțiune pentru deschiderea unui fișier virtual), **CLOSE**.

**Fișierele articol** prezintă particularitatea că un articol poate conține date de tipuri diferite, indiferent de ordinea lor de dispunere. De notat că în cazul fișierelor articol toate operațiile I/O se realizează pe blocuri fizice (512 octeți).

Pentru exploatarea fișierelor articol puteți folosi următorul set de instrucțiuni BASIC-PLUS: **OPEN**, **GET#** (instrucțiune pentru citirea unui articol al fișierului), **PUT#** (instrucțiune pentru scrierea unui articol), **FIELD#** (instrucțiune pentru identificarea dinamică a cîmpurilor din articol), **LSET**, **RSET** (instrucțiuni de transfer al datei în zona de alocare a variabilei).

## Identificarea fișierelor BASIC-PLUS

Un fișier BASIC-PLUS se poate identifica prin: **nume extern** (specificator de fișier) și **nume intern** (număr de fișier). Specificatorul de fișier asigură identificarea fișierului pe suportul extern de către sistemul de operare, iar numărul de fișier asigură identificarea fișierului de către programul BASIC.

Formatul general al unui specificator de fișier [26] este:

---

Format general

---

{Suport} : [{grup}, {membru}]{nume} · {tip}; {versiune}

---

unde:

{suport} reprezintă unitatea periferică fizică (logică) pe care s-a încărcat volumul ce conține fișierul;

**<grup>**, **<membru>** reprezintă codul utilizator care precizează catalogul în care este introdus fișierul pe suport;  
**<nume>** este numele fișierului sub care acesta este identificat pe suport;  
**<tip>** precizează natura conținutului fișierului;  
**<versiune>** indică numărul de versiune al fișierului care se exploatează.

### Reguli

- Numele pentru **<suport>** este alcătuit din două caractere alfabetice urmate, opțional, de un număr octal de una sau două cifre: CR (cititor de cartele), DK (disc), MM (banda magnetică), LP (imprimantă), TT (terminal).
- Codurile pentru **<grup>**, **<membru>** sînt numere octale în intervalul 0–377.
- **<nume>** este alcătuit din 9 caractere alfanumerice.
- **<tip>** este format din 3 caractere: **BAC** (fișier program în format sursă), **DAT** (fișier de date), **TMP** (fișier temporar ce se elimină la încheierea sesiunii utilizatorului).
- **<versiune>** este un număr octal cu valoarea cuprinsă între 0 și 7777. În cazul în care nu se precizează se consideră: numărul de versiune maxim existent (pentru fișierele de intrare), numărul de versiune maxim existent plus o unitate (pentru fișierele de ieșire).

*Observație.* Pentru suporturile orientate pe articol, specificatorul de fișier trebuie să conțină numai parametrul **<suport>** (numele suportului) restul informațiilor fiind ignorate.

### Înapoi la program

Programul este prezentat în volumul 2, pagina 228. În cele ce urmează vom face referiri numai asupra instrucțiunilor **OPEN**, **PRINT#** și **CLOSE**.

### Instrucțiunea OPEN

20 **OPEN "LIVRARI.DAT" FOR OUTPUT AS ASCII & FILE 1 TO WRITE**

Instrucțiunea **OPEN** servește la deschiderea unui canal de transfer asociat fișierului (v. specificatorul de fișier). De asemenea, prin intermediul instrucțiunii **OPEN** se realizează corespondența dintre numele extern și numele intern ale fișierului.

Formatul instrucțiunii este:

---

#### Format general

---

**OPEN** <specificator-fișier> [**FOR INPUT** | **OUTPUT**] **AS** [**ASCII**] **FILE** (nr. canal)  
**TO** [**READ** | **WRITE** | **APPEND**]

---

unde:

<specificator-fișier> este orice expresie de tip șir, avînd ca valoare un nume extern de fișier. Se plasează între apostrofuri;

(nr. canal) reprezintă numărul canalului de transfer ce se asociază fișierului. Poate fi orice expresie numerică întregă cuprinsă între 1 și 8 inclusiv.

### Reguli

- **READ** (v. formatul general) indică deschiderea fișierului pentru citire. Ulterior se poate folosi numai instrucțiunea **INPUT** (fișiere ASCII).
- **WRITE** indică deschiderea fișierului pentru scriere. Ulterior se poate folosi numai instrucțiunea **PRINT** (fișiere ASCII).

- **APPEND** indică deschiderea unui fișier numai la scriere prin extinderea fișierului (fișiere ASCII).
- **FOR INPUT** presupune deschiderea unui canal pentru un fișier deja creat.
- **FOR OUTPUT** presupune deschiderea unui canal pentru crearea unui fișier.
- Instrucțiunea **OPEN** fără **FOR INPUT** sau **FOR OUTPUT** realizează deschiderea unui canal pentru un fișier existent sau pentru crearea unui nou fișier (dacă fișierul specificat nu există).
- Deschiderea canalului pentru prelucrarea unui fișier are ca efect activarea unui buffer de I/O.
- Fișierele ASCII secvențiale se pot deschide numai în mod **READ**, **WRITE**, **APPEND**. Revenind la instrucțiunea **OPEN** din linia 20, remarcăți că ea servește la deschiderea canalului de transfer 1 asociat fișierului ASCII secvențial. "LIVRARI.DAT" deschis în mod **WRITE**, pentru crearea (**FOR OUTPUT**) acestuia.

### Instrucțiunea PRINT #

```
100 PRINT # 1, K%; ", "; C2; ", " ; C3
```

**PRINT #** realizează transferul datelor din memoria internă a calculatorului pe un suport orientat pe articol sau fișier. De notat că transferul datelor respectă regulile de editare ale instrucțiunii **PRINT**. Formatul instrucțiunii este:

| Format general                                    |
|---------------------------------------------------|
| <b>PRI [NT] #</b> (expresie-numerică) [, (listă)] |

unde:

(expresie-numerică) indică numărul canalului de transfer (0÷8) asociat fișierului;

(lista) reprezintă o listă de elemente (admise de limbaj) separate între ele prin " , " sau prin " ; ".

### Instrucțiunea CLOSE

```
120 CLOSE 1
```

Execuția unei instrucțiuni **CLOSE** are ca efect închiderea canalului de transfer deschis prin instrucțiunea **OPEN**.

Formatul instrucțiunii este:

| Format general                              |
|---------------------------------------------|
| <b>CLO [SE]</b> (exp. 1) [, (exp. 2)] ... ] |

unde:

(exp. 1), (exp. 2) ... reprezintă orice expresie numerică a cărei parte întregă reprezintă un număr de canal cuprins între 1 și 8 inclusiv.

#### Reguli

- Orice canal deschis pentru un fișier trebuie în mod obligatoriu închis.
- Absența parametrilor (exp. 1), (exp. 2) ... dintr-o instrucțiune **CLOSE** are ca efect închiderea tuturor canalelor de transfer la acel moment.

**Aplicație.** Să se creeze un fișier ASCII secvențial care să conțină temperaturile zilnice din mai multe stații meteorologice. Structura înregistrării fișierului este: Cod – stație, denumire – stație, temperatură – luni, temperatură – marți, ..., temperatură – duminică.

### Joc pe HC-85, TIM S, SPECTRUM

```

10 REM DE LA 1 LA 5
100 DIM a(3,3)
105 LET c$="123456789"
108 PRINT AT 8,4;"SELECTAȚI!";
 TAB 15;"TABLOU"
110 PRINT AT 10,6;"—";TAB 16;"—"
120 FOR i=1 TO 3
125 LET i2=i*2:LET i3=(i-1)*3
130 PRINT AT i2+9,5;"I";i3+1;i3+2;
 " ";i3+3;"I";TAB 15;"I I"
135 PRINT AT i2+10,5;"I I";TAB 15;"I I"
140 NEXT i
150 PRINT AT 16,5;"—";TAB 15;"—"
160 FOR a=1 TO 10
170 LET i=INT(RND*9)+1:
 GO SUB 400
180 NEXT a
190 LET m=0:LET t=0
200 LET t$=INKEY$
210 IF t$="" THEN GO TO 200
220 FOR i=1 TO 9
230 IF t$=c$(i) THEN GO TO 260
240 NEXT i
250 GO TO 200
260 LET m=m+1
270 GO SUB 400
280 IF t<9 THEN GO TO 200
290 CLS:PRINT FLASH 1;"Ați reușit!"
300 PRINT:PRINT "IN";m;
 "INCERCĂRI"
310 STOP
400 LET x=INT((i-1)/3)+1
410 LET y=-3*(x-1)
420 LET a(x,y)=a(x,y)-1
430 FOR i=1 TO 3
440 LET a(x,i)=a(x,i)+1
450 LET a(i,y)=a(i,y)+1
460 NEXT i
470 LET t=0
480 FOR i=1 TO 3
490 FOR j=1 TO 3
500 IF a(i,j)=6 THEN LET a(i,j)=0
510 IF a(i,j)=0 THEN LET t=t+1
520 PRINT AT 11+(i-1)*2,16+(j-1)*2;
 a(i,j)
530 NEXT j
540 NEXT i
550 RETURN

```

### Particularități ale programării în limbajul ABASIC

vol. 2, pag. 229

#### Fișiere ABASIC

Fișierele ABASIC sînt de tip *secvențial* cu lungimea maximă a înregistrării de 140 octeți. În număr de cel mult 8, fișierele ABASIC pot fi folosite doar în mod scriere (**PRINT/WRITE**) sau citire (**INPUT/LINPUT**). Fișierele se identifică printr-un nume precizat în instrucțiunea **OPEN** (respectă convențiile de sintaxă ARIEL) căruia i se asociază un număr intern (precedat opțional de "#").

De notat că fișierele ABASIC sînt constituite din șiruri de caractere de lungime variabilă, avînd structura liniilor de terminal utilizate.

#### Înapoi la program

Programul este prezentat în volumul 2, pag. 229. Vom analiza și de această dată instrucțiunile **OPEN**, **WRITE#**, **CLOSE**.

#### Instrucțiunea OPEN

```
30 OPEN #1,"LIVRARI.DAT"
```

Formatul instrucțiunii este:

| Format general                                     |
|----------------------------------------------------|
| <b>OPEN</b> [#](expresie-numerică) "<nume-fișier>" |

unde:

<expresie-numerică> reprezintă numărul întreg (1–8) prin care va fi referit fișierul;

<nume-fișier> este identificatorul fișierului (șir de caractere plasat între apostrofuri ce respectă convențiile de sintaxă ARIEL).

### Instrucțiunea **WRITE #**

80 **WRITE #1, COD, GANT, PREȚ**

Instrucțiunea are același efect ca și cele de scriere directă la terminal (**PRINT, PRINT USING**).

Formatul instrucțiunii este:

| Format general                                 |
|------------------------------------------------|
| <b>WRITE #</b> (expresie-numerică) [, (listă)] |

unde: (listă) reprezintă o listă de elemente admise de limbajul ABASIC, separate între ele prin ", " sau prin "; ".

### Instrucțiunea **CLOSE**

120 **CLOSE #1**

1. instrucțiunea închide fișierul #1. Ea are formatul:

| Format general                                      |
|-----------------------------------------------------|
| <b>CLOSE</b> [[#](expresie 1) [, (expresie 2) ...]] |

unde prin (expresie 1), (expresie 2) ... se precizează identificatorul intern al fișierelor a căror închidere este cerută.

#### Reguli

- **CLOSE** fără parametri are ca efect închiderea tuturor fișierelor deschise prin instrucțiunea **OPEN**.
- Dacă nu se folosește **CLOSE**, fișierele deschise în scriere (**PRINT/WRITE**) sint șterse la execuția uneia din comenzile: **RUN, LOAD, NEW, EXIT**.

**Aplicație.** Să se creeze un fișier ABASIC necesar generării următorului raport:

| AUTOR         | TITLUL LUCRĂRII      | EDITURA | ANUL APARIȚIEI | COTA CĂRȚII |
|---------------|----------------------|---------|----------------|-------------|
| Patrubani, N. | Microprocesorul Z 80 | TEHNICĂ | 1989           | 1-BS-203    |
| .             | .                    | .       | .              | .           |
| .             | .                    | .       | .              | .           |

## TEMA 7

 Răspundeți prin **DA** sau **NU** la următoarele întrebări:

- Pentru prelucrarea unui volum mic de date de către un program BASIC se utilizează instrucțiunile: **INPUT, LET, READ-DATA.**
- Pentru prelucrarea unui volum mare de date se impune ca datele să fie separate de program și să se organizeze în fișiere.
- Într-un fișier secvențial articolele sînt dispuse secvențial și devin accesibile tot secvențial.
- Într-un fișier ASCII secvențial, datele sînt separate în interiorul unui articol prin punct și virgulă.
- Într-un fișier se pot introduce date numerice, șiruri de caractere sau combinații ale acestora, separate prin spații sau virgulă.
- Canalul este unitatea de schimb de informație între un fișier și un program BASIC.
- În BASIC-AMSTRAD numărul canalului pentru unitatea de floppy-disc este 9.
- În BASIC-COMMODORE numărul canalului de transfer are valori de la 2 la 14 inclusiv.
- În BASIC-PLUS numărul canalului de transfer are valori de la 1 la 8 inclusiv.
- Înainte de a scrie într-un fișier, acesta trebuie deschis cu instrucțiunea **OPEN.**
- În limbajul BASIC-AMSTRAD, instrucțiunea **OPENOUT** are drept scop deschiderea canalului de transfer asociat unui fișier de intrare.
- Într-o instrucțiune **OPEN** din limbajul BASIC-COMMODORE se specifică numărul fișierului, numărul dispozitivului, numărul canalului de transfer și numele fișierului.
- În BASIC-COMMODORE numele fișierului din instrucțiunea **OPEN** nu trebuie să conțină obligatoriu tipul fișierului (**S, R**) și nici tipul operației (**R, W**).
- În instrucțiunea **OPEN** din BASIC-80 se indică opțional modul I/O prin una din literele: **O, I** sau **R.**
- Fișierele ASCII secvențiale BASIC-PLUS se pot deschide numai în mod **READ, WRITE, APPEND.**
- Fișierele ABASIC sînt de tip secvențial și selectiv.
- Instrucțiunea **CLOSE** realizează închiderea unui singur fișier.
- În BASIC-AMSTRAD scrierea datelor într-un fișier secvențial pe dischetă se realizează cu instrucțiunea **WRITE # 8.**
- În ABASIC scrierea datelor într-un fișier secvențial pe disc se realizează cu instrucțiunile **PRINT #/WRITE #.**

 Înlocuiți cuvintele care lipsesc din următoarele propoziții:

- a) Fișierul reprezintă ..... Fișierele BASIC pot fi .....
- b) Pentru deschiderea unui fișier de ieșire se utilizează instrucțiunea .....

..... în limbajul BASIC-AMSTRAD și instrucțiunile .....  
 ..... în BASIC-COMMODORE, BASIC-PLUS, BASIC-80,  
 ABASIC.

c) În limbajul BASIC-COMMODORE instrucțiunea

10 OPEN 3, 8, 3, "AGA.S, W"

indică .....

d) În limbajul BASIC-80 instrucțiunile:

10 OPEN "O", #2, "AGA.DAT"

20 OPEN "I", #2, "AGA.DAT"

30 OPEN "R" #2, "AGA.DAT"

precizează .....

e) În limbajul BASIC-PLUS instrucțiunile:

10 OPEN "A.DAT" FOR OUTPUT AS ASCII & FILE 1 TO WRITE

20 OPEN "A.DAT" FOR INPUT AS ASCII & FILE 1 TO READ

precizează .....

f) În limbajul BASIC-PLUS opțiunea **READ** într-o instrucțiune **OPEN** indică  
 ....., iar opțiunea **WRITE** indică .....

g) În BASIC-80, BASIC ....., instrucțiunea **OPEN**  
 alocă un buffer pentru operațiile de I/O.

h) În BASIC-PLUS "numele extern" pentru un fișier asigură identificarea  
 de către sistemul de operare a fișierului pe suportul extern, iar "numele  
 intern" .....

i) În limbajul BASIC-PLUS transferul de date între memoria internă și un  
 suport orientat pe articol sau fișier se realizează prin instrucțiunea .....

j) În limbajul BASIC-PLUS instrucțiunea **OPEN** fără opțiunea **FOR INPUT**  
 sau **FOR OUTPUT** realizează .....

k) Instrucțiunea **CLOSEOUT** închide canalul de transfer asociat fișierului  
 de ieșire în limbajul BASIC .....

l) În limbajul BASIC-80, BASIC ..... instrucțiunea **CLOSE** fără argu-  
 mente închide toate fișierele deschise.

m) Un fișier BASIC-PLUS se poate identifica prin ..... și .....

n) Secvența de instrucțiuni BASIC-PLUS:

10 OPEN "A.DAT" FOR OUTPUT AS

ASCII & FILE 1 TO WRITE

20 INPUT COD, PREȚ

30 PRINT #2, COD; ", "; PREȚ

40 CLOSE 3

50 END

conține erori în liniile .....

o) Secvența de instrucțiuni ABASIC:

10 OPEN #1, "A.DAT"

20 WHILE A <> 999

30 INPUT A

40 WRITE #1, A

50 WHILE

60 CLOSE #2

70 END

conține erori în liniile .....

**p) Secvența de instrucțiuni BASIC-AMSTRAD:**

```

10 OPENOUT "A. DAT"
20 INPUT "A, B", A, B
30 WHILE A <> 999
40 PRINT #9, A, B
50 INPUT A, B
60 WEND
70 CLOSE #9

```

conține erori în liniile .....

**r) Secvența de instrucțiuni BASIC-80:**

```

10 OPEN "O", #1, "AX"
20 INPUT "A, B, C"; A, B, C
30 IF A=999 THEN CLOSE : END
40 PRINT #1, A; ", "; B; ", "; C
50 GO TO 20

```

poate fi scrisă structurat sub forma:

**s) Secvența de instrucțiuni BASIC-80:**

```

5 DATA 1, 2, 3, 4, 5, 6, 7, 8
10 OPEN "O", #1, "B"
20 FOR I=1 TO 4
30 READ X, Y
40 PRINT #1, X; ", "; Y
45 PRINT X, Y
50 NEXT I
60 CLOSE 1
70 END

```

are ca efect .....

**Scrieți cite un program BASIC pentru fiecare din problemele de mai jos:**

**a)** Să se creeze un fișier dicționar de organizare secvențială în limbajele: a) BASIC-AMSTRAD; b) BASIC-COMMODORE; c) BASIC-80; d) BASIC-PLUS; e) ABASIC. Structura fișierului este: cod, denumire.

**b)** Prestigioasa revistă A.M.C. (Automatică, Management, Calculatoare) ce apare în cadrul Editurii Tehnice a împlinit în anul 1986, 25 ani de la apariție. Să se alcătuiască un fișier de organizare secvențială cu toți autorii care au publicat în revistă. Pe lângă numele autorilor, fișierul (creat în BASIC-AMSTRAD, BASIC-COMMODORE, BASIC-80, BASIC-PLUS, ABASIC) va mai conține: titlul articolului, număr AMC, anul apariției, număr pagini autor, redactor.

**c)** Să se creeze un fișier de organizare secvențială cu coeficienții și exponenții necesari calculului vitezei de așchiere la strunjire.

**d)** Să se creeze un fișier de organizare secvențială, cu zilele onomastice ale celor dragi. Structura fișierului este: Nume, Sex, Ziua de naștere, Telefon, Adresă.

**e)** Pentru gestionarea sălilor de curs și seminar dintr-un institut de învățământ superior s-a procedat la crearea unui fișier de organizare secvențială cu următoarea structură: cod-sală; destinație (curs/seminar); nume-



cadru didactic ce ocupă sala; denumire disciplină ce se predă în sală; ora (ex.: 14–16); număr-semestru (1/2). Programul de creare se va realiza în limbajele: BASIC–AMSTRAD, BASIC–COMMODORE, BASIC-80, BASIC–PLUS, ABASIC.

- Să se gestioneze pe un suport magnetic** (dischetă/disc) fondul de date privind cantitățile lunare de jucării fabricate de întreprinderea URȘULEȚUL. Fișierul se va numi JUCĂRII și va avea organizare secvențială. Structura înregistrării este: cod articol, a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3, preț. Se va proiecta și realiza un program BASIC.
- Să se gestioneze pe un suport magnetic** (dischetă/disc) fondul de date privind consumurile de materiale (articole) normate și realizate din țară și din import pentru activitatea de reparații utilaje grele. Fișierul se va numi CONSUMURI și va avea organizare secvențială. Structura înregistrării este: cod articol (material), cod proveniență (R.S.R./IMPORT), consum normat, consum realizat, preț de achiziție. Se va proiecta și realiza un program BASIC.

## SOLUȚIA TEMEI 7

- Nu răspundem
- Nu răspundem
- Nu răspundem
- Programul BASIC este:**

```

10 REM * Crearea fișier JUCĂRII IN BASIC-80 *
20 PRINT
30 OPEN "O", #1, "JUCĂRII.DAT"
40 PRINT "COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREȚ" ;
50 PRINT
60 INPUT COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREȚ
70 WHILE COD <> 999
80 WRITE #1, COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREȚ
90 PRINT
100 PRINT "COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREȚ" ;
110 INPUT COD, A1, A2, A3, B1, B2, B3, C1, C2, C3, D1, D2, D3, PREȚ
120 WEND
130 CLOSE #1
140 PRINT "Fișierul JUCĂRII.DAT a fost creat"
150 END

```

- Programul BASIC este:**

```

a) 10 REM * Creare fișier CONSUMURI în BASIC-AMSTRAD *
20 PRINT
30 OPENOUT "CONSUMURI . DAT"
40 PRINT "cod, prov, cn, cr, preț";

```

```
50 INPUT cod, prov, cn, cr, preț
60 WHILE cod < > 999
70 WRITE #9, cod, prov, cn, cr, preț
80 PRINT
90 PRINT "cod, prov, cn, cr, preț";
100 INPUT cod, prov, cn, cr, preț
110 WEND
120 CLOSEOUT
130 PRINT "Fișierul CONSUMURI . DAT a fost creat"
140 END

b) 10 REM * Creare fișier CONSUMURI IN BASIC-80, GWBASIC *
20 PRINT
30 OPEN "O", # 1, "CONSUMURI.DAT"
40 PRINT "cod, prov, cn, cr, preț"
50 INPUT cod, prov, cn, cr, preț
60 WHILE cod < > 999
70 WRITE # 1, cod, prov, cn, cr, preț
80 PRINT
90 PRINT "cod, prov, cn, cr, preț"
100 INPUT cod, prov, cn, cr, preț
110 WEND
120 CLOSE # 1
130 PRINT "Fișierul CONSUMURI.DAT a fost creat"
140 END
```

## CONVERSAȚIA 8

Proiectarea și realizarea unui program BASIC ghidat de un meniu pentru actualizarea, consultarea și listarea unui fișier secvențial de livrări. Instrucțiunile INKEY\$, GET, ERASE, OPENIN, CLOSEIN, WRITE# și ON... GOTO. Funcția EOF. Fișiere cu acces direct. Depanarea unui program. JOCURI, APLICAȚII și TESTE pentru cititor



## EXEMPLELE 8 – PC, m, M, F

### De la problemă la program

În cadrul acestei conversații vom "umbla" în fișierul LIVRĂRI.DAT (creat în conversația precedentă) în ideea familiarizării dvs. cu instrucțiunile BASIC pentru prelucrarea fișierelor de organizare secvențială.

Vom proiecta și realiza un program ghidat de un *menu* care să ofere posibilitatea optării pentru una din următoarele (șase) funcțiuni care se afișează efectiv pe ecran:

- |                               |                       |
|-------------------------------|-----------------------|
| 1) Încarcă fișier LIVRĂRI.DAT | 4) Actualizare fișier |
| 2) Salvează fișier            | 5) Listare fișier     |
| 3) Căutare în fișier          | 6) End                |

Fiecare opțiune va corespunde unei subrutine care realizează funcțiunea afișată.

Singurul „efort” care se cere utilizatorului acestui program este acela de a tasta cifra corespunzătoare opțiunii dorite, fără a-l mai „chinui” de-a acționa imediat tasta [CR]. Pentru reîntoarcerea în *menu* se va tasta de fiecare dată [CR].

Pentru fiecare din cele șase funcțiuni ale programului se vor afișa mesaje de avertizare, corespunzătoare.

Dacă, de exemplu, utilizatorul va opta pentru încărcarea fișierului LIVRĂRI.DAT de pe dischetă în memoria calculatorului (opțiune obligatorie la începutul execuției programului) programul (subrutina) va afișa mesajul „Fișier încărcat”.

Dacă utilizatorul va opta pentru funcțiunea „căutare în fișier”, programul (subrutina) va afișa un mesaj prin care să se solicite codul beneficiarului. În urma căutării secvențiale a beneficiarului cu codul indicat în fișierul LIVRĂRI.DAT se va preciza dacă acesta a fost sau nu găsit.

Cît privește actualizarea fișierului (v. opțiunea 4) se va modifica valoarea cantității de benzină livrate. În funcție de codul beneficiarului se va actualiza cîmpul corespunzător (CANT) cu o nouă valoare. În caz de eroare (beneficiarul nu se află în fișier) se va afișa un mesaj explicativ.

Pentru listarea fișierului (înainte de actualizare, sau după actualizare) se va scrie o subrutină care să corespundă opțiunii 5.

Dacă utilizatorul a ales funcțiunea 2, programul (subrutina) va salva pe suportul magnetic respectiv ultima versiune a fișierului LIVRĂRI.DAT. În urma salvării se va afișa mesajul „salvat fișier”.

Pentru oprirea execuției programului se va opta pentru funcțiunea 6 căreia i se va prevedea, de asemenea, în cadrul programului o subrutină.

### Specificații de programare

Schema de sistem, formatul datelor de ieșire, structura înregistrării de ieșire, tabela de variabile și specificațiile de programare sînt ilustrate în modulul de analiză structurată, fig. 8.1.

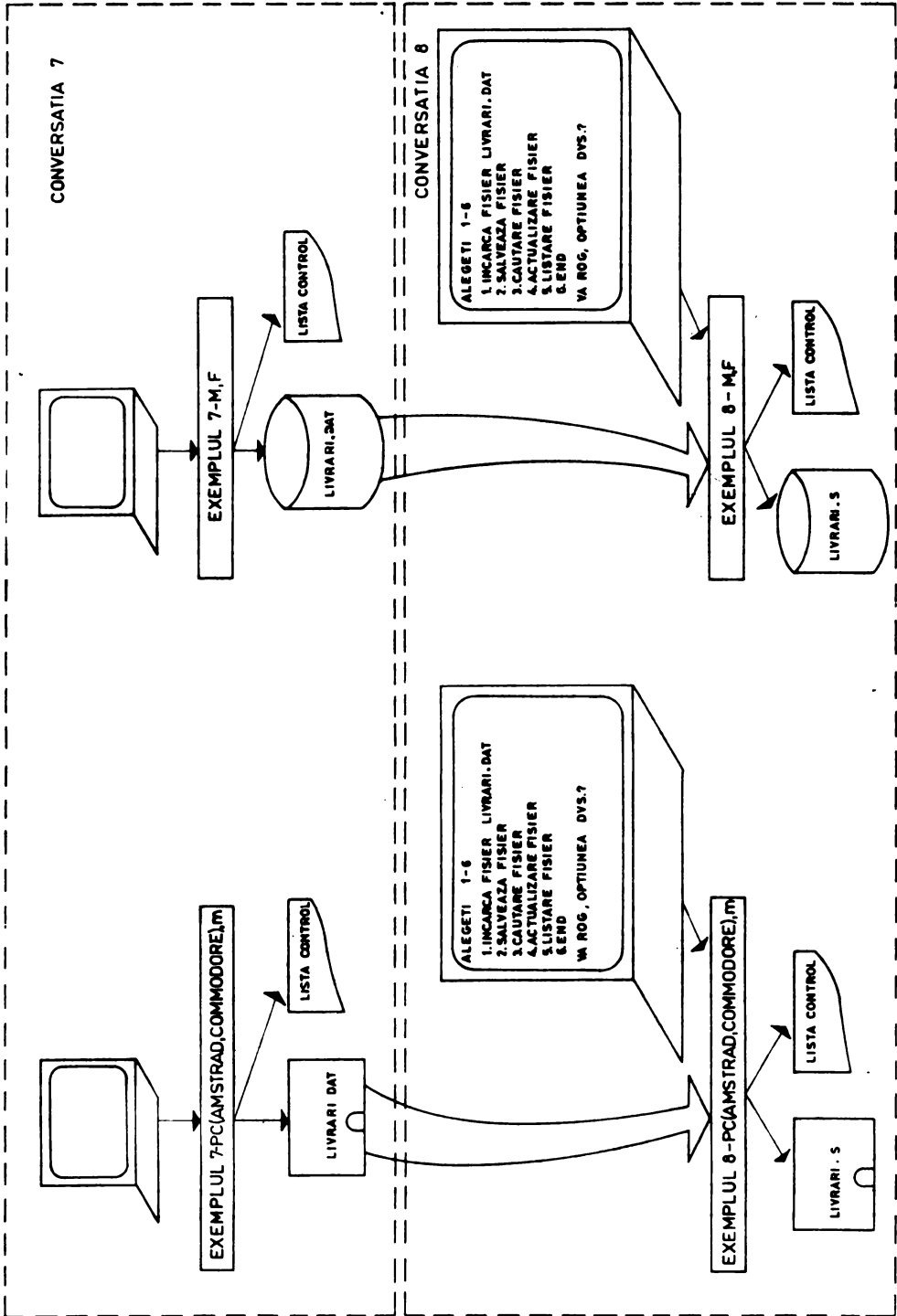


Fig. 8.1. Modulul de analiză structurată: a) schema de sistem.

## FORMATUL DATELOR DE IEȘIRE

Nume program: EXEMPLELE 8-PC (AMSTRAD, COMMODORE), m, M, F

| Cod | Cantitate | Preț |
|-----|-----------|------|
| **  | ***       | **   |
| **  | ***       | **   |
| .   | .         | .    |
| .   | .         | .    |
| **  | ***       | **   |

b)

| COD | CANTITATE | PREȚ |
|-----|-----------|------|
|-----|-----------|------|

c)

## SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 8-PC (AMSTRAD, COMMODORE), m, M, F

## Descrierea programului

Programul citește de la terminal 1÷6 opțiuni ale utilizatorului privind: încărcarea fișierului de livrări, căutarea în fișier, actualizarea fișierului, listarea fișierului și salvarea fișierului pe dischetă (disc). Fiecare opțiune corespunde unei subrutine care realizează funcțiunea precizată.

## Intrări

Fișierul LIVRARI.DAT

## Ieșiri

Fișierul LIVRARI.S

## Lista de funcțiuni ale programului

1. Inițializare variabilă OPTIUNE\$
2. Inițializare variabilă FISINCARC
3. Inițializare variabilă I
4. Inițializare variabilă J
5. Schimbare valoare (1) FISINCARC
6. Tastare [CR]
7. Ștergere ecran
8. Deschidere fișier livrări
9. Citire date intrare
10. Închidere fișier livrări
11. Afișare mesaj fișier încărcat
12. Deschidere fișier livrări salvat

13. Închidere fișier livrări salvat
14. Inițializare variabilă C
15. Salvare date
16. Incrementare variabilă C
17. Afișare mesaj fișier salvat
18. Citire cod beneficiar (CBEN)
19. Inițializare variabilă GASIT (0)
20. Citește date intrare
21. Reinițializare variabilă GASIT (1)
22. Afișare mesaj căutare fișier
23. Afișare rînd spații
24. Citire cod beneficiar actualizare (CODB)
25. Citire cantitate livrată actualizare
26. Afișare mesaj actualizare fișier
27. Tipărire date intrare
28. Ieșire din meniu
29. Afișare meniu
30. Specificarea opțiunii, după cum urmează:
  - 1) Încarcă fișier LIVRARI.DAT
  - 2) Salvează fișier
  - 3) Căutare în fișier
  - 4) Actualizare fișier
  - 5) Listare fișier
  - 6) END
31. Inițializare COD, CANT, PREȚ
32. Stop.

d)

Fig. 8.1. b) formatul datelor de ieșire; c) structura înregistrării de intrare/ieșire; d) specificații de programare;

## TABELA DE VARIABILE

Nume program: EXEMPLELE 8-PC (AMSTRAD, COMMODORE), m, M, F

| Variabile de intrare             | Variabile de stare                           | Variabile de ieșire     |
|----------------------------------|----------------------------------------------|-------------------------|
| OPȚIUNES: opțiune utilizator     | I: index cimpuri fișier                      | COD: cod beneficiar     |
| COD: cod beneficiar              | FISINCAR: stare fișier (încărcat/neîncărcat) | CANT: cantitate livrată |
| CANT: cantitate livrată          | C: Index cimpuri fișier                      | PREȚ: preț tonă benzină |
| PREȚ: preț tonă benzină          | GĂȘIT: stare cod beneficiar (gășit/negășit)  |                         |
| CODB: cod actualizare beneficiar |                                              |                         |
| CBEN: cod căutare beneficiar     |                                              |                         |

e)

## ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 8 - PC (AMSTRAD, COMMODORE), m, M, F

| Modul       | Funcțiuni                                |
|-------------|------------------------------------------|
| INIT        | 1,2                                      |
| EDIT-M      | 29                                       |
| OPȚIUNE     | 30                                       |
| INCARC-2000 | 7, 3, 31, 3, 9, 4, 10, 11, 5, 6          |
| SALV-3000   | 7, 12, 13, 14, 16, 17, 6, 5              |
| CAUT-4000   | 7, 18, 14, 19, 21, 16, 22, 6             |
| ACT-5000    | 7, 24, 14, 16, 21, 19, 24, 25, 26, 23, 6 |
| LIST-6000   | 7, 3, 4, 27, 20, 10, 8, 6                |
| END         | 28                                       |

f)

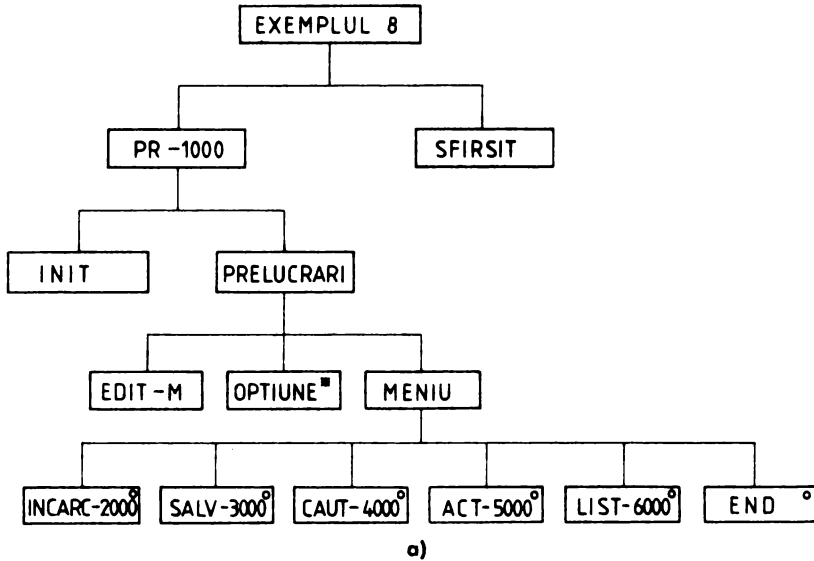
| Mesaj                                                      | Funcție                       | Subrutina |
|------------------------------------------------------------|-------------------------------|-----------|
| == Fișier încărcat == (tastați [CR])                       | 1. Încarcă fișier LIVRĂRI.DAT | 2000      |
| == Salvat fișier == (tastați [CR])                         | 2. Salvează fișier            | 3000      |
| Precizați cod beneficiar                                   | 3. Căutare în fișier          | 4000      |
| Beneficiarul xxx nu este în fișier                         |                               |           |
| Beneficiarul cu codul xxx se află în fișier (tastați [CR]) | 4. Actualizare fișier         | 5000      |
| Pentru ce cod doriți actualizarea                          |                               |           |
| Care este noua valoare a cantității livrate                |                               |           |
| Beneficiarul xxx nu este în fișier (tastați [CR])          | 5. Listare fișier             | 6000      |
| (tastați [CR])                                             | 6. End                        |           |

g)

Fig. 8.1. e) tabela de variabile; f) alocarea funcțiilor de prelucrare; g) mesaje subrutine.

## □ Documentația de proiectare

Diagrama de structură, schema logică și pseudocodul sint prezentate în modulul de proiectare structurată, fig. 8.2.



### PSEUDOCODUL

Nume program: EXEMPLUL 8 – PC (AMSTRAD, COMMODORE), m, M, F

|            |                                       |
|------------|---------------------------------------|
| EXEMPLUL 8 | SEQ                                   |
|            | EXECUTĂ 1000                          |
| EXEMPLUL 3 | END                                   |
| 1000       | SEQ                                   |
| INIT       | SEQ                                   |
|            | OPTIUNES=" "                          |
|            | FISINCARC=0                           |
| INIT       | END                                   |
| EDIT-M     | CIT TIMP OPTIUNES ≠ 6                 |
| A1         | SEQ                                   |
|            | PRINT "Alegeți 1-6"                   |
|            | PRINT                                 |
|            | PRINT "1. Incarcă fișier LIVRĂRI.DAT" |
|            | PRINT "2. Salvează fișier"            |
|            | PRINT "3. Căutare în fișier"          |
|            | PRINT "4. Actualizare fișier"         |
|            | PRINT "5. Listare fișier"             |
|            | PRINT "6. End"                        |
|            | PRINT "Vă rog opțiunea dvs."          |
| A1         | END                                   |
| EDIT-M     | SFIRȘIT                               |

b)

Fig. 8.2. Modulul de proiectare structurată: a) diagrama de structură; b) pseudocodul;



## PSEUDOCODUL

Nume program: EXEMPLUL 8 – PC (AMSTRAD, COMMODORE), m, M, F

```

OPTIUNE CIT TIMP OPTIUNE$=" "
 OPTIUNE$=" "
OPTIUNE SFIRȘIT
OPT 1 DACĂ OPTIUNE$="1"
 EXECUTĂ 2000
OPT 1 SFIRȘIT
OPT 2 DACĂ OPTIUNE$="2"
 Execută 3000
OPT 2 SFIRȘIT
OPT 3 DACĂ OPTIUNE$="3"
 Execută 4000
OPT 3 SFIRȘIT
OPT 4 DACĂ OPTIUNE$="4"
 Execută 5000
OPT 4 SFIRȘIT
OPT 5 DACĂ OPTIUNE$="5"
 Execută 6000
OPT 5 SFIRȘIT
OPT 6 DACĂ OPTIUNE$≠"b"
 Șterge ecranul
OPT 6 SFIRȘIT
OPT 7 DACĂ OPTIUNE$≠"6"
 OPTIUNE$=""
OPT 7 SFIRȘIT
1000 END
2000 SEQ
 Șterge ecranul
 I=1
X DACĂ FISINCARC=1
 ERASE COD, CANT, PREȚ
X SFIRȘIT
 Deschide LIVRĂRI . DAT
EOF CIT TIMP NOT EOF (1)
 Citește COD, CANT, PREȚ
 I=I+1
EOF SFIRȘIT
 Închide LIVRĂRI . DAT
 PRINT "Fișier încărcat"
 INPUT "(tastați CR)"; CR
 FISINCARC=1
2000 END
3000 SEQ
 Șterge ecranul
 Deschide LIVRĂRI . S
 C=1
SALV CIT TIMP C<I
 Scrie COD(C), CANT(C), PREȚ(C)
 C=C+1
SALV SFIRȘIT
 Închide LIVRĂRI . S
 PRINT "Salvat fișier"
 INPUT "(tastați CR)"; CR
3000 END
4000 SEQ
 Șterge ecranul

```

Fig. 8.2. b

## PSEUDOCODUL

Nume program: EXEMPLUL 8 – PC (AMSTRAD, COMMODORE), m, M, F

```

Citește CBEN
C=1
Deschide LIVRĂRI . DAT
GĂSIT=0
CAUT CIT TIMP C<1 și GĂSIT =0
 Citește COD(C), CANT(C), PREȚ(C)
X1 DACĂ CBEN=COD(C)
 GĂSIT=1
X1 SFIRȘIT
 C=C+1
CAUT SFIRȘIT
 Închide LIVRĂRI . DAT
X2 DACĂ GĂSIT+1=1
 PRINT "Beneficiarul"; CBEN; "Nu este în fișier"
X2 IN CAZ CONTRAR
 PRINT "Beneficiarul cu codul"; CBEN; "se află în fișier"
X2 SFIRȘIT
 INPUT "(tastați CR)"; CR
4000 END
5000 SEQ
 Șterge ecranul
 Citește CODB
 C=1
 GĂSIT=0
ACT CIT TIMP C<1 și GĂSIT=0
X3 DACĂ CODB=COD(C)
X31 SEQ
 GĂSIT=1
 Citește CANT(C)
X31 END
X3 SFIRȘIT
 C=C+1
ACT SFIRȘIT
GĂSIT DACĂ GĂSIT=0
 PRINT "Beneficiarul"; CODB; "nu este în fișier"
GĂSIT SFIRȘIT
 PRINT
 INPUT "(tastează CR)"; CR
5000 END
6000 SEQ
 Șterge ecranul
 I=0
 Deschide LIVRĂRI . DAT
LIST CIT TIMP NOT EOF (1)
 I=I+1
 Afișează COD(I), CANT(I), PREȚ(I)
 Citește COD(I), CANT(I), PREȚ(I)
LIST SFIRȘIT
 Închide LIVRĂRI . DAT
 INPUT "(tastează CR)"; CR
6000 END

```

Fig. 8.2. b

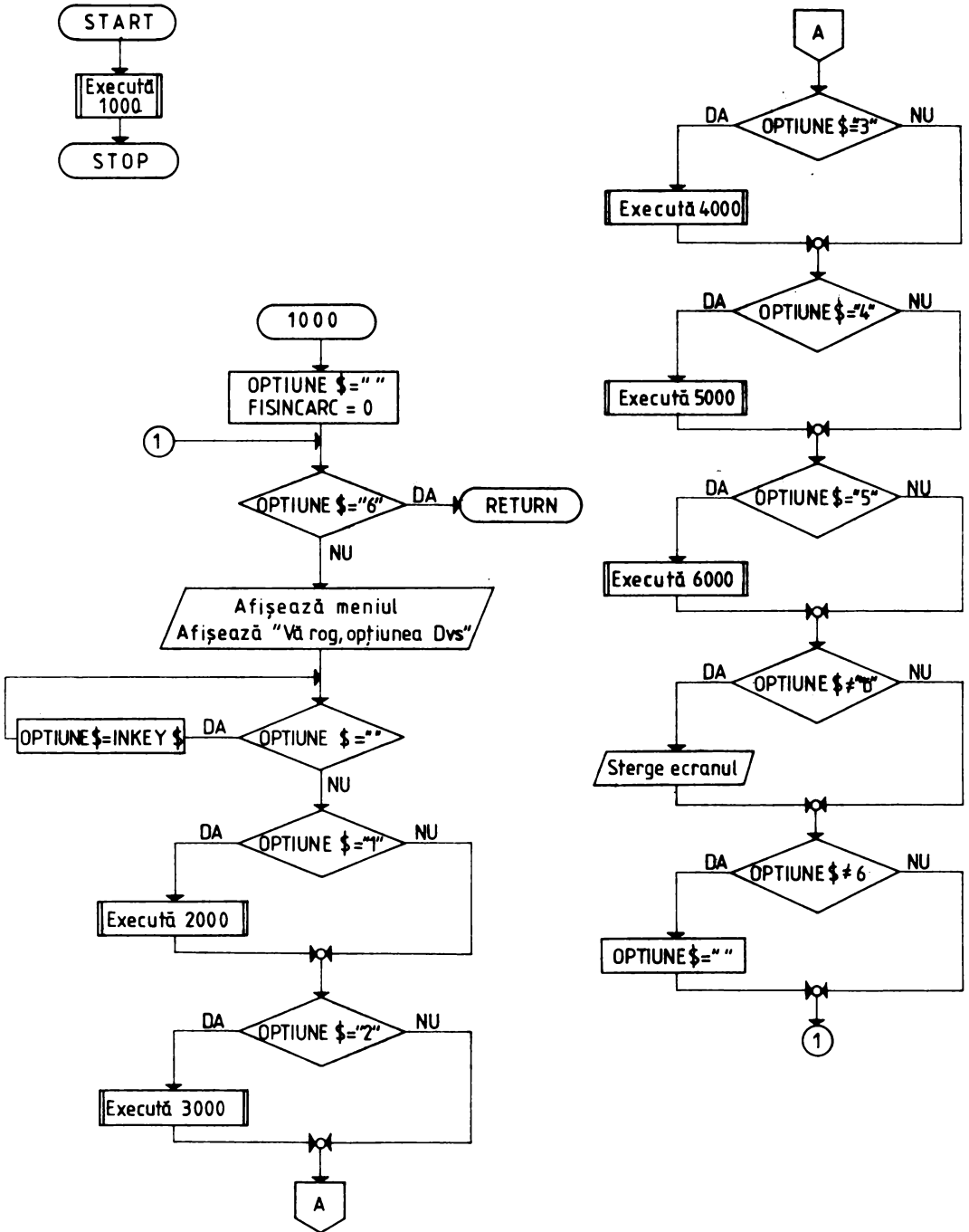


Fig. 8.2. c) schema logică.

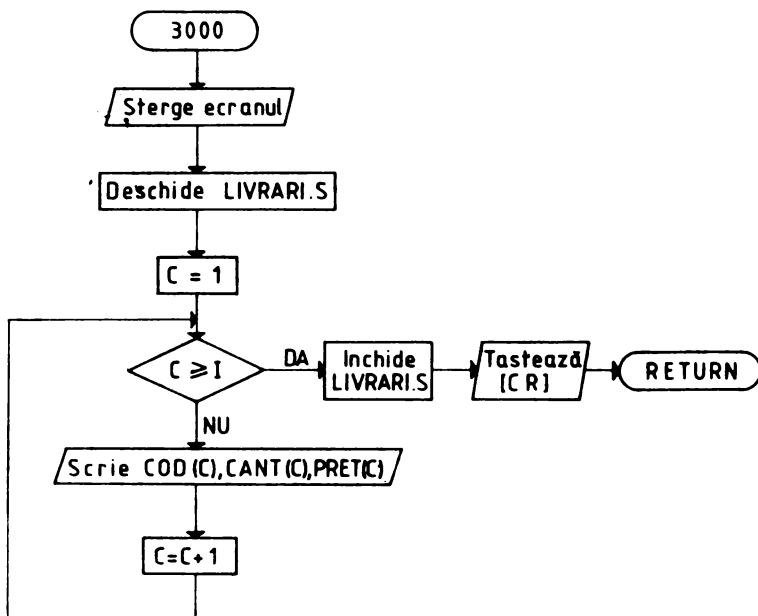
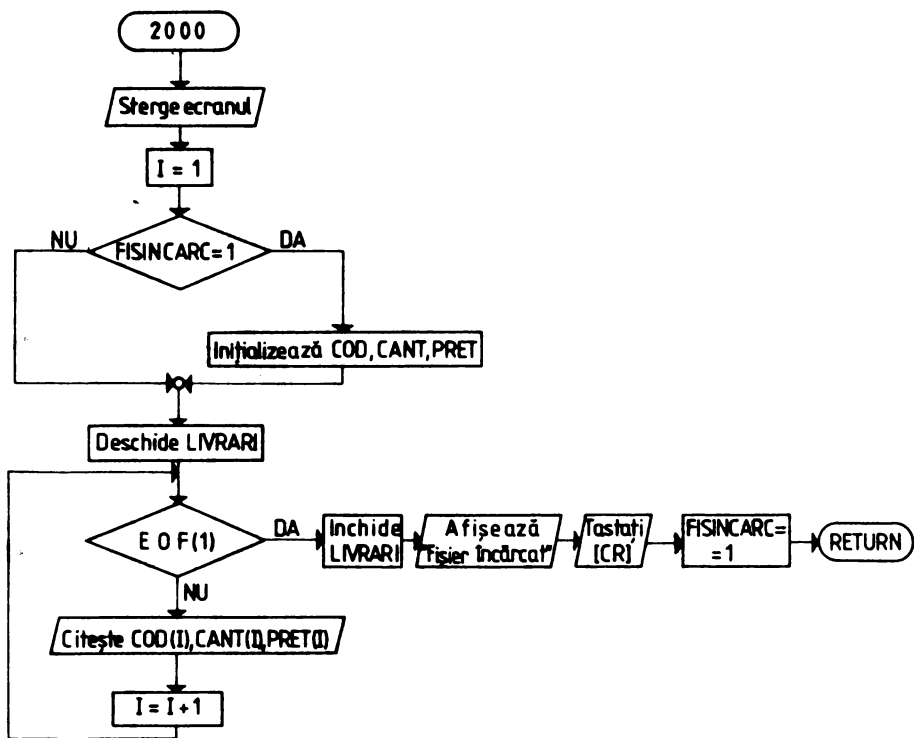


Fig. 8.2. c

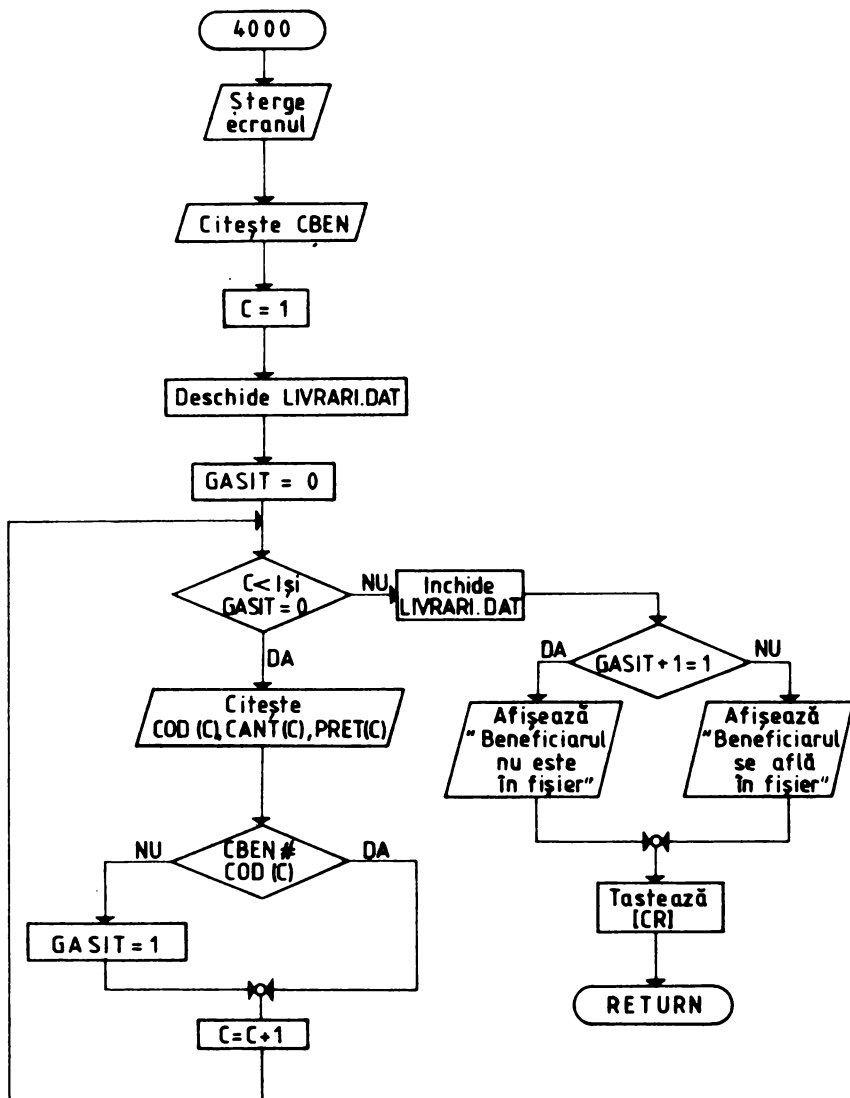


Fig. 8.2. c

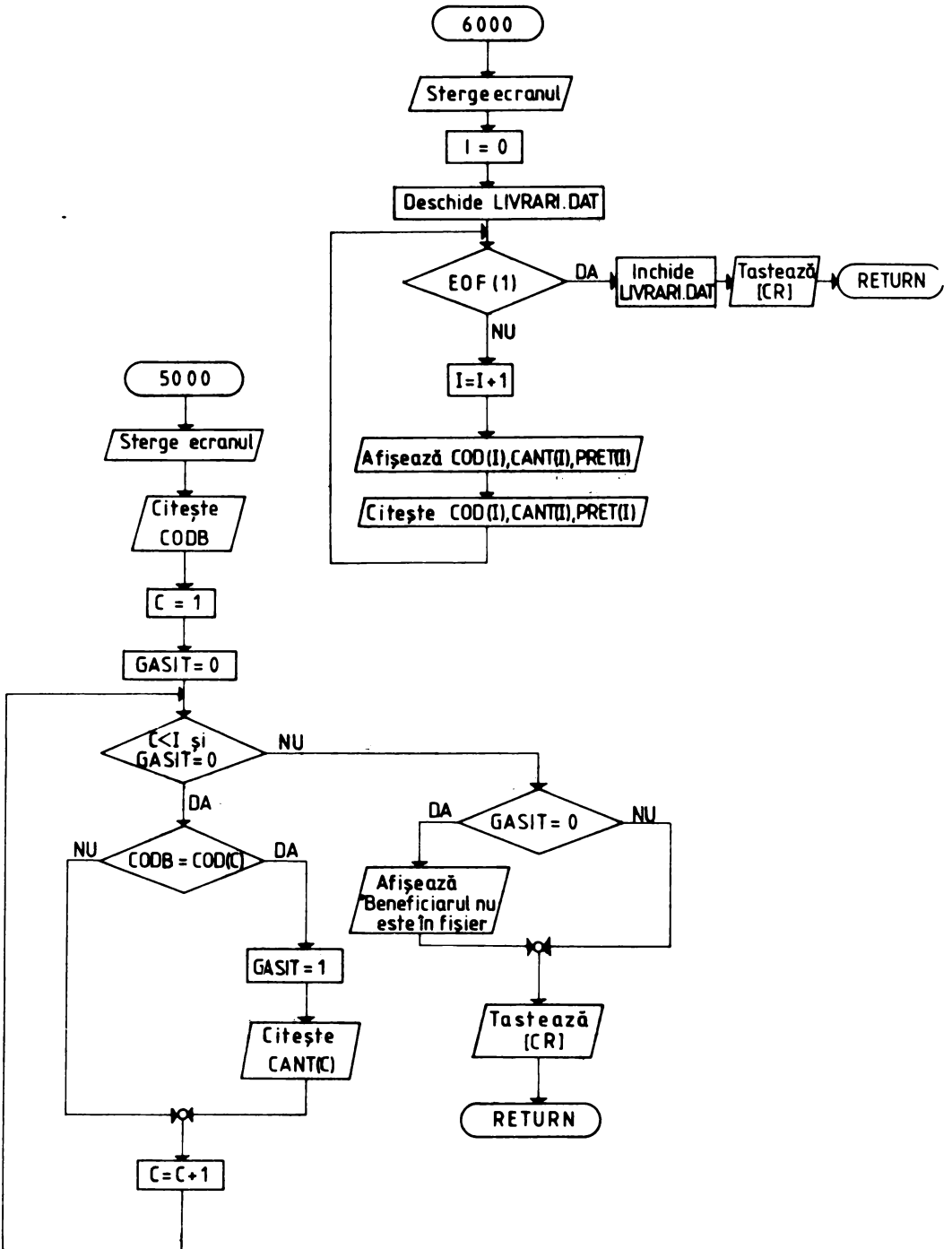


Fig. 8.2. c

## □ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 229

### Meniul conversației

Programul principal al conversației cuprinde doar patru linii dacă adăugăm și linia de comentariu (10).

```
10 REM " ** Intreținere fișier LIVRĂRI . DAT ** "
15 DIM COD (100), CANT (100), PREȚ (100)
20 GOSUB 1000
30 END
```

În linia 15 s-au declarat vectorii de date COD, CANT și PREȚ cu cel mult 100 de elemente.

Prin apelarea subrutinei 1000 programul afișează pe ecran meniul conversației, structurat, după cum puteți observa în șase funcțiuni.

```
1000 OPTIUNES=""
1010 FISINCARC=0
1020 WHILE OPTIUNES <> "6"
1030 PRINT
1040 PRINT "Alegeți 1-6"
1050 PRINT
1060 PRINT " 1. Incarcă fișier LIVRĂRI . DAT"
1070 PRINT " 2. Salvează fișier"
1080 PRINT " 3. Căutare în fișier"
1090 PRINT " 4. Actualizare fișier"
1100 PRINT " 5. Listare fișier"
1110 PRINT " 6. End"
1120 PRINT "Vă rog opțiunea Dvs."
1130 WHILE OPTIUNES=""
1140 OPTIUNES=INKEY$
1150 WEND
1160 IF OPTIUNES="1" THEN GOSUB 2000
1170 IF OPTIUNES="2" THEN GOSUB 3000
1180 IF OPTIUNES="3" THEN GOSUB 4000
1190 IF OPTIUNES="4" THEN GOSUB 5000
1200 IF OPTIUNES="5" THEN GOSUB 6000
1210 IF OPTIUNES <> " " THEN CLS
1220 IF OPTIUNES <> "6" THEN OPTIUNES=""
1230 WEND
1240 RETURN
```

Instrucțiunea **PRINT** din linia 1120

```
1120 PRINT "Vă rog opțiunea Dvs."
```

constituie o primă invitație la conversație.

Cît timp valoarea variabilei șir OPTIUNES nu se schimbă (în linia 1000 ea a fost inițializată cu șirul vid "") programul ciclează pînă cînd veți acționa o tastă.

### Instrucțiunea INKEY\$

Ce trebuie făcut în acest caz, ne veți întreba? Să tastați pur și simplu un număr cuprins între 1 și 6 (v. meniul programului) fără a mai acționa însă tasta [CR] după cum sînteți obișnuiți.

De ce să nu mai apăsăm pe tasta [CR] ne veți întreba, din nou? Pentru a vă răspunde la această întrebare, permiteți-ne să vă facem mai întâi cunoștință cu instrucțiunea **INKEY\$** pe care sigur o veți îndrăgi mai ales în cadrul jocurilor pe calculator.

```
1130 WHILE OPȚIUNES=""
1140 OPȚIUNES=INKEY$
1150 WEND
```

Instrucțiunea **INKEY\$** interoghează tastatura așteptând să introduceți în program o valoare șir. Dacă nu se acționează nici o tastă a claviaturii, **INKEY\$** returnează un șir vid.

Acum înțelegeți de ce programul ciclează (v. bucla 1130-1150) atita timp cit de la tastatură nu acționați nici o tastă.

Formatul general al instrucțiunii este

|                |
|----------------|
| Format general |
| <b>INKEY\$</b> |

*Observație.* În BASIC-AMSTRAD puteți utiliza și **INKEY** pentru numere întregi (v. vol. 2, pag. 31).

Așadar, să ne hotărîm și să acționăm una din taste. Pentru început este obligatoriu să tastați 1.

**Remarcă.** Ori de cîte ori utilizați **INKEY\$** sau **INKEY** nu mai este nevoie ca după acționarea tastei respective să apăsați pe [CR].

### Opțiunea 1.

#### Incarcă fișier LIVRĂRI . DAT

Prin tastarea cifrei 1 se apelează subrutina 2000 (v. linia 1160) care încarcă în memorie, de pe dischetă fișierul LIVRĂRI . DAT.

```
2000 REM * 1. Incarcă fișier LIVRĂRI . DAT
2005 CLS
2010 I=1
2020 IF FISINCARC=1 THEN ERASE COD, CANT, PREȚ
2040 OPENIN "LIVRĂRI . DAT"
2050 WHILE NOT EOF
2060 INPUT #9, COD(I), CANT(I), PREȚ(I)
2070 I=I+1
2080 WEND
2090 CLOSEIN
2100 PRINT
2110 PRINT "==" Fișier încărcat == "
2120 INPUT "tastați [CR]"; CR
2130 FISINCARC=1
2140 RETURN
```

Pentru încărcarea prin program a fișierului trebuie să ne asigurăm dacă fișierul a mai fost sau nu încărcat. După aceea vom deschide fișierul pe un canal de transfer, vom citi cite o înregistrare iar în final vom închide canalul de transfer asociat fișierului.



## Instrucțiunea ERASE

```
2020 IF FISINCARC=1 THEN ERASE COD, CANT, PREȚ
```

În cazul în care fișierul a mai fost încărcat (FISINCARC=1), cu instrucțiunea **ERASE** se elimină variabilele tablou din program (COD, CANT, PREȚ). De notat că, zona de memorie eliberată poate fi folosită în alte scopuri (de exemplu redimensionarea variabilelor tablou). Formatul general al instrucțiunii este:

|                                       |
|---------------------------------------|
| Format general                        |
| <b>ERASE</b> <listă variabile tablou> |

## Instrucțiunea OPENIN

```
2040 OPENIN "LIVRARI.DAT"
```

Instrucțiunea **OPENIN** (v. linia 2040) deschide fișierul LIVRARI.DAT creat în conversația precedentă (7) cu **OPENOUT**, înainte de a fi citit în memorie.

Formatul general al instrucțiunii este:

|                               |
|-------------------------------|
| Format general                |
| <b>OPENIN</b> "<nume-fișier>" |

unde, <nume-fișier> reprezintă numele fișierului de tip ASCII folosit în instrucțiunea **OPENOUT**.

## Funcția EOF

```
2050 WHILE NOT EOF
2060 INPUT #9, COD(I), CANT(I), PREȚ(I)
2070 I=I+1
2080 WEND
```

Pentru testarea stării fișierului deschis (v. linia 2050) se utilizează funcția **EOF** (End of File, sfârșit de fișier, în lb. engleză). Cât timp nu este sfârșit de fișier (**WHILE NOT EOF**) se execută instrucțiunile din liniile 2060 și 2070 (se citește de pe dischetă, în memorie, o înregistrare din fișier și se incrementază variabila I).

**EOF** returnează valoarea -1 atunci când se detectează sfârșitul de fișier sau fișierul nu este deschis și 0 în caz contrar.

*Observație.* Instrucțiunea **INPUT** din linia 2060 folosește canalul de deschidere nr. 9 (discheta).

## Instrucțiunea CLOSEIN

```
2090 CLOSEIN
```

Pentru închiderea fișierului de livrări deschis cu **OPENIN** limbajul BASIC AMSTRAD utilizează instrucțiunea **CLOSEIN**.

Formatul general al instrucțiunii este

|                |
|----------------|
| Format general |
| <b>CLOSEIN</b> |

*Observație.* Pentru reîntoarcerea în meniu acționați tasta [CR].

## Opțiunea 2. Salvează fișier

Prin tastarea cifrei 2 se apelează subrutina 3000 (v. linia 1170) care salvează pe dischetă fișierul de date existent în memorie.

Altfel spus, subrutina creează în **OUTPUT**, pe dischetă un fișier secvențial ASCII.

```

3000 REM * 2. Salvează fișier
3005 CLS
2010 OPENOUT "LIVRARI.S"
3020 C=1
3030 WHILE C < 1
3040 WRITE #9, COD(C), CANT(C), PREȚ(C)
3050 C=C+1
3060 WEND
3070 CLOSEOUT
3080 PRINT "Salvat fișier=="
3090 INPUT "tastați [CR]"; CR
3100 RETURN

```

*Observație.* Pentru reîntoarcerea în meniu tastați [CR].

## Opțiunea 3. Căutare în fișier

Prin tastarea cifrei 3 se apelează subrutina 4000 (v. linia 1180) care verifică dacă un beneficiar se găsește sau nu în fișierul de livrări.

```

4000 REM * 3. Căutare în fișier
4010 CLS
4020 INPUT "Precizați cod beneficiar"; CBEN
4030 C=1
4035 OPENIN "LIVRARI.DAT"
4040 GĂSIT=0
4050 WHILE C<1 AND GĂSIT=0
4052 INPUT #9, COD(C), CANT(C), PREȚ(C)
4054 IF CBEN < > COD(C) THEN 4060
4056 GĂSIT=1
4060 C=C+1
4070 WEND
4070 WEND
4072 CLOSEIN
4075 ON GĂSIT+1 GOTO 4080, 4092
4078 REM Beneficiarul a fost sau nu găsit în fișier.
4080 PRINT "Beneficiarul"; CBEN; "nu este în fișier"

```

```

4090 PRINT
4091 GOTO 4100
4092 PRINT "Beneficiarul cu codul"; CBEN; "se află în fișier"
4100 INPUT "(tastați [CR])"; CR
4110 RETURN

```

Regăsirea beneficiarului se face în funcție de codul său (v. linia 4020). Cât timp  $C < 1$  și  $G\dot{A}SIT=0$  (condiție compusă!) se execută instrucțiunile din liniile 4052–4060. În urma identificării beneficiarului variabila  $G\dot{A}SIT$  se poziționează pe 1.

La ieșirea din buclă se închide fișierul și se imprimă, funcție de valoarea variabilei de stare  $G\dot{A}SIT$ , mesajul privind rezultatul cercetării secvențiale a fișierului de livrări: "Beneficiarul a fost sau nu găsit în fișier" (v. liniile 4080 și 4092).

### Instrucțiunea ON ... GOTO

```
4075 ON G\dot{A}SIT+1 GOTO 4080, 4092
```

Cu instrucțiunea **ON ... GOTO** (v. linia 4075) se realizează saltul la una din liniile de după **GOTO**: 4080/4092.

Linia 4080 se selectează atunci când  $G\dot{A}SIT+1$  are valoarea 1. Pentru valoarea 2 a aceleiași variabile se selectează linia 4092.

Formatul general al instrucțiunii este

| Format general                                     |
|----------------------------------------------------|
| <b>ON</b> <expresie> <b>GOTO</b> <listă-nr. linii> |

în care: <expresie> este un număr întreg cuprins între 0 și 255; <lista – nr. linii> reprezintă numerele de linie ale programului, selectate în funcție de valoarea <expresie>.

#### Observații

- Dacă <expresie> este egală cu zero sau mai mare decât numărul de elemente din listă, selecția nu va mai avea loc;
- Dacă <expresie> este negativă apare un mesaj de eroare.

### Opțiunea 4.

#### Actualizare fișier

Prin tastarea cifrei 4 se apelează subrutina 5000 (v. linia 1190) care actualizează fișierul de livrări (se modifică valoarea cîmpului CANT).

```

5000 REM * 4. Actualizare fișier
5005 CLS
5010 INPUT "Pentru ce cod doriți actualizarea"; CODB
5020 C=1
5030 G\dot{A}SIT=0
5040 WHILE C<1 AND G\dot{A}SIT=0
5050 IF CODB <> COD (C) THEN 5060
5051 INPUT "Care este noua valoare a cantității livrate"; CANT(C)
5052 G\dot{A}SIT=1
5060 C=C+1
5070 WEND

```

```

5080 IF GĂSIT=0 THEN PRINT "Beneficiarul"; CODB; "nu este în fișier"
5090 PRINT
5100 INPUT "(tastați[CR])"; CR
5110 RETURN

```

Actualizarea cîmpului CANT are loc atunci cînd CODB (v. linia 5010) este egal cu COD(C) (v. linia 5050). Prin program (v. linia 5051) se citește dinamic CANT(C), adică noua valoare a cantității livrate. Din acel moment variabila de stare GĂSIT se poziționează pe 1. În situația cînd beneficiarul nu se află în fișier, instrucțiunea din linia 5080 afișează un mesaj explicit.

*Observație.* Pentru reîntoarcerea în meniu acționați tasta [CR].

### Opțiunea 5.

#### Listare fișier

Prin tastarea cifrei 5 se apelează subrutina 6000 (v. linia 1200) care listează conținutul fișierului de livrări.

```

6000 REM * 5. Listare fișier
6010 CLS
6015 PRINT "Cod Cantitate Preț"
6020 I=0
6030 OPENIN "LIVRĂRI.DAT"
6050 WHILE NOT EOF
6055 I=I+1
6060 PRINT COD(I); " "; CANT(I); " "; PREȚ(I)
6070 INPUT #9, COD(I), CANT(I), PREȚ(I)
6080 WEND
6090 CLOSEIN
6100 INPUT "(tastați [CR])"; CR
6110 RETURN

```

*Observație.* Pentru reîntoarcerea în meniu acționați tasta [CR].

### Opțiunea 6. End

Prin acționarea tastei cu cifra 6 se revine la linia 30 din programul principal. În acel moment conversația s-a încheiat.

**Aplicație.** Să se proiecteze și realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea și listarea unei fișier privind cheltuielile de întreținere, nominalizate ale locatarilor unui bloc. Se urmărește ca la sfîrșitul fiecărei luni să se genereze un raport identic cu cel pe care-l lecturați cu o mare atenție în holul blocului dvs. Pentru stabilirea informațiilor minimale necesare constituirii fondului de date vă recomandăm să-l abordați direct pe ... administrator.

## Codificarea în limbajul BASIC-COMMODORE

Diferențele de scriere a programului BASIC-COMMODORE apar la instrucțiunile **WHILE-WEND** (BASIC-COMMODORE nu are implementată structura **CIT TIMP**), **OPEN**, **CLOSE**, **INPUT #**, **WRITE #** și **INKEY** (v. instrucțiunea **GET**).

Cum dvs. aveți deja experiență în portabilitatea programelor, considerăm că nu vă va fi greu să scrieți singuri programul pentru calculatorul COMMODEORE operând în programul precedent următoarele modificări:

- Buclele **WHILE-WEND** din liniile 1020, 1230, 1130, 1150; 2050, 2080; 3030, 3060; 4050, 4070; 5040, 5070; 6050, 6090 se vor înlocui cu instrucțiunile **IF** și **GOTO** care simulează structura de iterație **CIT TIMP** (v. conversația 5);
- Instrucțiunea de ștergere a ecranului din liniile: 1210, 2005, 3005, 4010, 5005, 6010 se va înlocui cu

```
2005 PRINT CHR$(147);
```

- Instrucțiunile **OPEN** de deschidere a fișierului de livrări în input din liniile 2040, 4035, 6030 se vor înlocui cu instrucțiunea

```
2040 OPEN, 2, 8, 2, "LIVRARI.DAT S, R"
```

- Instrucțiunea **OPEN** de deschidere a fișierului de livrări în output (pentru salvare) din linia 3010 se va înlocui cu instrucțiunea

```
OPEN 2, 8, 2, "LIVRARI.S, W"
```

- Instrucțiunea **CLOSE** de închidere a fișierului de livrări din liniile 2090, 3070, 4072, 6095 se va înlocui cu instrucțiunea

```
2090 CLOSE 2
```

```
3070 CLOSE 2
```

```
4072 CLOSE 2
```

```
6095 CLOSE 2
```

- Instrucțiunea **INKEY\$** din linia 1140 se va înlocui cu instrucțiunea **GET**:

```
1135 GET A$
```

```
1140 OPTIUNES=A$
```

Formatul general al instrucțiunii este

| Format general              |                                             |
|-----------------------------|---------------------------------------------|
| <b>GET</b> [#(nr. fișier),] | { <variabilă numerică><br><variabilă șir> } |

unde, (nr. fișier) reprezintă numele fișierului; el este un întreg cuprins între 1 și 255;

<variabilă numerică> și

<variabilă șir> reprezintă valoarea (numerică/șir) transmisă de la tastatură.

#### Remarci

- Dacă <variabilă numerică> primește un caracter nenerumeric se afișează eroare;
- Dacă <variabila șir> nu primește nici o valoare se consideră implicit, drept valoare șirul vid;
- Dacă <variabila numerică> nu primește nici o valoare se consideră implicit drept valoare zero.

- Instrucțiunea **INPUT #** din liniile 2060, 4052, 6070 se va înlocui cu instrucțiunea:

```
2060 INPUT #2, COD(I), CANT(I), PREȚ(I)
4052 INPUT #2, COD(C), CANT(C), PREȚ(C)
6070 INPUT #2, COD(I), CANT(I), PREȚ(I)
```

- Instrucțiunea **WRITE #** din linia 3040 se va înlocui cu instrucțiunea:

```
3040 PRINT #2, COD(C), CANT(C), PREȚ(C)
```

- Depistarea sfârșitului unui fișier secvențial se realizează prin instrucțiunile:

```
RS=ST
IF RS > 0 THEN CLOSE
```

unde, **ST** reprezintă starea fișierului (cuvânt cheie BASIC-COMMODORE).

**Aplicație.** Să se proiecteze și realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea și listarea unui fișier (secvențial) privind nomenclatorul străzilor municipiului București.

Informațiile minimale necesare constituirii fondului de date sint:

- cod stradă
- denumirea străzii
- sector
- oficiu poștal

## Joc pe HC-85, TIM S, SPECTRUM

```
1 REM CHEIA
5 LET nt=0
10 DIM t(4,20)
20 LET a$="10100101011101010110"
30 LET flag=0
40 RANDOMIZE
50 FOR j=1 TO 3
60 LET r=INT(RND*19+1)
70 FOR k=1 TO 20
80 LET dis=k+r
90 IF dis<21 THEN GO TO 110
100 LET dis=dis-20
110 LET t(j, k)=VAL a$(dis)
120 NEXT k
130 NEXT j
140 IF flag>0 THEN GO TO 180
150 FOR k=1 TO 20
160 LET t(4,k)=t(1,k)+t(2,k)+t(3,k)
170 NEXT k
180 LET flag=flag+1
190 IF flag>1 THEN GO TO 210
200 IF flag=1 THEN GO TO 50
210 CLS
220 LET tel=0
```

```
230 FOR k=1 TO 20
240 PRINT AT 10,k+2;t(1,k)
250 PRINT AT 11,k+2;t(2,k)
260 PRINT AT 12,k+2;t(3,k)
270 LET v=t(4,k)-t(3,k)-t(2,k)-t(1,k)
280 LET v=ABS(v)
290 IF v=0 THEN LET tel=tel+1
310 PRINT AT 14,k+2;v
320 NEXT k
330 IF tel=20 THEN GO TO 450
340 INPUT "LINIA=";j
350 INPUT "MUTAREA=";s
360 LET nt=nt+1
370 FOR i=1 TO s
380 LET h=t(j,1)
390 FOR k=1 TO 19
400 LET t(j,k)=t(j,k+1)
410 NEXT k
420 LET t(j,20)=h
430 NEXT i
440 GO TO 210
450 PRIN FLASH 1; "AȚI REUȘIT IN";
nt; "ÎNCERCĂRI"
```

□ **Particularități ale programării  
în limbajul BASIC-80**

vol. 2, pag. 231

Cu mici diferențe, programul conversației este identic cu cel executat pe calculatorul AMSTRAD. În continuare vă invităm pe dvs. să le identificați ca apoi să le discutăm împreună.

După cum ați putut constata, o primă diferență apare la instrucțiunea **OPEN** (v. liniile 2040, 3010, 4035, 6030). În limbajul BASIC-80, prin parametrul **(MOD)** (v. formatul general al instrucțiunii **OPEN**, conversația 7) se specifică modul de intrare/ieșire. Instrucțiunile 2040, 4035 și 6030

2040 **OPEN** "I", #2, "LIVRARI.DAT"

deschid fișierul de livrări în input ("I"), iar instrucțiunea

3010 **OPEN** "O", #2, "LIVRARI.S"

deschide fișierul LIVRARI.S în output.

Instrucțiunea de ștergere a ecranului (v. liniile 1210, 2005, 3005, 4010, 5005, 6010) se va înlocui cu **CHR\$(24)**.

În BASIC-80 închiderea fișierelor se realizează cu **CLOSE** urmat de numărul fișierului.

2090 **CLOSE** #1

3070 **CLOSE** #2

4072 **CLOSE** #1

6095 **CLOSE** #1

Sperăm că ați remarcat cum instrucțiunile **INPUT** din liniile 2060, 4052, 6080 s-au înlocuit cu:

2060 **INPUT** #1, COD(I), CANT(I), PREȚ(I)

4052 **INPUT** #1, COD(C), CANT(C), PREȚ(C)

6080 **INPUT** #1, GOD(I), CANT(I), PREȚ(I)

În linia 3040 ați găsit vreo deosebire față de programul implementat pe AMSTRAD?

3040 **PRINT** #2, COD(C), CANT(C), PREȚ(C)

În BASIC-80 pentru scrierea datelor într-un fișier secvențial poate fi utilizată și instrucțiunea **WRITE #**.

Formatul general al instrucțiunii este:

| Format general                                    |
|---------------------------------------------------|
| <b>WRITE</b> [#] <număr-fișier>, <listă-expresii> |

unde, <număr-fișier> este numărul sub care fișierul a fost deschis în output ("O"); <listă-expresii> reprezintă șiruri sau expresii numerice, separate prin virgulă.

*Observație.* Diferența dintre **WRITE #** și **PRINT #** constă în aceea că **WRITE #** inserează o virgulă după fiecare element introdus în fișier și **[CR]** după ultimul element din listă.

O ultimă diferență apare la testarea sfârșitului de fișier (secvențial). În BASIC-80 se utilizează tot funcția **EOF**, dar cu formatul

|                                                                                                           |
|-----------------------------------------------------------------------------------------------------------|
| <hr style="border: 1px solid black;"/> Format general <hr style="border: 1px solid black;"/>              |
| <hr style="border: 1px solid black;"/> <b>EOF ((număr-fișier))</b> <hr style="border: 1px solid black;"/> |

unde, <număr-fișier> reprezintă numărul fișierului precizat în instrucțiunile **OPEN, CLOSE, INPUT**.

*Observație.* **EOF** returnează valoarea -1 atunci când s-a atins sfârșitul fișierului.

## TEST

Încercați să treceți de la programul scris în limbajul BASIC-80 la programul scris în limbajul BASIC-AMSTRAD.

**Aplicație.** Să se proiecteze și realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea și listarea unui fișier (secvențial) privind cheltuielile unor secții de prelucrare (mecanice) dintr-o întreprindere constructoare de mașini.

Structura înregistrării fișierului de cheltuieli este:

- Cod secție
- Denumire secție
- Cheltuieli
  - ianuarie
  - februarie
  - martie
  - aprilie
  - 
  - 
  - decembrie

## □ Crearea și întreținerea unui fișier de livrări în acces direct în limbajul BASIC-80

Limbajul BASIC-80 dispune și de facilități privind lucrul cu fișiere în acces direct. BASIC-80 pune la dispoziția utilizatorului trei instrucțiuni speciale: **FIELD #**, **LSET** și **RSET**, precum și un set de funcții de conversie.

În cele ce urmează vă prezentăm două programe pentru crearea (formatarea), respectiv întreținerea unui fișier de livrări în acces direct.

### Creare fișier

Structura fișierului este: cod beneficiar, denumire beneficiar, cantitate livrată și preț.

```

10 REM "Formatare fișier LIV.DAT"
20 OPEN "R", #1, "LIV.DAT"
30 FIELD #1, 1 AS C1$, 20 AS C2$, 4 AS C3$, 7 AS C4$
40 LSET C1$=CHR$(255)
50 FOR I=1 TO 98
60 PUT #1, I
70 NEXT I
80 CLOSE #1
90 END

```



## Instrucțiunea FIELD #

După deschiderea fișierului în acces direct LIV . DAT (v. linia 20), instrucțiunea **FIELD #** alocă un spațiu, respectiv 1, 20, 4, 7 caractere variabilelor: C1\$ (cod beneficiar), C2\$ (denumire beneficiar), C3\$ (cantitate livrată), C4\$ (preț).

Instrucțiunea **FIELD #** are următorul format:

| Format general                                                                                                   |
|------------------------------------------------------------------------------------------------------------------|
| <b>FIELD [#]</b> (nr. fișier), (lungime cîmp) <b>AS</b> (var șir)<br>{,(lungime cîmp) <b>AS</b> (var. șir) ... } |

în care: (nr. fișier) este numărul fixat pentru fișier la deschidere;  
(lungime cîmp) reprezintă numărul de caractere ale cîmpului.

### Remarci

- Instrucțiunea **FIELD #** este o instrucțiune executabilă dar execuția ei nu produce nici un transfer de informații, ci numai declară cîmpurile buffer-ului ca avînd nume de variabilă pentru identificarea lor ulterioară.
- Instrucțiunea **FIELD #** nu se folosește pentru variabilele utilizate în instrucțiunile **LET** sau **INPUT**.
- Numărul total al octeților alocați instrucțiunii **FIELD #** nu trebuie să depășească lungimea înregistrării specificate la deschiderea fișierului în acces direct.

## Instrucțiunea LSET

Instrucțiunea **LSET** din linia 40 realizează transferul efectiv al datei (**CHR\$(255)**) în zona de alocare a variabilei.

Formatul instrucțiunii **LSET** este:

| Format general                      |
|-------------------------------------|
| <b>LSET</b> (var. șir) = (exp. șir) |

unde: (var. șir) reprezintă orice variabilă simplă sau indexată de tip șir;  
(exp. șir) reprezintă orice expresie de tip șir a cărei valoare va face obiectul transferului (valorile numerice trebuie convertite în șiruri înainte de a fi transferate cu **LSET** – aliniază șirul la stînga și **RSET** – aliniază șirul la dreapta).

## Instrucțiunea PUT

În sfîrșit, instrucțiunea **PUT** (v. linia 60) scrie 98 de înregistrări (v. ciclul **FOR-NEXT** din liniile 50–70) în fișierul LIV . DAT.

Formatul instrucțiunii este:

| Format general                              |
|---------------------------------------------|
| <b>PUT [#]</b> (nr. fișier) [, (nr. inreg)] |

unde: (nr. inreg.) reprezintă numărul înregistrării din fișier.

*Observație.* Dacă (nr. inreg) lipsește, atunci înregistrarea va primi numărul următor celui dat ultimei înregistrări în fișier. (nr. inreg) este cuprins între 1 și 32767.

## Întreținere fișier

Programul următor (cu greșeli!) actualizează fișierul LIV.DAT, prilej cu care vi se prezintă funcția **MKS\$** (convertește un număr în simplă precizie într-un șir de 4 octeți) din linia 515 și instrucțiunea **GET #** (v. linia 100) care servește la transferarea unui articol din fișierul deschis, pe canalul indicat, în memoria internă, articolul fiind considerat ca o entitate. Funcția **CVS** (v. linia 120) convertește șirurile de cifre în valori numerice.

```

10 REM "Actualizare fișier LIV.DAT"
20 OPEN "R", #1, "LIV.DAT", 32
30 FIELD # 1,1 AS C1$, 20 AS C2$, 4 AS
 C3$, 7 AS C4$
40 PRINT CHR$(24)
50 PRINT "****ACTUALIZARE FIȘIER LIV.
 DAT****"
60 INPUT "Cod beneficiar"; CB
70 IF CB=0 THEN CLOSE#1 : END
80 IF CB>=1 AND CB<=98 AND CB=
 INT (CB) THEN 100
90 PRINT "Cod eronat" : GOTO 60
100 GET#1, CB
110 IF C1$=CHR$(255) THEN PRINT
 "Cod inexistent" : GOTO 300
120 DEN$=C2$: CANT=CVS(C3$) :
 PREȚ=CVS(C4$)
130 PRINT "1. Denumire beneficiar";
 DEN$
140 PRINT "2. Cantitatea livrată"; CANT
150 PRINT "3. Preț" : PREȚ
160 PRINT INPUT "OK (Y/M/D)"; Y$
170 IF Y$="Y" THEN 40
180 IF Y$="M" THEN 200
190 IF Y$="D" THEN 400
195 GOTO 160
200 REM "Modificare"
210 INPUT "Număr linie="; I
220 IF I=0 THEN GOSUB 500 : GOTO 40
230 GOSUB 1000 : GOTO 210

300 REM "Creare"
310 INPUT "Creare un nou element
 (D/N) ";Y$
320 IF Y$="D" THEN 350
330 IF Y$="N" THEN 40
340 GOTO 310
350 FOR I=1 TO 3 : GOSUB 1000 : NEXT I
360 GOSUB 500 : GOTO 40
400 REM "Ștergere"
410 LSET C1$=CHR$(255) : PUT#1, CB
420 PRINT "Înregistrare ștersă" : INPUT
 "Tastează o cheie"; Y$
430 GOTO 40
500 REM "Scriere înregistrare"
510 LSET C1$=CHR$(0) : LSET C2$=
 DEN$
515 LSET C3$=MKS$(CANT) : LSET
 C4$=MKS$(PREȚ)
520 PUT#1, CB
530 RETURN
1000 REM "Intrări subrutine"
1010 ON I GOSUB 1100, 1200, 1300 :
 RETURN
1100 INPUT "1. Denumire beneficiar";
 DEN$: RETURN
1200 INPUT "2. Cantitatea livrată";
 CANT : RETURN
1300 INPUT "3. Preț";
 PREȚ : RETURN
2000 END

```

## Instrucțiunea GET #

Formatul general al instrucțiunii este:

|                                                        |
|--------------------------------------------------------|
| Format general                                         |
| <b>GET [#] &lt;nr. fișier&gt; [,&lt;nr. înreg&gt;]</b> |

unde: <nr. fișier> este numărul atribuit fișierului la deschidere;  
 <nr. înreg> reprezintă numărul înregistrării de citit. Dacă este omis,  
 se citește următoarea înregistrare din fișier (cel mai mare număr de  
 înregistrări posibile este 32767).

## □ Particularități ale programării în limbajul BASIC-PLUS

Pentru scrierea programului BASIC-PLUS se vor avea în vedere următoarele **particularități**:

- limbajul BASIC-PLUS nu are implementată structura **WHILE**. Buclele **WHILE-WEND** (v. liniile 1020, 1230, 1130, 1150, 2050, 2080, 3030, 3060, 4050, 5040, 5070, 6050, 609C) din EXEMPLUL 8-PC (AMSTRAD) se vor simula cu **IF** și **GO TO**;
- pentru deschiderea unui canal pentru un fișier deja existent se utilizează instrucțiunea **OPEN...FOR INPUT** (v. liniile 2040, 4035 și 6030 din programul BASIC-AMSTRAD). Dacă fișierul indicat prin (specificator fișier) nu există pe suport, execuția instrucțiunii va fi abandonată (se tipărește un mesaj de eroare);
- instrucțiunea **OPEN...FOR OUTPUT** presupune deschiderea unui canal pentru un nou fișier (v. linia 3010 din programul BASIC-AMSTRAD). Dacă fișierul indicat prin (specificator fișier) nu există acesta se creează pe suportul menționat; dacă există, se creează un nou fișier;

*Observație.* Deschiderea canalului pentru prelucrarea unui fișier este automat însoțită de alocarea în memoria internă a spațiului necesar bufferului de intrare/ieșire.

- instrucțiunea **CLOSE** provoacă închiderea unui canal de transfer deschis printr-o instrucțiune **OPEN** (v. liniile 2090, 3070, 4072, 6095 din programul BASIC-AMSTRAD);
- limbajul BASIC-PLUS nu admite instrucțiunea **INKEY\$** (v. linia 1140 din programul BASIC-AMSTRAD); se recomandă utilizarea instrucțiunii **INPUT**;
- pentru transferul efectiv al datelor dintr-un fișier deschis pe un anumit canal de transfer în memoria internă se utilizează instrucțiunea **INPUT #** (v. liniile 2060, 4052, 6070 din programul BASIC-AMSTRAD). Fișierele care se exploatează prin **INPUT#** trebuie să aibă inserat caracterul "," între elementele unui articol.

*Observație.* Dacă (expresie numerică) este zero (v. formatul general al instrucțiunii **INPUT#**, conversația 7), suportul care se ia în considerare este terminalul utilizator.

Pentru citirea unui singur articol din fișierul asociat canalului, se utilizează instrucțiunea **INPUT LINE #** al cărei format general este:

| Format general                                          |
|---------------------------------------------------------|
| <b>INPUT LINE#</b> (expresie numerică), (variabilă șir) |

unde, (expresie numerică) este o expresie numerică a cărei valoare (în-treagă) cuprinsă între 0 și 8 precizează numărul canalului de transfer;

(variabila șir) este o variabilă de tip numeric căreia i se atribuie ca valoare textul articolului.

### Reguli

- **INPUT LINE #** trebuie obligatoriu precedat de instrucțiunea **OPEN**;
- Caracterul "," își pierde semnificația de separator al datelor articolului;
- Instrucțiunea **PRINT #** asigură transferul de date între memoria internă și un suport orientat pe un articol (LP, TT etc.) sau un fișier (v. linia 3040 din programul BASIC-AMSTRAD);
- pentru prelucrarea fișierului ASCII secvențiale se mai pot utiliza instrucțiunile: **PRINT #...USING** (instrucțiune de scriere a unui articol prin editarea sub un anumit format); **MAT INPUT#** (instrucțiune de citire a unui masiv dintr-un fișier);

**MAT PRINT#** (instrucțiune de scriere a unui masiv pe un fișier); **KILL** (instrucțiune de eliminare a unui fișier ASCII secvențial);

- Marcarea sfârșitului fizic al unui fișier secvențial ASCII intră în sarcina utilizatorului. La exploatare se va testa sfârșitul de fișier indicat prin program. În caz contrar se va semnala un mesaj de eroare (v. liniile 2050, 6050 din programul BASIC-AMSTRAD).

**Aplicație.** Să se proiecteze și realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea, consultarea și listarea unui fișier (secvențial) de beneficiari ai unei întreprinderi constructoare de utilaj petrolier.

Informațiile minimale necesare constituirii fondului de date pentru beneficiarii întreprinderii sînt:

- Cod beneficiar
- Denumire beneficiar
- Adresă
- Cont bancă

## □ Particularități ale programării în limbajul ABASIC

Pentru scrierea programului se vor avea în vedere următoarele **particularități**:

- instrucțiunea **OPEN**, de deschidere a fișierului este urmată (opțional) de caracterul "#" apoi de numărul fișierului și de numele fișierului ce respectă convențiile de sintaxă ARIEL (v. liniile 2040, 3010, 4035, 6030 din programul BASIC-AMSTRAD);
- în limbajul ABASIC nu există instrucțiunea **INKEY\$**;
- pentru citirea datelor dintr-un fișier (secvențial) se utilizează instrucțiunile de citire: **INPUT #** și **LINPUT #**;
- pentru scrierea datelor într-un fișier (secvențial) se pot folosi instrucțiunile: **PRINT #**, **PRINT # . . . USING**, **WRITE #**. Instrucțiunile au același efect ca și cele de scriere directă la terminal;
- detectarea sfârșitului de fișier se realizează folosind funcția

**EOF** ((expresie))

care ia valoarea -1 atunci cînd s-a citit sfârșitul fișierului identificat prin (expresie) sau 0 dacă sfârșitul fișierului nu a fost detectat.

**Aplicație.** Să se proiecteze și realizeze un program BASIC, ghidat de un meniu pentru crearea, actualizarea și listarea unui fișier (secvențial) privind nomenclatorul produselor unei întreprinderi de confecții.

Informațiile minimale necesare constituirii fondului de date sînt:

- cod produs
- denumire produs
- caracteristici
- preț
- anul fabricației.

## TEMA 8

 Răspundeți prin DA sau NU la următoarele întrebări:

- Instrucțiunea **INKEY\$** interoghează tastatura.
- Dacă nu se acționează nici o tastă, **INKEY\$** returnează un șir vid.
- Instrucțiunea **ERASE** elimină variabilele tablou dintr-un program.
- În BASIC-AMSTRAD, instrucțiunea **INKEY** poate fi utilizată și pentru numere întregi.
- Instrucțiunea **OPENIN** este specifică limbajului BASIC-PLUS.
- Funcția **EOF** testează sfârșitul de fișier.
- Instrucțiunea **CLOSEIN** închide fișierele BASIC-80.
- Cu instrucțiunea **ON ... GO TO** se realizează saltul la una din liniile de după **GO TO**.
- Instrucțiunea **INKEY\$** este acceptată de limbajul BASIC-COMMODE.
- **EOF** returnează valoarea -1 atunci când s-a depistat sfârșitul unui fișier secvențial.

 Înlocuiți cuvintele care lipsesc din următoarele propoziții:

- a) Funcția **EOF** (<expresie>) poate fi utilizată în instrucțiunile..... ,  
.....
- b) Instrucțiunea **INKEY\$** este folosită pentru ..... în timp ce instrucțiunea **INPUT** se utilizează pentru .....
- c) Instrucțiunea **ERASE** este implementată în BASIC .....
- d) În BASIC-COMMODE, depistarea sfârșitului unui fișier secvențial se realizează prin .....
- e) În limbajul BASIC-80 prin parametrul <MOD> al unei instrucțiuni **OPEN** se specifică .....
- f) În limbajul BASIC-80 diferența dintre **WRITE#** și **PRINT#** constă în .....
- g) Instrucțiunea **OPEN ... FOR INPUT** din limbajul BASIC-PLUS presupune ..... iar instrucțiunea **OPEN ... FOR OUTPUT** presupune .....
- h) În limbajul ..... instrucțiunea **INPUT LINE#** trebuie obligatoriu precedată de .....
- i) Setul de instrucțiuni al limbajului BASIC-PLUS pentru prelucrarea fișierelor ASCII secvențiale este următorul: .....
- j) În secvența de instrucțiuni BASIC-PLUS:
- ```
10 OPEN "AA . A" FOR INPUT AS FILE 2 TO WRITE
20 OPEN "AA . B" FOR OUTPUT AS FILE 4 TO READ
30 PRINT# 3, M, N$
40 INPUT# 2, A, C%
```
- instrucțiunile sînt incorecte.

Scrieți cite un program BASIC pentru fiecare din problemele de mai jos:

a) Să se gestioneze pe sistemele de calcul FELIX C, mini, micro și calculatoare personale datele referitoare la personalul care locuiește în municipiul București. Informațiile minimale necesare constituirii și actualizării setului de date privind personalul care locuiește în municipiul București sînt:

- număr matricol
- nume
- prenume
- adresă
 - sector
 - stradă
 - număr
- data nașterii
 - zi
 - lună
 - an
- sex

Tranzacțiile care afectează conținutul seturilor de date prezentate anterior sînt definite după cum urmează:

- tranzacții de tip adăugări
- tranzacții de tip modificări
- tranzacții de tip ștergeri

FTRANZ

- tip tranzacție
- număr matricol
- nume
- prenume
- adresă
- sector
- stradă
- număr
- sex

tip tranzacție avînd una din valorile:

- 13 = introducerea unei noi înregistrări;
- 14 = ștergerea unei înregistrări;
- 15 = modificarea conținutului unei înregistrări.

Rapoartele de ieșire sînt următoarele:

1) LISTA L1: LISTA CU PERSONALUL CARE LOCUIEȘTE IN SECTORUL 1:

Conținut:

- număr curent
- număr matricol
- nume și prenume
- strada
- număr
- data nașterii
 - zi
 - luna
 - an
- sex

NR. LOCUITORILOR DIN SECTORUL 2: * * *

NR. LOCUITORILOR DIN SECTORUL 3: * * *

NR. LOCUITORILOR DIN SECTORUL 6: * * *

2) LISTA L2

Conținut:

- | | |
|-----------|-------------|
| – nume | – număr |
| – prenume | – sector |
| – stradă | – nume oraș |

unde, nume-oraș este BUCUREȘTI.

Formatul listei L2 urmează să fie precizat de către fiecare dintre dumneavoastră.

b) Să se calculeze integrala definită

$$\int_{1.23}^{18.25} (x^2 + x + 1) dx$$

prin metoda trapezelor, dreptunghiurilor și Simpson. Programul va fi ghidat de un meniu.

c) Să se determine prin metodele Newton-Raphson și bisecție soluția ecuației

$$x^3 - 2x - 9 = 0$$

cunoscând că o rădăcină a acestei ecuații este cuprinsă în intervalul (2,3÷2,5). Precizia este 10^{-5} . Programul va fi ghidat de un meniu.

d) Să se gestioneze pe mini, micro și calculatoare personale datele referitoare la transformatoarele din posturile de transformare ce aparțin unui centru județean de distribuție a energiei electrice. Informațiile minime necesare constituirii și actualizării seturilor de date sînt: cod post transformare; caracteristici transformator; anul punerii în funcțiune a transformatorului; seria; puterea; grupa de conexiuni; tensiunea de scurtcircuit; bateriile de condensatoare pentru compensarea factorului de putere; circuitele de iluminat public; circuitele de forță; adresa.

Programul BASIC va fi ghidat de un meniu care să permită realizarea următoarelor funcțiuni: 1. help; 2. creare fișier; 3. adăugare înregistrări; 4. modificare cimpuri; 5. ștergere înregistrări; 6. modificare structură fișier; 7. listare fișier; 8. end.

Observație. Programul este operațional la I.R.E Ploiești.

e) Există în practica multor I.R.E.-uri din țară de a raporta situațiile referitoare la consumul de energie la nivelul județului. Raportarea înglobează situația consumurilor de energie pe ministere, situația energiei suplimentare generate de autoproducători cît și situația consumului de energie al principalilor consumatori la nivelul județului.

Să se gestioneze pe mini, micro și calculatoare personale datele referitoare la consumul de energie electrică la nivel de județ. Informațiile minime necesare constituirii și actualizării seturilor de date sînt: cod mi-

nister; denumire minister, energie totală contur județ; puterea totală pe palierul 2 (3, 4) pe județ; numărul maxim de ministere pe județ; numărul maxim de întreprinderi ale ministerului; energia totală pe paliere pe întreprindere; energia pe palierul 2 (3, 4) pe întreprindere. Programul BASIC va fi ghidat de un meniu și va genera raportul zilnic ce conține: energia totală contur județ; puterile totale pe palierul 1 (2, 3, 4) pe județ; energia totală pe județ; puterile totale pe palierul 1 (2, 3, 4) pe județ.

Observație. Programul este operațional la I.R.E. Ploiești.

Se reia problema din conversația precedentă, complicînd-o după cum urmează.

Să se proiecteze și să se realizeze un program BASIC ghidat de un meniu care să permită realizarea următoarelor funcțiuni: **I.** Încarcă fișier JUCĂRII.DAT; **S.** Salvează fișier; **C.** Căutare în fișier; **L.** Listare fișier; **F.** Stop.

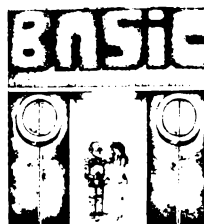
Se reia problema din conversația precedentă, complicînd-o astfel. Să se proiecteze și să se realizeze un program BASIC ghidat de un meniu cu următoarea structură:

1. Încărcare fișier CONSUMURI.DAT; **2.** Actualizare fișier; **3.** Listare fișier; **4.** Stop.

Indicație. Se vor utiliza subrutinele din EXEMPLELE 8 – PC, m, M, F.

CONVERSAȚIA 9

Proiectarea și realizarea unui program BASIC-PLUS ghidat de un meniu pentru crearea, actualizarea și listarea unui fișier masiv virtual de livrări. Instrucțiunile DIM, OPEN... AS VIRTUAL și CLOSE. JOCURI, APLICAȚII și TESTE pentru cititor



EXEMPLUL 9-M

□ De la problemă la program

Vom relua problema din conversațiile 7 și 8, cu deosebirea că fișierul de livrări va fi organizat ca un *fișier masiv virtual*. Fișierele masive virtuale sînt masive rezidente pe un disc magnetic, ca extensie a memoriei interne, ale căror dimensiuni sînt limitate nu mai de mărimea spațiului liber pe suportul fizic (disc). Față de masivele reale (declarate în memoria calculatorului) nu există nici o restricție privind utilizarea acestora.

Limbajul BASIC-PLUS asigură accesul la oricare din elementele fișierului, indiferent de locul pe care acestea le ocupă în fișierul de pe disc.

În cadrul acestei conversații vom proiecta și realiza un program (BASIC-PLUS) ghidat de un meniu pentru: crearea fișierului masiv virtual de livrări, adăugarea de beneficiari noi (funcțiunea nu apare în conversațiile 7 și 8), căutarea în fișier, actualizare fișier, listare fișier, sfîrșit execuție program.

Funcțiunea de „adăugare beneficiari noi” realizează, după cum precizează și numele, extinderea fișierului cu beneficiari al căror cod nu există

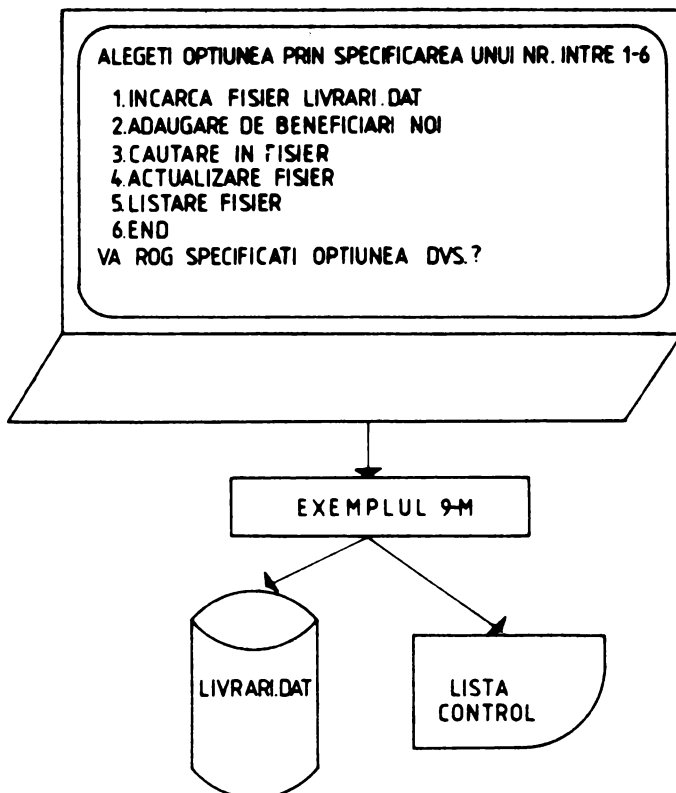


Fig. 9.1. Schema de sistem.

în fișier. Prin specificarea unui număr cuprins între 1 și 6 utilizatorul poate opta pentru una din funcțiunile precizate.

Deoarece programul are ca scop ilustrarea facilităților limbajului BASIC-PLUS pentru prelucrarea fișierelor masive virtuale de tip întreg, real și datorită faptului că sînteți experimentați cu conceptele generale de analiză și proiectare, vă vom prezenta numai schema de sistem a aplicației (v. figura 9.1).

Astfel, veți putea să vă concentrați mai mult asupra instrucțiunilor:
DIM#, OPEN ... AS VIRTUAL.

□ Codificarea în limbajul BASIC-PLUS

vol. 2, pag. 234

Meniul conversației

Programul este alcătuit dintr-un program principal (liniile 100–130) care apelează o subrutină (140) de prezentare a meniului.

```

100 REM **INTREȚINERE FIȘIER LIVRĂRI. DAT**
110 DIM # 1,C(100), Q(100), P(100)
120 GOSUB 140
130 END

140 REM SUBRUTINA DE PREZENTARE A MENIULUI
150 PRINT
160 PRINT "ALEGEȚI OPȚIUNEA PRIN SPECIFICAREA UNUI NR. INTRE 1-6"
170 PRINT
180 PRINT " 1. INCARCĂ FIȘIER LIVRĂRI. DAT"
190 PRINT " 2. ADĂUGARE DE BENEFICIARI NOI"
200 PRINT " 3. CĂUTARE ÎN FIȘIER"
210 PRINT " 4. ACTUALIZARE FIȘIER"
220 PRINT " 5. LISTARE FIȘIER"
230 PRINT " 6. END"
240 INPUT "VĂ ROG SPECIFICAȚI OPȚIUNEA DVS.", O%
250 IF O% < 7 THEN 270
260 PRINT "****OPȚIUNE ERONATĂ****" : GO TO 150
270 IF O%=1 THEN GOSUB 350
280 IF O%=2 THEN GOSUB 510
290 IF O%=3 THEN GOSUB 680
300 IF O%=4 THEN GOSUB 820
310 IF O%=5 THEN GOSUB 1000
320 GO TO 340
330
340 RETURN
350 REM*1. INCARCĂ FIȘIER LIVRĂRI. DAT
360 OPEN "LIVRĂRI. DAT" AS VIRTUAL FILE 1
370 FOR K=1 TO 100
380 C(K)=0 : Q(K)=0 : P(K)=0
390 NEXT K

```

În funcție de opțiunea indicată se tastează un număr (între 1 și 5), se selectează una din cele cinci subrutine (350, 510, 680, 820, 1000) ce aparțin subrutinei de prezentare a meniului (140). Pentru oprirea execuției programului se tastează cifra 6.

Fișiere masive virtuale. Structură, acces

Un fișier masiv virtual poate fi imaginat ca o succesiune de zone. Un masiv virtual apare utilizatorului ca o succesiune de blocuri cu lungimea de 512 octeți fiecare. Un bloc conține un număr întreg de elemente ale masivului (v. tabelul 9.1).

Tabelul 9.1

Tip masiv	Lungime element (octeți)	Număr elemente/bloc
Întreg	2	256
Real	4	128
Șir	n	512/n

Reguli

- Lungimea (m) unui element de masiv șir este obligatoriu o putere a lui 2, cuprinsă între 2 și 512;
- Referirea unui element de masiv se realizează în cadrul programului prin expresii de indici care se transformă, prin mecanismul de tratare a masivelor virtuale, într-o pereche de adrese relative;
- Bufferul alocat de calculator pentru un fișier masiv virtual este de dimensiunea unui bloc disc (512 octeți);
- În fișierul masiv virtual nu se păstrează nici o informație privind numele sau dimensiunile masivelor existente.

Declararea masivului virtual prin instrucțiunea DIM#

110 DIM#1,C(100), Q(100), P(100)

Programul prelucrează masivele virtuale C, Q și P. Pentru prelucrarea acestora este necesară declararea mai întâi a celor trei masive virtuale prin instrucțiunea DIM#. În cadrul instrucțiunii se precizează canalul de transfer (prin care se realizează transferul elementelor între memoria internă și fișierul disc suport) și lista masivelor.

Formatul general al instrucțiunii este:

Format general
DIM # <expresie numerică> , <listă>

în care:

<expresie numerică> este orice expresie numerică cu valoarea întreagă cuprinsă între 1 și 8 ce indică numărul canalului de transfer;

<listă> este o listă de elemente indicând numele masivelor și dimensiunile acestora.

Reguli

- Un fișier disc poate fi suportul mai multor masive virtuale de același tip sau de tipuri diferite (în linia 110 cele trei masive: C, Q și P au același fișier disc, deschis pe canalul 1).

- Șirurile-elemente ale unui masiv virtual de tip șir (nenumeric) au aceeași lungime fixă, declarată prin **DIM#** (v. conversația 10).
- Dacă **DIM#** nu conține declarația de lungime, aceasta se consideră implicit egală cu 16.

Opțiunea 1. Încarcă fișier **LIVRĂRI . DAT**

Subrutina realizează funcțiunea de creare a fișierului masiv virtual **LIVRĂRI . DAT**. Pentru început se deschide fișierul suport al masivului virtual pe canalul de transfer 1, prin instrucțiunea **OPEN**.

Instrucțiunea **OPEN ... AS VIRTUAL**

Pentru a obține accesul la fișierul disc, suport al masivelor virtuale C, Q, P programul trebuie să conțină o instrucțiune **OPEN** de deschidere a canalului de transfer (1), asociat fișierului „**LIVRĂRI . DAT**”.

360 **OPEN** "LIVRĂRI.DAT" **AS VIRTUAL FILE** 1

Formatul general al instrucțiunii **OPEN ... AS VIRTUAL** este:

Format general

OPEN <specificator fișier> [**FOR INPUT** | **OUTPUT AS VIRTUAL FILE** <nr. canal> [**TO MODIFY**] [, **FILESIZE** <lungime fișier>]

în care parametrii <specificator fișier>, <nr. canal>, <lungime fișier> și opțiunile **INPUT** | **OUTPUT** au aceeași semnificație ca la instrucțiunea **OPEN** (v. conversația 7).

Reguli

- Pentru un fișier masiv virtual, modul de deschidere acceptat este **MODIFY**. Modul **MODIFY** asigură accesul pentru citire, scriere, modificare.
- Pentru scrierea de noi articole, fișierul poate fi extins, pas cu pas, prin alocări succesive de spațiu suplimentar pe disc.
- Extensia fișierului pas cu pas poate fi evitată atribuind o anumită valoare (de regulă zero) ultimului element din ultimul masiv declarat pe fișierul respectiv (metoda se aplică numai atunci când se cunoaște lungimea finală a fișierului – v. și opțiunea **FILESIZE**).
- Masivul virtual (numeric, șir) nu este niciodată inițializat (cu zero, respectiv șir vid).

Inițializarea masivului virtual

Conform ultimei reguli listate (v. instrucțiunea **OPEN ... AS VIRTUAL**) intră în sarcina noastră să inițializăm (cu zero), cele trei masive numerice: C, Q și P.

```
370 FOR K=1 TO 100
380   C(K)=0 : Q(K)=0 : P(K)=0
390 NEXT K
```

TEST

Scrieți secvența de instrucțiuni de mai sus utilizând instrucțiunea matricială **MAT...**
...ZER.

Observație. Conform ultimei reguli de la instrucțiunea **OPEN...AS VIRTUAL**, cele trei masive au fost extinse prin atribuirea lui zero ultimului element al masivelor.

De la tastatură se introduc: **COD**, **CANTITATE**, **PREȚ** pentru cel mult 99 de beneficiari (v. liniile 400–500).

```

400 INPUT "COD : ";K%0
410 IF K%0 > 99 GO TO 400
420 IF K%0 = 99 GO TO 470
430 INPUT "CANTITATE : "; C2
440 INPUT "PREȚ : "; C3
450 C(K%0)=K%0 : Q(K%0)=C2 : P(K%0)=C3
460 GO TO 400
470 CLOSE 1
480 PRINT
490 PRINT "===FIȘIER INCĂRCAT==="
500 RETURN

```

Opțiunea 2. Adăugare de beneficiari noi

Subrutina realizează introducerea în fișier a noi beneficiari. Subrutina testează codul beneficiarului. În caz de eroare – codul este mai mare sau egal cu 99 se afișează mesajul "COD BENEFICIAR ERONAT". În situația când se introduce un cod deja existent în fișier se va tipări "BENEFICIAR EXISTENT – OPERAȚIE NEPERMISĂ".

```

510 REM*2. ADĂUGĂRI DE NOI ÎNREGISTRĂRI
520 OPEN "LIVRĂRI.DAT" AS VIRTUAL FILE 1
530 INPUT "PRECIZAȚI CODUL BENEFICIARULUI : "; K%0
540 IF K%0 < 99 GO TO 570
550 PRINT "COD BENEFICIAR ERONAT"
560 GO TO 530
570 IF C(K%0)=0 THEN 600
580 PRINT "BENEFICIAR EXISTENT – OPERAȚIE NEPERMISĂ"
590 GO TO 650
600 INPUT "CANTITATE : "; C2
610 INPUT "PREȚ : "; C3
620 C(K%0)=K%0 : Q(K%0)=C2 : P(K%0)=C3
630
640
650 CLOSE 1
660 PRINT "===TERMINAT ADĂUGARE==>"
670 RETURN

```

Instrucțiunea CLOSE

```
650 CLOSE 1
```

Instrucțiunea (**CLOSE**) comandă închiderea canalului de transfer (1). De notat că această închidere este obligatorie.

Formatul general al instrucțiunii este:

Format general

CLOSE <nr. canal>

unde, <nr. canal> este numărul canalului de transfer.

Regulă. Eliberarea canalelor de transfer prin care au fost prelucrate masivele virtuale poate fi realizată și cu instrucțiunile **END/CHAIN** cu observația că, în acest caz, pot fi pierdute ultimele modificări operate în masivele respective.

Opțiunea 3. Căutare în fișier

Subrutina realizează funcția de regăsire a unui beneficiar după cod. În urma căutării în fișier, subrutina afișează un mesaj corespunzător găsirii sau nu a beneficiarului indicat.

```

680 REM * 3. CĂUTARE IN FIȘIER
690 OPEN "LIVRARI.DAT" AS VIRTUAL FILE 1
700 INPUT "PRECIZAȚI CODUL BENEFICIARULUI : "; K%
710 IF K% < 99 GO TO 740
720 PRINT "COD BENEFICIAR ERONAT"
730 GO TO 700
740 A$="NU A"
750 IF C(K%)=0 THEN 770
760 A$="A"
770 CLOSE 1
780 REM BENEFICIARUL A FOST SAU NU GĂSIT IN FIȘIER
790 PRINT
800 PRINT "BENEFICIARUL CU CODUL"; K%; A$; "FOST GĂSIT IN FIȘIER"
810 RETURN

```

Opțiunea 4. Actualizare fișier

Subrutina actualizează cele două cimpuri CANTITATE și PREȚ corespunzătoare codului beneficiarului indicat. Atunci când beneficiarul nu se află în fișier se afișează mesajul "BENEFICIARUL * * * NU ESTE PREZENT IN FIȘIER".

```

820 REM * 4. ACTUALIZARE FIȘIER
830 INPUT "PENTRU CE BENEFICIAR DORIȚI ACTUALIZAREA"; K%
840 IF K% < 99 GO TO 870
850 PRINT "COD BENEFICIAR ERONAT"
860 GO TO 830
870 OPEN "LIVRARI.DAT." AS VIRTUAL FILE 1
880 IF C(K%)=0 THEN 970
890 G% = 1
900 PRINT "LA BENEF. : "; K%; "ESTE CANT. : "; Q(K%); " ȘI PREȚUL : "; P(K%)
910 PRINT "SPECIFICAȚI NOILE VALORI:"
920 INPUT "CANTITATE"; C2
930 INPUT "PREȚ"; C3
940 Q(K%)=C2 : P(K%)=C3
950
960 IF G%=1 THEN 980
970 PRINT "BENEFICIARUL"; K%; "NU ESTE PREZENT IN FIȘIER"
980 CLOSE 1
990 RETURN

```

Opțiunea 5. Listare fișier

Subrutina tipărește pe ecranul monitorului întregul conținut al fișierului.

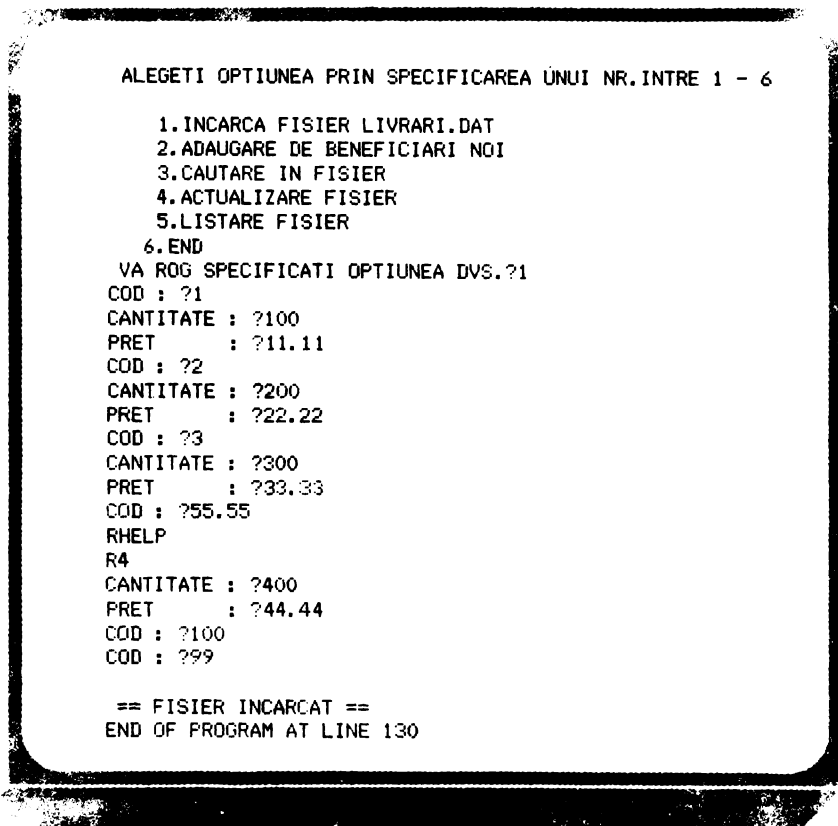
```

1000 REM * 5. LISTARE FIȘIER
1010 OPEN "LIVRARI.DAT" AS
      VIRTUAL FILE 1
1020 PRINT "COD BENEF."; TAB (30);
      "CANTITATE"; TAB (50); "PREȚ"
1030 FOR I=1 TO 100
1040   IF C(I)=0 THEN 1060
1050   PRINT C(I); TAB (30); Q(I);
      TAB (50); P(I)
1060 NEXT I
1070 CLOSE 1
1080 PRINT : PRINT "LISTARE
      TERMINATĂ"
1090 RETURN

```

Rezultatele execuției programului

Încărcarea fișierului LIVRĂRI . DAT, adăugarea de beneficiari noi, căutarea în fișier, actualizarea fișierului, listarea fișierului, oprirea execuției programului sînt ilustrate în figura 9.2.



a)

Fig. 9.2. Opțiuni: a) opțiunea 1 (la introducerea codului 55.55 de la terminal, s-a editat caracterul „R” deoarece s-a introdus pentru o variabilă întreagă (K%) o valoare reală; s-a tastat în continuare 4);

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA	DVS. ?5	
COD BENEF.	CANTITATE	PRET
1	100	11.11
2	200	22.22
3	300	33.33
4	400	44.44

LISTARE TERMINATA
END OF PROGRAM AT LINE 130

;
;

b)

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA DVS. ?2
PRECIZATI CODUL BENEFICIARULUI : ?8
CANTITATE : ?800
PRET : ?88.88
== TERMINAT ADAUGARE ==

END OF PROGRAM AT LINE 130

c)

Fig. 9.2. b) opțiunea 5; c) opțiunea 2;

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR. INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA DVS. ?5

COD BENEF.	CANTITATE	PRET
1	100	11.11
2	200	22.22
3	300	33.33
4	400	44.44
8	800	88.88

LISTARE TERMINATA
END OF PROGRAM AT LINE 130

d)

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR. INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA DVS. ?2
PRECIZATI CODUL BENEFICIARULUI : ?7
CANTITATE : ?700
PRET : ?77.77
== TERMINAT ADAUGARE ==
END OF PROGRAM AT LINE 130

e)

Fig. 9.2. d) optiunea 5; e) optiunea 2;

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

- 1.INCARCA FISIER LIVRARI.DAT
- 2.ADAUGARE DE BENEFICIARI NOI
- 3.CAUTARE IN FISIER
- 4.ACTUALIZARE FISIER
- 5.LISTARE FISIER
- 6.END

VA ROG SPECIFICATI OPTIUNEA	DVS.??	
COD BENEF.	CANTITATE	PRET
1	100	11.11
2	200	22.22
3	300	33.33
4	400	44.44
7	700	77.77
8	800	88.88

LISTARE TERMINATA
END OF PROGRAM AT LINE 130

f)

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

- 1.INCARCA FISIER LIVRARI.DAT
- 2.ADAUGARE DE BENEFICIARI NOI
- 3.CAUTARE IN FISIER
- 4.ACTUALIZARE FISIER
- 5.LISTARE FISIER
- 6.END

VA ROG SPECIFICATI OPTIUNEA DVS.??
 PRECIZATI CODUL BENEFICIARULUI : ?100
 COD BENEFICIAR ERONAT
 PRECIZATI CODUL BENEFICIARULUI : ?99
 COD BENEFICIAR ERONAT
 PRECIZATI CODUL BENEFICIARULUI : ?98
 CANTITATE : ?9800
 PRET : ?98.98
 == TERMINAT ADAUGARE ==
 END OF PROGRAM AT LINE 130

g)

Fig. 9.2. f) opțiunea 5; g) opțiunea 2;

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

- 1.INCARCA FISIER LIVRARI.DAT
- 2.ADAUGARE DE BENEFICIARI NOI
- 3.CAUTARE IN FISIER
- 4.ACTUALIZARE FISIER
- 5.LISTARE FISIER
- 6.END

VA ROG SPECIFICATI OPTIUNEA DVS.?5

COD BENEF.	CANTITATE	PRET
1	100	11.11
2	200	22.22
3	300	33.33
4	400	44.44
7	700	77.77
8	800	88.88
98	9800	98.98

LISTARE TERMINATA
END OF PROGRAM AT LINE 130

h)

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR.INTRE 1 - 6

- 1.INCARCA FISIER LIVRARI.DAT
- 2.ADAUGARE DE BENEFICIARI NOI
- 3.CAUTARE IN FISIER
- 4.ACTUALIZARE FISIER
- 5.LISTARE FISIER
- 6.END

VA ROG SPECIFICATI OPTIUNEA DVS.?3
PRECIZATI CODUL BENEFICIARULUI : ?98

BENEFICIARUL CU CODUL 98 A FOST GASIT IN FISIER
END OF PROGRAM AT LINE 130

i)

Fig. 9.2. h) optiunea 5; i) optiunea 3;

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR. INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA DVS.?3
 PRECIZATI CODUL BENEFICIARULUI : ?15

BENEFICIARUL CU CODUL 15 NU A FOST GASIT IN FISIER
 END OF PROGRAM AT LINE 130

j)

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR. INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA DVS.?4
 PENTRU CE BENEFICIAR DORITI ACTUALIZAREA ?15
 BENEFICIARUL 15 NU ESTE PREZENT IN FISIER
 END OF PROGRAM AT LINE 130

k)

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR. INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA DVS.?4
 PENTRU CE BENEFICIAR DORITI ACTUALIZAREA ?4
 LA BENEF.: 4 ESTE CANT.: 400 SI PRETUL : 44.44
 SPECIFICATI NOILE VALORI :

CANTITATE?400.44

PRET ?44.444

END OF PROGRAM AT LINE 130

l)

Fig. 9.2. j) opțiunea 3; k) opțiunea 4; l) opțiunea 4;

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR. INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA	DVS. ?5	
COD BENEF.	CANTITATE	PRET
1	100	11.11
2	200	22.22
3	300	33.33
4	400.44	44.444
7	700	77.77
8	800	88.88
98	9800	98.98

LISTARE TERMINATA
END OF PROGRAM AT LINE 130

m)

ALEGETI OPTIUNEA PRIN SPECIFICAREA UNUI NR. INTRE 1 - 6

1. INCARCA FISIER LIVRARI.DAT
2. ADAUGARE DE BENEFICIARI NOI
3. CAUTARE IN FISIER
4. ACTUALIZARE FISIER
5. LISTARE FISIER
6. END

VA ROG SPECIFICATI OPTIUNEA DVS. ?6
END OF PROGRAM AT LINE 130

n)

Fig. 9.2. m) opțiunea 5; n) opțiunea 6.

Aplicație. Să se realizeze specificațiile de programare, documentația de proiectare și programul BASIC-PLUS pentru crearea și întreținerea unui fișier masiv virtual de strunguri. Structura fișierului este următoarea: cod, uzină constructoare, puterea, diametrul maxim al piesei prelucrate și lungimea dintre virfuri. Exemplu: 100, IMU ARAD, SN 400 7.5, 400, 1000; 200, IMU ARAD, SNA 320, 5, 320, 750.

Joc pe HC-85, TIM S, SPECTRUM

```

10 REM CUTIA NEAGRA
20 DIM b(10,10,10)
30 GO TO 1000
100 REM INTRARE TIR
110 PRINT "INTRARE TIR"
120 GO SUB 500
130 LET dx=(px=2)-(px=9)
140 LET dy=(py=2)-(py=9)
150 LET dz=(pz=2)-(pz=9)
160 IF (dx=0) AND (dy=0) AND (dz=0)
    THEN GO TO 120
170 RETURN
200 REM CALCUL REZULTAT
210 FOR x=-1 TO 1
220 FOR y=-1 TO 1
230 FOR z=-1 TO 1
240 IF b(px+x,py+y,pz+z) <> 1
    THEN GO TO 280
250 LET dx=dx-x
260 LET dy=dy-y
270 LET dz=dz-z
280 NEXT z
290 NEXT y
300 NEXT x
310 LET dx=SGN dx: LET dy=SGN dy:
    LET dz=SGN dz
320 LET px=px+dx: LET py=py+dy:
    LET pz=pz+dz
330 IF (px=1)+(px=10)+(py=1)+
    (py=10)+(pz=1)+(pz=10)=0
    THEN GO TO 210
340 PRINT "REZULTAT: "; px-dx-1;
    " "; py-dy-1; " "; pz-dz-1
350 RETURN
400 REM INTRARE INCERCARE
410 PRINT "INTRARE POZIȚIE ATOM"
420 GO SUB 500

430 IF b(px,py,pz)=1 THEN PRINT
    "BRAVO!" : RETURN
440 PRINT "FALS!"
450 RETURN
500 REM INTRARE COORDONATE
510 PRINT "TASTAȚI COORDONATELE"
520 INPUT "P1="; px: LET px=px+1
530 IF px<2 OR px>9
    THEN GO TO 520
540 INPUT "P2="; py: LET py=py+1
550 IF py<2 OR py>9 THEN GO TO 540
560 INPUT "P3="; pz: LET pz=pz+1
570 IF pz<2 OR pz>9 THEN GO TO 560
580 PRINT "INTRARE: "; px-1; " ";
    py-1; " "; pz-1
590 RETURN
1000 REM PROGRAM PRINCIPAL
1010 FOR a=1 TO 5
1020 LET px=INT (RND * 6+3) : LET
    py=INT (RND * 6+3) : LET pz=
    INT (RND * 6+3)
1025 IF b(px,py,pz)=1 THEN
    GO TO 1020
1030 LET b(px,py,pz)=1
1035 NEXT a
1040 GO SUB 100: REM INTRARE TIR
1050 GO SUB 200: REM CALCUL
    REZULTAT
1060 INPUT "TIR SAU POZIȚIE ATOM
    ?:(t/p) " : t$
1070 IF t$ <> "t" AND
    t$ <> "p" THEN GO TO 1060
1080 IF t$="t" THEN GO TO 1040
1090 GO SUB 400: REM INTRARE
    INCERCARE
1100 GO TO 1060

```

TEMA 9

Răspundeți prin DA sau NU la următoarele întrebări:

- În BASIC-PLUS se pot defini masive virtuale rezidente pe un fișier disc, ca extensie a memoriei interne.
- Masivele virtuale pot fi utilizate fără nici o restricție, comparativ cu masivele reale.
- Prin utilizarea masivelor virtuale se extinde memoria internă.

- Mărima bufferului din memoria internă, în cazul unui fișier masiv virtual este diferită de dimensiunea unui bloc disc.
- În fișierul masiv virtual nu este păstrată nici o informație privind numele masivului conținut.
- În fișierul masiv virtual se păstrează informații privind dimensiunile masivelor conținute.
- Numărul de elemente/bloc pentru un masiv de tip întreg (0%) este 256.
- Instrucțiunea **OPEN ... AS VIRTUAL** deschide canalul de transfer asociat unui fișier masiv virtual.
- Masivele virtuale numerice sint automat inițializate cu zero.
- Masivele virtuale șir nu sint niciodată inițializate.
- Cifra 1 din instrucțiunea

110 DIM # 1, C2(100), C(100), P(100)

indică numărul canalului de transfer.

Inlocuiți cuvintele care lipsesc din următoarele propoziții:

- a) Masivele reale se declară în iar masivele virtuale
- b) Lungimea unui element de masiv șir este o putere a lui
- c) Pentru prelucrarea, prin program, a unui masiv virtual, trebuie realizate următoarele funcțiuni:,,
- d) Declararea masivului virtual se realizează prin instrucțiunea.....
- e) Deschiderea fișierului suport al masivului virtual pe un canal de transfer se realizează prin instrucțiunea
- f) Închiderea canalului de transfer cu instrucțiunea într-un program BASIC-PLUS este
- g) Pentru un fișier masiv virtual, singurul mod de deschidere acceptat este modul
- h) Când se dorește scrierea unor noi articole în fișier, fără a cunoaște lungimea finală a fișierului, se poate extinde fișierul, pas cu pas, prin iar în cazul că se știe lungimea finală a fișierului se
- i) Masivele virtuale numerice trebuie inițializate prin program cu iar masivele virtuale șir trebuie inițializate cu
- j) În secvența de instrucțiuni:

```
10 DIM #1, A(100)
20 OPEN "A" AS VIRTUAL FILE 2
30 CLOSE 1
```

s-au strecurat următoarele greșeli:,,

k) În cadrul următorului program de creare a unui fișier masiv virtual de articole (cel mult 29) structurat în: COD, CANTITATE

```
10 DIM #1, C1(30), C(30)
20 OPEN "A.DAT" AS VIRTUAL FILE 1
```



```

30 C(30)=0
40 INPUT "COD : "; K%
50 IF K% > 29 GO TO ...
60 IF K%=29 GO TO ...
70 INPUT "CANTITATE : "; C2
80 C1(K)=K% : C(K%)=C2
90 GO TO 40
100 CLOSE 2
110 PRINT
120 PRINT "FIȘIERUL A . DAT A FOST CREAT"
130 END

```

s-au strecurat greșeli în liniile , ,

În linia 50 instrucțiunea **GOTO** este urmată de eticheta iar în linia 60 trebuie adăugat după instrucțiunea **GO TO** numărul de linie
 Instrucțiunea de atribuire din linia 30:

```
30 C(30)=0
```

a fost scrisă pentru

În acest caz s-a aplicat regula: când se cunoaște lungimea finală a fișierului, se poate evita extensia pas cu pas printr-o singură extensie a fișierului atribuind (de regulă) valoarea , element din ultimul masiv declarat în fișierul respectiv.

Scrieți cite un program BASIC-PLUS pentru fiecare din problemele de mai jos:

a) Să se creeze un fișier masiv virtual de livrări având structura: COD BENEFICIAR, CANTITATE, PREȚ.

b) Să se creeze și să se întrețină un fișier masiv virtual ce conține zilele onomastice ale celor dragi. Se va realiza un program BASIC-PLUS ghidat de un meniu.

c) Să se creeze și să se întrețină un fișier masiv virtual ce conține rezultatele meciurilor de rugby susținute de echipele divizionare A. Se va realiza un program ghidat de un meniu.

Să se gestioneze și să se întrețină pe minicalculatoarele INDEPENDENT și CORAL fondul de date privind cantitățile lunare de jucării fabricate de Întreprinderea URSULEȚUL.

Se va proiecta și realiza un program BASIC-PLUS ghidat de un meniu pentru crearea, actualizarea și listarea fișierului masiv virtual JUCĂRII .DAT. Structura înregistrării este cea cunoscută (v. conversațiile precedente).

Observație. Se vor utiliza subrutinele din EXEMPLUL 9-M.

Să se proiecteze și să se realizeze un program BASIC-PLUS ghidat de un meniu pentru crearea și întreținerea fișierului masiv virtual CONSUMURI . DAT a cărui structură a fost definită în conversațiile precedente.

SOLUȚIA TEMEI 9

Răspunsurile sint: DA, DA, DA, NU, DA, NU, DA, DA, NU, DA, DA.

Cuvintele lipsă sint:

a) memoria internă/pe fișiere disc; b) 2, cuprinsă între 2 și 512; c) declararea masivului virtual, deschiderea fișierului suport al masivului virtual, închiderea canalului de transfer asociat fișierului suport; d) DIM#; e) OPEN ... AS VIRTUAL; f) CLOSE/obligatorie; g) MODIFY; h) alocări succesive/se atribuie, de regulă, zero ultimului element din ultimul masiv declarat; i) zero/șir vid; j) în linia 20 trebuie scris FILE 2; k) linia 80 : C1 (K⁰/₀)=K⁰/₀; linia 100 : CLOSE 2; 50 IF K⁰/₀>29 GO TO 40; 60 IF K⁰/₀=29 GO TO 100; Instrucțiunea de atribuire din linia 30 a fost scrisă pentru extinderea fișierului; vezi h.

Programele BASIC sint:

a)

```

100 REM ** CREARE FIȘIER LIVRĂRI . DAT **
110 DIM #1, C1(100), C(100), P(100)
120 PRINT
130 OPEN "LIVRĂRI . DAT" AS VIRTUAL FILE 1
140 P(100)=0
150 PRINT "VA ROG INTRODUCEȚI: COD BENEFICIAR/CANTITATE/PREȚ"
160 PRINT
170 INPUT "COD : " ; K%
180 IF K% > 99 GO TO 170
190 IF K%=99 GO TO 240
200 INPUT "CANTITATE : " ; C2
210 INPUT "PREȚ : " ; C3
220 C1 (K%/10)=K%/10 : C(K%/10)=C2 : P(K%/10)=C3
230 GO TO 160
240 CLOSE 1
250 PRINT
260 PRINT "FIȘIERUL LIVRĂRI . DAT A FOST CREAT"
270 PRINT
280 END

```

Nu răspundem.

Nu răspundem.

CONVERSAȚIA 10

Proiectarea și realizarea programului principal pentru crearea și întreținerea unui fișier masiv virtual de livrări (adăugare, consultare, numărare, ștergere, modificare) care apelează mai multe subrutine al căror format sursă se cunoaște. Instrucțiuni BASIC-PLUS pentru prelucrarea fișierelor masive virtuale de tip șir. Depanarea unui program. JOCURI, APLICAȚII și TESTE pentru cititor



EXEMPLUL 10-M

□ De la problemă la program

Vom aborda și în cadrul acestei conversații problema creerii și întreținerii unui fișier masiv virtual de livrări, cu singura deosebire că structura fișierului se modifică în: denumire beneficiar, cantitate livrată și preț.

Funcțiunile programului sînt: creare fișier, adăugare beneficiari, consultare fișier, numărare înregistrări existente și listare fișier, ștergere înregistrări, modificare înregistrări, oprirea execuției programului.

Programul are ca obiectiv ilustrarea facilităților limbajului BASIC-PLUS pentru prelucrarea fișierelor masive virtuale de tip șir el fiind ghidat ca și programele conversațiilor precedente de un meniu. Datorită faptului că dvs. aveți deja experiența conversației cu minicalculatoarele INDEPENDENT și CORAL vă vom prezenta, de această dată în plus, pe lângă specificațiile de programare (v. modulul de analiză structurată, figura 10.1),

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLUL 10 – M

Descrierea programului

Programul creează și întreține un fișier masiv virtual de livrări. Programul este ghidat de un meniu care permite alegerea uneia din următoarele opțiuni: crearea fișier, adăugare beneficiari, consultare fișier, numărare înregistrări existente și listare fișier, ștergere (din fișier), modificare înregistrare, sfîrșit sesiune actuală.

Intrări

Se introduc de la tastatură: codul opțiunii (un număr între 1 și 7), numele beneficiarului, cantitatea livrată, preț tonă benzină.

Ieșiri

Fișierul masiv virtual BENEF.DAT.

Lista de funcțiuni ale programului

1. Deschidere fișier
2. Inchidere fișier
3. Afișare cap tabel BENEFICIAR/CANTITATE LIVRATĂ/PREȚ
4. Inițializare I3
5. Căutare beneficiari înscriși în fișier
6. Incrementare variabilă I3
7. Inițializare variabilă indexată B\$ cu șir vid
8. Inițializare variabile indexate P, C cu zero
9. Incrementare variabilă T1
10. Afișare mesaj "FIȘIERUL EXISTĂ"
11. Testare condiție fișier plin
12. Afișare mesaj "FIȘIER ESTE PLIN * ADAUGARILE SE REFUZA" în caz de umplere fișier
13. Citire nume beneficiar în variabila NS
14. Inițializare variabilă T cu zero
15. Inițializare variabilă A cu 31
16. Identificare prima căsuță goală
17. Inițializare variabilă T cu 1 în caz că beneficiarul există în fișier
18. Afișare mesaj "BENEFICIARUL x x x ESTE DEJA ÎN FIȘIER"
19. Reținere nume beneficiar dacă $A < 31$
20. Citire cantitate în variabila C(A)
21. Citire preț în variabila P(A)
22. Inițializare variabilă T cu zero
23. Detectare prezență beneficiar solicitat
24. Testare condiție $T < > 0$ și tipărire cap tabel
25. Tipărire beneficiar, cantitate, preț
26. Inițializare variabilă T cu 1
27. Tipărire mesaj "A FOST ȘTERS BENEFICIARUL" atunci cînd $T=1$
28. Tipărire mesaj "BENEFICIAR INEXISTENT ÎN FIȘIER" atunci cînd T este diferit de 1
29. Detectare prezență beneficiar solicitat
30. Afișare mesaj "SPECIFICAȚI MODIFICARILE DORITE" atunci cînd T este diferit de zero

a)

Fig. 10.1. Modulul de analiză structurată: a) specificații de programare;

SPECIFICAȚII DE PROGRAMARE

- | | |
|---|---|
| <p>31. Afișare mesaj "BENEFICIARUL x x x
NU EXISTA IN FIȘIER"
atunci cind T este egal cu zero</p> <p>32. Citire nume beneficiar in variabila M\$</p> <p>33. Citire cantitate in variabila C1</p> <p>34. Citire preț in variabila P1</p> <p>35. Atribuire valoare variabilă M\$
lui B\$ (T)</p> <p>36. Atribuire valoare variabilă C1 lui C(T)</p> <p>37. Atribuire valoare variabilă P1 lui P(T)</p> <p>38. Afișare mesaj "NOUA INREGISTRARE"</p> | <p>39. Specificarea opțiunii:</p> <p>1 Creare fișier beneficiari</p> <p>2 Adăugare beneficiari</p> <p>3 Consultare fișier beneficiari</p> <p>4 Numărare înregistrări existente
și listare fișier</p> <p>5 Ștergere din fișier beneficiari</p> <p>6 Modificare înregistrare beneficiari</p> <p>7 Sfârșit sesiune actuală</p> <p>40. Stop</p> |
|---|---|

a) (continuare)

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M

Variabile de intrare	Variabile de stare	Variabile de ieșire
<p>O: opțiune meniu</p> <p>N\$: nume beneficiar</p> <p>M\$: nume beneficiar</p> <p>C1: cantitate</p> <p>P1: Preț</p>	<p>T1: stare fișier</p> <p>I3: variabilă contorizare număr beneficiari</p> <p>T: stare beneficiar</p> <p>A: locație goală (index)</p> <p>B\$: nume beneficiar</p> <p>C: cantitate</p> <p>P: preț</p> <p>I, P9: indecși</p>	<p>B\$: nume beneficiar</p> <p>C: cantitate</p> <p>P: preț</p>

b)

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M (Subrutina 450)

Variabile de intrare	Variabile de stare	Variabile de ieșire
	<p>I: variabila de control</p> <p>B\$: nume beneficiar</p> <p>C: cantitatea</p> <p>P: preț</p> <p>T1: stare fișier</p>	

c)

Fig. 10.1. b) tabela de variabile; c) tabela de variabile (subrutina 450);

TABELA DE VARIABLE

Nume program: EXEMPLUL 10 – M (subrutina 570)

Variabile de intrare	Variabile de stare	Variabile de ieșire
N\$: nume beneficiar C : cantitate P : preț	T1 : stare fișier I3 : variabilă contorizare număr beneficiari T : stare beneficiar (există/nu există în fișier) A : locație goală (index) B\$: nume beneficiar din fișier	B\$: nume beneficiar C : cantitate P : preț

d)

TABELA DE VARIABLE

Nume program: EXEMPLUL 10 – M (subrutina 800)

Variabile de intrare	Variabile de stare	Variabile de ieșire
N\$: nume beneficiar	T1 : Stare fișier T : stare beneficiar B\$: nume beneficiar (din fișier) I : index	B\$: nume beneficiar C : cantitate P : preț

e)

TABELA DE VARIABLE

Nume program: EXEMPLUL 10 – M (subrutina 950)

Variabile de intrare	Variabile de stare	Variabile de ieșire
N\$: numele beneficiarului	T : stare beneficiar B\$: nume beneficiar (din fișier)	N\$: numele beneficiarului P : preț C : cantitate

f)

TABELA DE VARIABLE

Nume program: EXEMPLUL 10 – M (subrutina 1130)

Variabile de intrare	Variabile de stare	Variabile de ieșire
N\$: nume beneficiar M\$: nume beneficiar C1 : cantitate P1 : preț	T : stare beneficiar T1 : stare fișier B\$: nume beneficiar (din fișier)	N\$: nume beneficiar B\$: nume beneficiar C : cantitate P : preț

g)

Fig. 10.1. d) tabela de variabile (subrutina 570); e) tabela de variabile (subrutina 800); f) tabela de variabile (subrutina 950); g) tabela de variabile (subrutina 1130);

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M (subrutina 1350)

Variabile de intrare	Variabile de stare	Variabile de ieșire
	T1: stare fișier I3: variabilă contorizare număr beneficiari P9: index (variabilă de control)	B\$: nume beneficiar C: cantitate P: preț

h)

TABELA DE VARIABILE

Nume program: EXEMPLUL 10 – M (subrutina 1630)

Variabile de intrare	Variabile de stare	Variabile de ieșire
	B\$: nume beneficiar I3: Variabilă contorizare număr beneficiari	I3: variabilă contorizare număr beneficiari

i)

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

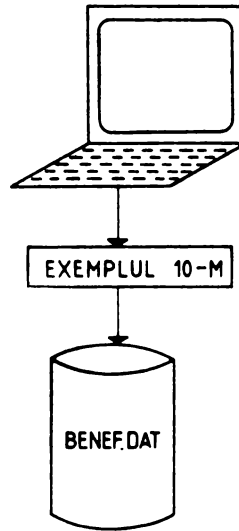
Nume program: EXEMPLUL 10 – M

Modul	Funcțiuni
DESID-1470	1
INIT	7, 8, 9, 10
INCHID-1530	2
NUMARA-1630	1, 4, 5, 6, 2
PREL-AD	9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21
PREL-CONS	9, 22, 13, 23, 24, 25
CAP-TAB-1590	3, 4, 5, 6, 2
PREL-STERG	22, 9, 13, 7, 8, 26, 27, 28
PREL-MOD	22, 9, 13, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 25

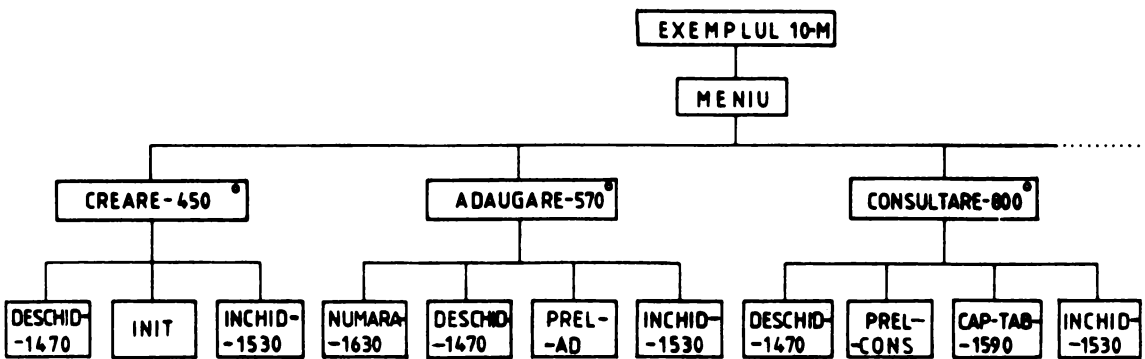
j)

Fig. 10.1. h) tabela de variabile (subrutina 1350); i) tabela de variabile (subrutina 1630); j) alocarea funcțiilor de prelucrare.

documentația de proiectare (v. modulul de proiectare structurată, fig. 10.2) și listele cu programul sursă al subrutinelor ce intră în structura programului.

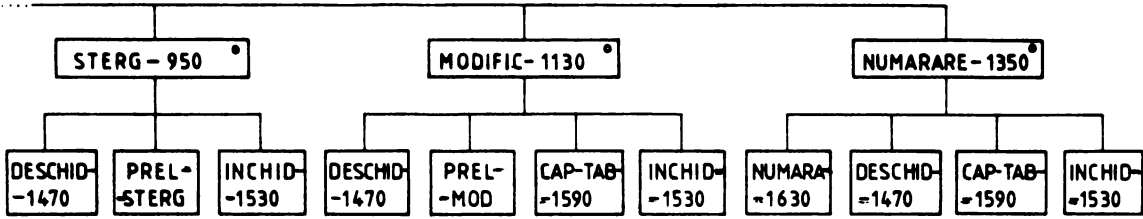


a)

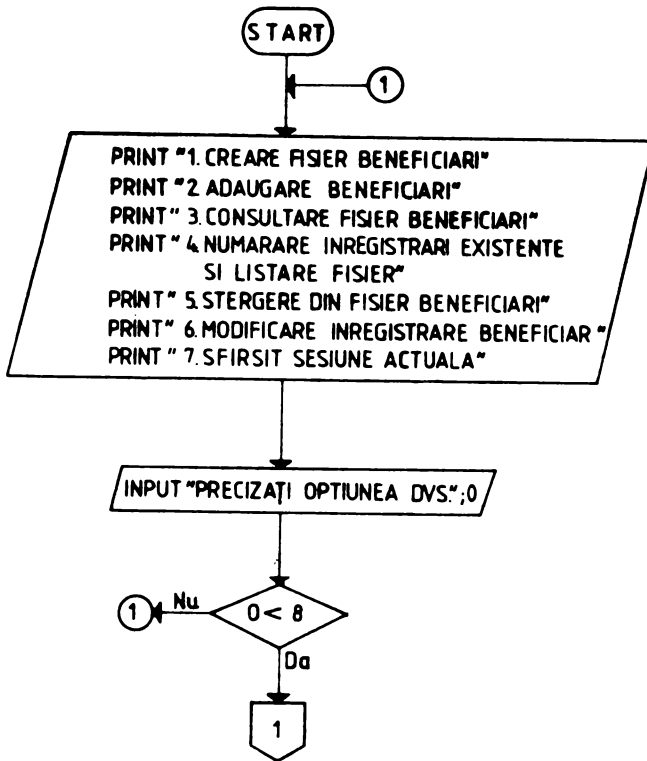


b)

Fig. 10.2. Modulul de proiectare structurată: a) schema de sistem; b) diagrama de structură.

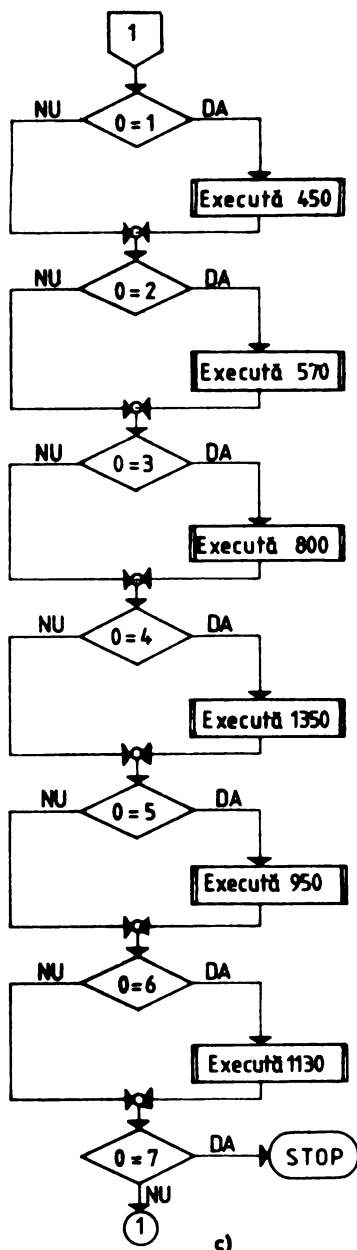


b) (continuare)



c)

Fig. 10.2. c) schema logică.



STRUCTURA ÎNREGISTRĂRII

 Nume program: EXEMPLUL 10 – M

Nume beneficiar

Cantitate livrată

Preț

 Fig. 10.2. d) structura înregistrării.

Astfel, veți putea să vă îndreptați atenția mai mult asupra proiectării și realizării programului principal al aplicației urmînd ca singuri să asamblați subrutinele al căror program sursă vi se oferă (v. figura 10.3).

 EXEMPLUL 10-M (subrutina 450)

```

450 REM ***** CREARE FIȘIER BENEFICIARI
460 IF T1 ( ) 0 THEN 530
470 GO SUB 1470 ! DESCHIDERE
480 FOR I=1 TO 30
490   B$(I)=" " : P(I)=0 : C(I)=0
500 NEXT I
510 T1=T1+1
520 GO TO 550
530 PRINT "FIȘIERUL EXISTĂ"
540 GO TO 340
550 GO SUB 1530 ! INCHIDERE
560 RETURN
  
```

a)

 Exemplul 10-M (subrutina 570)

```

570 REM ***** ADĂUGARE BENEFICIARI NOI
580 T1=T1+1
590 GO SUB 1630 ! NUMĂRARE ARTICOLE
600 GO SUB 1470 ! DESCHIDERE
610 IF I3=30 THEN 710
620 INPUT "NUMELE BENEFICIARULUI : ", N$
630 T=0
640 A=31
650 FOR I=30 TO 1 STEP -1
660   IF C(I)=0 THEN A=I ! REȚIN PRIMA CĂSUȚĂ GOALĂ
670   IF N$=B$(I) THEN T=1 ! BENEFICIAR EXISTENT ÎN FIȘIER
680 NEXT I
690 IF T=1 THEN 770
700 IF A<31 THEN 730
710 PRINT "*FIȘIERUL ESTE PLIN*ADĂUGAREA SE REFUZĂ*"
720 GO TO 780
730 B$(A)=N$ : I3=I3+1 ! REȚIN NUMELE BENEFICIARULUI
740 INPUT "CANTITATE", C(A) ! REȚIN CANTITATEA
750 INPUT "PREȚ", P(A) ! REȚIN PREȚUL
760 GO TO 780
770 PRINT "BENEFICIARUL"; N$; "ESTE DEJA PREZENT ÎN FIȘIER"
780 GO SUB 1530 ! INCHIDERE
790 RETURN
  
```

b)

Fig. 10.3. Programele sursă ale subrutinelor; a) subrutina 450; b) subrutina 570;

EXEMPLUL 10-M (subrutina 800)

```

800 REM ***** CONSULTARE FIȘIER BENEFICIARI
810 T1=T1+1
820 GO SUB 1470 ! DESCHIDERE
830 T=0
840 INPUT "NUMELE BENEFICIARULUI : ", N$
850 FOR I=1 TO 30
860   IF N$=B$(I) THEN T=I ! DETECTARE PREZENTĂ BENEFICIAR SOLICITAT
870 NEXT I
880 IF T <> 0 THEN 910
890 PRINT "BENEFICIARUL"; N$; "NU EXISTĂ ÎN FIȘIER"
900 GO TO 930
910 GO SUB 1590
920 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
930 GO SUB 1530 ! ÎNCHIDERE
940 RETURN

```

c)

EXEMPLUL 10-M (subrutina 1350)

```

1350 REM SRT DE NUMĂRARE A ÎNREG. EXISTENTE ÎN FIȘIER
1360 T1=T1+1
1370 GO SUB 1630 ! NUMĂRARE
1380 PRINT : PRINT
1390 PRINT "FIȘIERUL CONȚINE"; I3; "ÎNREGISTRĂRI"
1400 GO SUB 1470
1410 GO SUB 1590
1420 PRINT B$(P9), C(P9), P(P9) FOR P9=1 TO I3
1430 PRINT
1440 GO SUB 1530
1450 RETURN

```

d)

EXEMPLUL 10-M (subrutina 950)

```

950 REM ***** ȘTERGERE BENEFICIAR DIN FIȘIER
950 T=0
970 T1=T1+1
980 INPUT "NUMELE BENEFICIARULUI : ", N$
990 GO SUB 1470 ! DESCHIDERE
1000 FOR I=1 TO 30
1010   IF N$ <> B$(I) THEN 1060
1020   B$(I)=" "
1030   T=1
1040   P(I)=0
1050   C(I)=0
1060 NEXT I
1070 IF T=1 THEN 1100
1080 PRINT "BENEFICIAR INEXISTENT ÎN FIȘIER"
1090 GO TO 1110
1100 PRINT "A FOST ȘTERS BENEFICIARUL"; N$
1110 GO SUB 1530 ! ÎNCHIDERE
1120 RETURN

```

e)

Fig. 10.3. c) subrutina 800; d) subrutina 1350; e) subrutina 950;

EXEMPLUL 10-M (subrutina 1130)

```

1130 REM ***** MODIFICARE INREGISTRARE BENEFICIAR
1140 GO SUB 1470 ! DESCHIDERE
1150 T=0 : T1=T1+1
1160 INPUT "NUMELE BENEFICIARULUI : ", N$
1170 FOR I=1 TO 30
1180 IF N$(I) THEN T=I ! DETECTARE PREZENTĂ BENEFICIAR SOLICITAT
1190 NEXT I
1200 IF T <> 0 THEN 1230
1210 PRINT "BENEFICIARUL"; N$; "NU EXISTĂ IN FIȘIER"
1220 GO TO 1330
1230 PRINT "SPECIFICAȚI MODIFICĂRILE DORITE"
1240 INPUT "NUME BENEFICIAR", M$
1250 INPUT "CANTITATE", C1
1260 INPUT "PREȚ", P1
1270 IF M$ <> " " THEN B$(T)=M$
1280 IF C1 <> 0 THEN C(T)=C1
1290 IF P1 <> 0 THEN P(T)=P1
1300 PRINT "NOUĂ INREGISTRARE : "
1310 GO SUB 1590
1320 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
1330 GO SUB 1530 ! INCHIDERE
1340 RETURN

```

f)

EXEMPLUL 10-M (subrutinele 1470, 1530, 1590, 1630)

```

1460 REM SUBROUTINE
1470 REM DESCHIDERE FIȘIER
1480 OPEN "BENEF. DAT" AS VIRTUAL FILE 1
1490 PRINT
1500 PRINT " ***INCEPUT FAZĂ ***"
1510 PRINT
1520 RETURN
1530 REM INCHIDERE FIȘIER
1540 CLOSE 1
1550 PRINT
1560 PRINT " ***TERMINAT FAZA ***"
1570 PRINT
1580 RETURN
1590 REM CAP TABEL
1600 PRINT "BENEFICIAR /CANTITATE LIVRATĂ/ PREȚ"
1610 PRINT " ..... " : PRINT
1620 RETURN
1630 REM SUBROUTINĂ NUMĂRARE BENEFICIARI INSCRIȘI IN FIȘIER
1640 OPEN "BENEF. DAT" AS VIRTUAL FILE 1
1650 I3=0
1660 FOR I=1 TO 30
1670 IF B$(I)=" " THEN 1690
1680 I3=I3+1
1690 NEXT I
1700 CLOSE 1
1710 RETURN

```

g)

Fig. 10.3. f) subrutina 1 130; g) subrutinele 1 470, 1 530, 1 590, 1 630.

□ Codificarea în limbajul BASIC-PLUS

vol. 2, pag. 236

Programul conversației

Programul principal al conversației cuprinde liniile 100–440.

```

100 REM PROGRAMUL CREAȚĂ ȘI ÎNTREȚINE UN FIȘIER DE BENEFICIARI
110 REM FUNCȚIUNILE PROGRAMULUI:
120 REM – CREARE FIȘIER BENEFICIARI
130 REM – ADĂUGĂRE BENEFICIARI
140 REM – CONSULTARE FIȘIER BENEFICIARI
150 REM – ȘTERGERE BENEFICIARI
160 REM – LISTARE BENEFICIARI
170 REM – MODIFICARE BENEFICIAR EXISTENT
180 REM DEFINIREA FIȘIERULUI VIRTUAL
190 DIM #1, B$(30)=40, C(30), P(30), B1$(30, 30)
200 REM B$ – NUME BENEFICIAR
210 REM C – CANTITATE
220 REM P – PREȚ
230 PRINT
240 PRINT " 1. CREARE FIȘIER BENEFICIARI"
250 PRINT " 2. ADĂUGĂRE BENEFICIARI"
260 PRINT " 3. CONSULTARE FIȘIER BENEFICIARI"
270 PRINT " 4. NUMĂRARE ÎNREGISTRĂRI EXISTENTE ȘI LISTARE FIȘIER"
280 PRINT " 5. ȘTERGERE DIN FIȘIER BENEFICIARI"
290 PRINT " 6. MODIFICARE ÎNREGISTRARE BENEFICIAR"
300 PRINT " 7. SFIRȘIT SESIUNE ACTUALĂ"
310 PRINT
320 INPUT "PRECIZAȚI OPȚIUNEA DVS."; O
330 O < 8 THEN 360
340 PRINT "OPȚIUNE ERONATĂ"
350 GO TO 230
360 IF O=1 THEN GO SUB 450
370 IF O=2 THEN GO SUB 570
380 IF O=3 THEN GO SUB 800
390 IF O=4 THEN GO SUB 1350
400 IF O=5 THEN GO SUB 950
410 IF O=6 THEN GO SUB 1130
420 IF O=7 THEN GO TO 1720
430 GO TO 230
440 PRINT

```

După ce în liniile 240–300 se listează meniul (cu cele șapte opțiuni), în linia 320 programul citește dinamic opțiunea utilizatorului. În cazul cînd pentru variabila de intrare O se introduce o valoare mai mare sau egală cu 8, se afișează mesajul "OPȚIUNE ERONATĂ" și ciclul se reia. Funcție de opțiunea aleasă programul selectează una din subrutinele: 450, 570, 800, 1350, 950, 1130 pe care vă invităm să le analizați în detaliu.

TEST

Trasați schemele logice ale subrutinelor din program.

În cele ce urmează ne vom opri numai asupra posibilităților limbajului BASIC-PLUS privind prelucrarea fișierelor masive virtuale de tip șir.

În linia 190 a programului principal am declarat numărul virtual de tip șir B\$ (numele beneficiarului).

190 DIM# 1, B\$(30)=40, C(30), P(30), B1\$(30, 30)

În limbajul BASIC-PLUS șirurile-elemente ale unui masiv virtual de tip șir (nenumeric) au aceeași lungime fixă, declarată prin DIM# (v. linia 190).

Dacă această lungime declarată (40) nu este o putere a lui 2, cuprinsă între 2 și 512, ea este implicit transformată la puterea lui 2 imediat superioară. Astfel, linia 190 declară masivul virtual B\$ ca avînd elemente de lungime maximă 64, deoarece $32 < 40 < 64$.

Formatul general al instrucțiunii DIM# pentru declararea masivelor virtuale de tip șir este:

Format general
DIM# (expresie numerică) (listă)

în care: (expresie numerică) are semnificația știută (v. conversația 9); (listă) reprezintă o listă de elemente ce precizează numele masivelor și dimensiunilor acestora.

Pentru masivele șir, elementele pot fi de forma:

Format general
<masiv>((dim 1) [, <dim 2>]) [= <constantă întreagă>] <variabila întreagă>

unde:

<masiv> reprezintă un nume de variabilă indexată;

<dim 1>, <dim 2> reprezintă dimensiunile declarate ale masivelor; poate fi orice constantă sau variabilă simplă întreagă;

<constantă întreagă> și <variabilă întreagă> reprezintă lungimea maximă admisă pentru elementele masivului șir.

Reguli

- Un element aparținînd unui masiv șir nu poate depăși lungimea declarată prin DIM#.
- Un element al unui masiv șir poate avea o lungime mai mică decît cea declarată prin DIM#, în acest caz făcîndu-se alinierea sa la stînga și completîndu-se spațiul disponibil cu caracterul nul (cod ASCII=0).
- Dacă DIM# nu conține declarația de lungime, aceasta se consideră implicit egală cu 16.
- Pentru referirea unui masiv creat într-un alt program trebuie să fie menționată corect poziția relativă a acestui masiv față de începutul fișierului (cazul masivului B1\$ declarat în linia 190; acesta va fi folosit în programul din conversația următoare).

Aplicație. Să se realizeze specificațiile de programare, documentația de proiectare și programul BASIC pentru crearea, consultarea, actualizarea și listarea unui fișier masiv virtual de profiluri standardizate.

Profilurile sînt bare metalice cu secțiunea transversală constantă și cu o anumită formă standardizată sau prescrisă printr-o normă. Profilurile se folosesc în construcții metalice civile, în construcții de coloane tubulare și alte utilaje, în construcții de nave, de material rulant, de autovehicule etc.

Din punct de vedere al procedeelor de fabricare se deosebesc următoarele tipuri de profiluri: profiluri fasonate, laminate la cald din oțeluri; profiluri fasonate formate la rece din bandă, profiluri simple laminate la cald din oțeluri, profiluri simple calibrate etc.

În tabelul 10.1 se prezintă mai multe tipuri de profiluri standardizate cărora li se precizează: simbolul, denumirea, tipul (structurat în cod și denumire), secțiunea (codurile

Tabelul 10.1

Simbol profil	Denumire profil	Tip profil Cod	Denumire	Sec- țiune	Simbol general	Dimensiuni secțiune	Lungime de fabricație	STAS
1	2	3	4	5	6	7	8	9
L	oțel cornier cu aripi egale	1	fasonate, laminare la cald din oțel	1	axaxg	a= 20 ... 160 g= 3 ... 18	3 000 ... 12 000	424-80
LL	oțel cornier cu aripi neegale	1	fasonate, laminare la cald din oțel	2	axbxg	a= 30 ... 150 b= 20 ... 100 g= 3 ... 134	4 000 ... 12 000	425-80
I	oțel I	1	fasonate, laminare la cald din oțel	3	lh	h= 80 ... 400 b= 42 ... 155 g= 3,9 ... 14,4	5 000 ... 15 000 (6 000 ... 15 000)	565-80
IE	oțel I economic	1	fasonate, laminare la cald din oțel	3	IEh	h= 100 ... 400 b= 55 ... 155 g= 4,5 ... 8,0	8 000 ... 15 000 (6 000 ... 15 000)	7550-77
T	oțel cu aripi egale și muchii rotunjite	1	fasonate, laminare la cald din oțel	4	Th	h= 20 ... 50 a=h=20 ... 50 g= 3 ... 6	4 000 ... 8 000	566-80
U	oțel U	1	fasonate, laminare la cald din oțel	5	Uh	h= 65 ... 300 b= 42 ... 100 g= 5,5 ... 10,0	5 000 ... 15 000 (6 000 ... 15 000)	564-80
UE	oțel U economic	1	fasonate, laminare la cald din oțel	5	UEh	h= 50 ... 300 b= 32 ... 100 g= 4,4 ... 6,5	8 000 ... 15 000 (6 000 ... 15 000)	7551-77

LI	profil cornier cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	axaxg	a=16...125 g=1...5	4 000...8 000	7836-80
LLI	profil cornier au aripi neegale	2	fasonate, formate la rece din bandă de oțel	0	axbxg		4 000...8 000	8250-80
TI	profil T cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	axhxg	h=30 și 40 a=h g=2	4 000...8 000	8249-80
UI	profil U cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	hxbxg	h=16...200 b=10...100 g=1,5...5,0	4 000...8 000	7835-80
UNI	profil U cu aripi neegale	2	fasonate, formate la rece din bandă de oțel	0	hxBxbxg	h=20...140 B=20...100 b=16...60 g=1,5...5,0	4 000...8 000	8610-80
ZI	profil Z cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	hxbxg	h=20...100 b=16...50 g=1,5...4,0	4 000...8 000	8296-80
ZNI	profil Z cu aripi neegale	2	fasonate, formate la rece din bandă de oțel	0	hxBxbxg	h=25...120 B=20...50 b=16...40 g=1,5...4,0	4 000...8 000	8609-80

Tabelul 10.1 (continuare)

1	2	3	4	5	6	7	8	9
ΩI	profil omogen cu aripi egale	2	fasonate, formate la rece din bandă de oțel	0	hxbxag	h=20...100 b=20...50 a=10...32 g=1...3	4 000...8 000	8367-80
LT	oțel lat	3	simple, laminat la cald sau trase la rece din oțeluri	6	axb	a=20...150 b=5...50	3 000...7 000 (1 500...7 000)	395-80
LTC	oțel lat calibrat	3	simple, laminat la cald sau trase la rece din oțeluri	6	axb	a=18...50 b=5...16	2 000...6 000 (1 500...6 000)	6972-80
4L	oțel pătrat	3	simple, laminat la cald sau trase la rece din oțeluri	7	4La	a=8...60	3 000...9 000 (1 500...9 000)	334-80
4LC	oțel pătrat calibrat	3	simple, laminat la cald sau trase la rece din oțeluri	7	Ta	a=3,5...50,0	1 500...6 000 (500...6 000)	6554-80

6L	oțel hexagonal	3	simple, laminat la cald sau trase la rece din oțeluri	8	6LS	S=8...72	2 000...6 000	7828-78
TR	oțel hexagonal calibrat	3	simple, laminat la cald sau trase la rece din oțeluri	8	TRS	S=4...70	1 500...6 000	2305-80
Ø	oțel rotund	3	simple, laminat la cald sau trase la rece din oțeluri	9	Ød	d=10...60	3 000...10 000	333-80
TS	oțel rotund calibrat	3	simple, laminat la cald sau trase la rece din oțeluri	9	TSd	d=2...70	1 500...6 000	1800-80

secțiunii corespund figurii 10.4), simbolul general, dimensiunile secțiunii, lungimea de fabricație și STAS-ul.

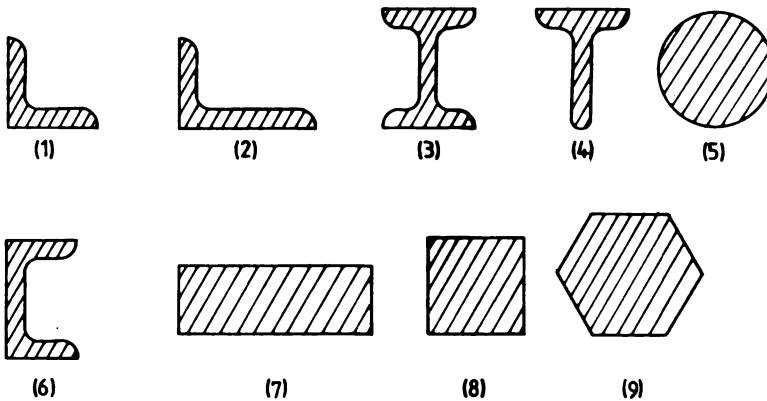


Fig. 10.4. Profiluri standardizate.

Observație. Se recomandă ca programul BASIC-PLUS pe care-l realizați să fie ghidat de către un meniu.

Rezultatele opțiunilor

Rezultatele conversației cu minicalculatoarele INDEPENDENT și CORAL prin intermediul programului EXEMPLUL 10-M sînt ilustrate în figura 10.5.

```

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

```

PRECIZATI OPTIUNEA DVS. ?1

**** INCEPUT FAZA ****

**** TERMINAT FAZA ****

a)

Fig. 10.5. Opțiuni: a) opțiunea 1;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.??

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PECO TITAN
 CANTITATE ?1000
 PRET ?7500

**** TERMINAT FAZA ****

b)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.??

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PECO MILITARI
 CANTITATE ?30000
 PRET ?7500

**** TERMINAT FAZA ****

c)

Fig. 10.5. b) opțiunea 2; c) opțiunea 2;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE 2 INREGISTRARI

**** INCEPUT FAZA ****

BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PECO TITAN	1000	7500
PECO MILITARI	30000	7500

**** TERMINAT FAZA ****

d)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PECO-COLENTINA
 CANTITATE ?1000
 PRET ?7100

**** TERMINAT FAZA ****

e)

Fig. 10.5. d) optiunea 4; e) optiunea 2;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?1
 FISIERUL EXISTA
 OPTIUNE ERONATA

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PECO-DR.TABEREI
 CANTITATE ?3000
 PRET ?8000

**** TERMINAT FAZA ****

f)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?CRAIOVA-VALEA ROSIE
 CANTITATE ?4000
 PRET ?6000

**** TERMINAT FAZA ****

g)

Fig. 10.5. f) opțiunea 1; g) opțiunea 2;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PLOIESTI-BARIERA
 CANTITATE ?3000
 PRET ?7000

**** TERMINAT FAZA ****

h)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?BUZAU-CENTRU
 CANTITATE ?3000
 PRET ?7800

**** TERMINAT FAZA ****

i)

Fig. 10.5. h) opțiunea 2; i) opțiunea '2;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS. ?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?IASI-COPOU
CANTITATE ?300
PRET ?7890

**** TERMINAT FAZA ****

j)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS. ?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?URZICENI
CANTITATE ?1000
PRET ?7000

**** TERMINAT FAZA ****

k)

Fig. 10.5: j) opțiunea 2; k) opțiunea 2:

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE 9 INREGISTRARI

**** INCEPUT FAZA ****

BENEFICIAR /CANTITATE LIVRATA / PRET

```
=====
```

PECO TITAN	1000	7500	
PECO MILITARI	30000	7500	
PECO-COLENTINA		1000	7100
PECO-DR.TABEREI		3000	8000
CRAIOVA-VALEA ROSIE		4000	6000
PLOIESTI-BARIERA		3000	7000
BUZAU-CENTRU	3000	7800	
IASI-COPOU	300	7890	
URZICENI	1000	7000	

**** TERMINAT FAZA ****

l)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?6

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?CRAIOVA-VALEA ROSIE

SPECIFICATI MODIFICARILE DORITE

NUME BENEFICIAR ?CRAIOVA

CANTITATE ?4000

PRET ?7800

NOUA INREGISTRARE :

BENEFICIAR /CANTITATE LIVRATA / PRET

```
=====
```

CRAIOVA	4000	7800	
---------	------	------	--

**** TERMINAT FAZA ****

m)

Fig. 10.5. l) optiunea 4; m) optiunea 6;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE 9 INREGISTRARI

**** INCEPUT FAZA ****

BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PECO TITAN	1000	7500	
PECO MILITARI	30000	7500	
PECO-COLENTINA		1000	7100
PECO-DR. TABEREI		3000	8000
ORAIOVA	4000	7800	
PLOIESTI-BARIERA		3000	7000
BUZAU-CENTRU	3000	7800	
IASI-COPOU	300	7890	
URZICENI	1000	7000	

**** TERMINAT FAZA ****

n)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?6

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PECO-COLENTINA

SPECIFICATI MODIFICARILE DORITE

NUME BENEFICIAR ?PECO-CV

CANTITATE ?6789

PRET ?7800

NOUA INREGISTRARE :

BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PECO-CV	6789	7800
---------	------	------

**** TERMINAT FAZA ****

o)

Fig. 10.5. n) optiunea 4; o) optiunea 6;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?6

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PLOIESTI-BARIERA

SPECIFICATI MODIFICARILE DORITE

NUME BENEFICIAR ?PLOIESTI

CANTITATE ?5655

PRET ?7600

NOUA INREGISTRARE :

BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PLOIESTI	5655	7600
----------	------	------

**** TERMINAT FAZA ****

INREGISTRARI 0000-0000 7600-0000 LISTARE NUMARARE
 1000-0000 7600-0000 1000-0000 7600-0000
 1000-0000 7600-0000 1000-0000 7600-0000

p)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE 9 INREGISTRARI

**** INCEPUT FAZA ****

BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PECO TITAN	1000	7500	
PECO MILITARI	30000	7500	
PECO-CV	6789	7800	
PECO-DR.TABEREI		3000	8000
CRAIOVA	4000	7800	
PLOIESTI	5655	7600	
BUZAU-CENTRU	3000	7800	
IASI-COPOU	300	7890	
URZICENI	1000	7000	

**** TERMINAT FAZA ****

r)

Fig. 10.5. p) optiunea 6; r) optiunea 4;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?5
 NUMELE BENEFICIARULUI : ?ICITPR-PLOIESTI

**** INCEPUT FAZA ****

BENEFICIAR INEXISTENT IN FISIER

**** TERMINAT FAZA ****

s)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?5
 NUMELE BENEFICIARULUI : ?PECO-DR.TABEREI

**** INCEPUT FAZA ****

A FOST STERS BENEFICIARUL PECO-DR.TABEREI

**** TERMINAT FAZA ****

t)

Fig. 10.5. s) opțiunea 5; t) opțiunea 5;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE 8 INREGISTRARI

**** INCEPUT FAZA ****

BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PECO TITAN	1000	7500
PECO MILITARI	30000	7500
PECO-CV	6789	7800
	0	0
CRAIOVA	4000	7800
PLOIESTI	5655	7600
BUZAU-CENTRU	3000	7800
IASI-COPOU	300	7890

**** TERMINAT FAZA ****

u)

- 1.- CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?SUCEAVA

CANTITATE ?5600

PRET ?7800

**** TERMINAT FAZA ****

v)

Fig. 10.5. u) optiunea 4; v) optiunea 2;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE 9 INREGISTRARI

**** INCEPUT FAZA ****

BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PECO TITAN	1000	7500
PECO MILITARI	30000	7500
PECO-CV	6789	7800
SUCEAVA	5600	7800
CRAIOVA	4000	7800
PLOIESTI	5655	7600
BUZAU-CENTRU	3000	7800
IASI-COPOU	300	7890
URZICENI	1000	7000

**** TERMINAT FAZA ****

z)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?TITU

CANTITATE ?5600

PRET ?7600

**** TERMINAT FAZA ****

w1)

Fig. 10.5. z) opțiunea 4; w.1) opțiunea 4;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?CERNAVODA
CANTITATE ?5400
PRET ?6000

**** TERMINAT FAZA ****

w2)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?ARAD
CANTITATE ?5555
PRET ?7780

**** TERMINAT FAZA ****

w3)

Fig. 10.5. w.2) opțiunea 2; w.3) opțiunea 2;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?ALBA-IULIA
CANTITATE ?500
PRET ?7914

**** TERMINAT FAZA ****

w4)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?2

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?VALENI
CANTITATE ?1000
PRET ?7400

**** TERMINAT FAZA ****

w5)

Fig. 10.5. w.4) opțiunea 2; w.5) opțiunea 2;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?4

FISIERUL CONTINE 14 INREGISTRARI

**** INCEPUT FAZA ****

BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PECO TITAN	1000	7500
PECO MILITARI	30000	7500
PECO-CV	6789	7800
SUCEAVA	5600	7800
CRAIOVA	4000	7800
PLOIESTI	5655	7600
BUZAU-CENTRU	3000	7800
IASI-COPOU	300	7890
URZICENI	1000	7000
TITU	5600	7600
CERNAVODA	5400	6000
ARAD	5555	7780
ALBA-IULIA	500	7914
VALENI	1000	7400

**** TERMINAT FAZA ****

w6)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?3

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PLOIESTI
BENEFICIAR /CANTITATE LIVRATA / PRET

=====

PLOIESTI	5655	7600
----------	------	------

**** TERMINAT FAZA ****

w7)

Fig. 10.5. w.6) optiunea 4; w.7) optiunea 3;

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?3

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?PRAHOVA
BENEFICIARUL PRAHOVA NU EXISTA IN FISIER

**** TERMINAT FAZA ****

w8)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?3

**** INCEPUT FAZA ****

NUMELE BENEFICIARULUI : ?COMBINATUL PETROCHIMIC VALEA CALUGAREASCA
BENEFICIARUL COMBINATUL PETROCHIMIC VALEA CALUGAREASCA
NU EXISTA IN FISIER

**** TERMINAT FAZA ****

w9)

1. CREARE FISIER BENEFICIARI
2. ADAUGARE BENEFICIARI
3. CONSULTARE FISIER BENEFICIARI
4. NUMARARE INREGISTRARI EXISTENTE SI LISTARE FISIER
5. STERGERE DIN FISIER BENEFICIARI
6. MODIFICARE INREGISTRARE BENEFICIAR
7. SFIRSIT SESIUNE ACTUALA

PRECIZATI OPTIUNEA DVS.?7
STOP AT LINE 1720

w10)

Fig. 10.5. w.8) opțiunea 3; w.9) opțiunea 3; w.10) opțiunea 7.

Joc pe COMODORE

```

10 REM JOCL BILELOR
20 PRINT CHR$(147)
30 LO=1024 : HI=2023 : WI=40 :
  D=54272
40 L=HI-LO : TK=81
50 INPUT "CITE BILE (PINA LA 9)"; N
60 DIM S (N, 10, 2), T(N), P(N), C(N)
70 FOR I=1 TO N : P(I)=INT
  (RND(1) * L)+LO
80 PRINT
90 PRINT "CITE MIȘCĂRI PENTRU
  BILA #"; I
100 INPUT T(I)
110 FOR J=1 TO T(I)
120 PRINT
130 PRINT "INTRODUCEȚI H ȘI V PEN-
  TRU BILĂ"; I; "MUTA M INTROD"; J
140 INPUT S(I, J, 1), S(I, J, 2)
150 NEXT J : NEXT I
160 PRINT CHR$(147)
170 FOR I=1 TO N : C(I)=C(I)+1
180 IF C(I)>T(I) THEN C(I)=1
190 NP=P(I)+S(I, C(I), 1)+WI *
  S(I, C(I), 2)
200 IF NP>HI THEN NP=NP-L
210 IF NP<LO THEN NP=NP+L
220 POKE NP, TK : POKE P(I), 32:
  POKE NP+D, 6+I
230 P(I)=NP
240 NEXT I
250 GOTO 170

```

TEMA 10

Răspundeți prin DA sau NU la următoarele întrebări:

- Lungimea unui element de masiv șir este cuprinsă între 2 și 512.
- Linia de program

150 DIM #2, XS (35)

declară masivul X\$ ca avînd elemente de lungime maximă 64.

- Elementul unui masiv șir nu poate depăși lungimea declarată prin DIM#.

- Linia de program

30 DIM #3, A\$(20)=4

declară masivul A\$ cu 20 elemente, fiecare element avînd lungimea maximă de 4 octeți.

Înlocuiți cuvintele care lipsesc din următoarele propoziții:

a) Linia de program

10 DIM #1, BS (10)=70

declară masivul _____ cu _____ elemente, fiecare element avînd lungimea maximă de _____ octeți.

b) Linia de program

30 DIM #7, N\$(60)

declară masivul _____ cu _____ elemente, fiecare element avînd lungimea maximă de _____ octeți.

c) Linia de program

10 DIM #6, US (30)=8

declară masivul cu elemente, fiecare element avînd lungimea maximă de octeți.

d) Linia de program

10 DIM #1, C\$ (?)

declară masivul C\$ cu elemente, fiecare element avînd lungimea maximă de 16 octeți.

Scrieți cite un program BASIC pentru fiecare din problemele de mai jos:

a) Se consideră fișierul masiv virtual ACHIZIȚII care conține pentru fiecare articol utilizat în activitatea de întreținere a utilajelor dintr-o întreprindere constructoare de mașini, denumirea sa și prețurile de achiziție din țară, est și vest. Să se creeze și să se întrețină acest fișier utilizînd subrutinele descrise în EXEMPLUL 10-M.

b) Se consideră fișierul masiv virtual PROIECTE care conține pentru fiecare articol (proiect) codul, denumirea și caracteristicile tehnice ale produsului proiectat. Să se creeze și să se întrețină acest fișier utilizînd subrutinele de adăugare, consultare, numărare, ștergere și modificare prezentate în Conversația 10.

c) În programul de mai jos (EXEMPLUL 10-M, versiunea 1) s-au strecurat mai multe greșeli de logică. Identificați aceste erori și realizați o versiune corectă a programului.

```

100 REM PROGRAMUL CREAȚĂ ȘI ÎNTREȚINE UN FIȘIER DE BENEFICIARI
110 REM FUNCȚIUNILE PROGRAMULUI
120 REM - CREARE FIȘIER BENEFICIARI
130 REM - ADĂUGARE BENEFICIARI
140 REM - CONSULTARE FIȘIER BENEFICIARI
150 REM - ȘTERGERE BENEFICIARI
160 REM - LISTARE BENEFICIARI
170 REM - MODIFICARE BENEFICIAR EXISTENT
180 REM DEFINIREA FIȘIERULUI VIRTUAL
190 DIM #1, B$(30)=40, C(30), P(30), B1$(30, 30)
200 REM B$ - NUME BENEFICIAR
210 REM C - CANTITATE
220 REM P - PREȚ
230 PRINT
240 PRINT "1. CREARE FIȘIER BENEFICIARI"
250 PRINT "2. ADĂUGARE BENEFICIARI"
260 PRINT "3. CONSULTARE FIȘIER BENEFICIARI"
270 PRINT "4. NUMĂRARE ÎNREGISTRĂRI EXISTENTE ȘI LISTARE FIȘIER"
280 PRINT "5. ȘTERGERE DIN FIȘIER BENEFICIARI"
290 PRINT "6. MODIFICARE ÎNREGISTRARE BENEFICIAR"
300 PRINT "7. SFIRȘIT SESIUNE ACTUALĂ"
310 PRINT
320 INPUT "PRECIZAȚI OPȚIUNEA DVS."; 0
330 IF 0<8 THEN 350
340 PRINT "OPȚIUNE ERONATA" : GO TO 230
350 IF 0=1 THEN GO SUB 440

```

```

360 IF 0=2 THEN GO SUB 540
370 IF 0=3 THEN GO SUB 760
380 IF 0=4 THEN GO SUB 1260
390 IF 0=5 THEN GO SUB 900
400 IF 0=6 THEN GO SUB 1030
410 IF 0=7 THEN GO TO 1630
420 GO TO 230
430 PRINT
440 REM ++++++ CREARE FIȘIER BENEFICIARI
450 IF T1 <> 0 THEN 510
460 GO SUB 1380 ! DESCHIDERE
470 FOR I=1 TO 30
480 B$(I)=" " : P(I)=0 : C(I)=0
490 NEXT I
500 T1=T1+1 : GO TO 340
510 PRINT "FIȘIERUL EXISTĂ" : GOTO 350
520 GOSUB 1440
530 RETURN
540 REM ++++++ ADAUGARE BENEFICIARI NOI
550 T1=T1+1
560 GO SUB 1540 ! NUMĂRARE ARTICOLE
570 GO SUB 1380 ! DESCHIDERE
580 IF I3=30 THEN 680
590 INPUT "NUMELE BENEFICIARULUI : " ; N$
600 T=0 : A=0
610 FOR I=30 TO 1 STEP -1
620 IF C(I)=0 THEN A=I ! REȚIN PRIMA CĂSUȚĂ GOALĂ
630 IF N$=B$(I) THEN T=1 BENEFICIAR EXISTENT ÎN FIȘIER
640 PRINT "A=" ; A, "I=" ; I ; B$(I) ; TAB(60) ; N$ ; C(I) ; P(I)
650 NEXT I
660 IF T=1 THEN 730
670 IF A<31 THEN 690
680 PRINT " * FIȘIERUL ESTE PLIN * ADAUGĂRILE SE REFUZĂ * " : GO TO 520
690 B$(A)=N$ : I3=I3+1 ! REȚIN NUMELE BENEFICIARULUI
700 INPUT "CANTITATE" , C(A) ! REȚIN CANTITATEA
710 INPUT "PREȚ" , P(A) ! REȚIN PREȚUL
720 GO TO 740
730 PRINT "BENEFICIARUL" ; N$ ; "ESTE DEJA PREZENT ÎN FIȘIER"
740 GO SUB 1440 ! ÎNCHIDERE
750 RETURN
760 REM ++++++ CONSULTARE FIȘIER BENEFICIARI
770 I1=T1+1
780 GO SUB 1380 ! DESCHIDERE
790 T=0
800 INPUT "NUMELE BENEFICIARULUI : " , N$
810 FOR I=1 TO 30
820 IF N$=B$(I) THEN T=I ! DETECTARE PREZENTĂ BENEFICIAR SOLICITAT
830 NEXT I
840 IF T <> 0 THEN 860
850 PRINT "BENEFICIARUL" ; N$ ; "NU EXISTĂ ÎN FIȘIER" : GO TO 880
860 GO SUB 1500
870 PRINT B$(T) ; TAB(21) ; C(T) ; TAB(31) ; P(T)
880 GO SUB 1440 ! ÎNCHIDERE
890 RETURN
900 REM ++++++ ȘTERGERE BENEFICIAR DIN FIȘIER
910 T=0 : T1=T1+1
920 INPUT "NUMELE BENEFICIARULUI : " , N$
930 GO SUB 1380 ! DESCHIDERE
940 FOR I=1 TO 30
950 IF N$ <> B$(I) THEN 970
960 B$(I)=" " : C(I)=0 : P(I)=0 : T=1
970 NEXT I
980 IF T=1 THEN 1000

```

```

990 PRINT "BENEFICIAR INEXISTENT IN FIȘIER" : GO TO 638
1000 PRINT "A FOST ȘTERS BENEFICIARUL"; N$
1010 GO SUB 1440 ! INCHIDERE
1020 RETURN
1030 REM ++++++ MODIFICARE INREGISTRARE BENEFICIAR
1040 GO SUB 1380 ! DESCHIDERE
1050 T=O : T1=T1+1
1060 INPUT "NUMELE BENEFICIARULUI : ", N$
1070 FOR I=1 TO 30
1080 IF N$(I) THEN T=I ! DETECTARE PREZENTĂ BENEFICIAR SOLICITAT
1090 NEXT I
1100 IF T <> O THEN 1140
1110 PRINT "BENEFICIARUL"; N$; "NU EXISTĂ IN FIȘIER" : GO TO 1230
1120 GO SUB 1500
1130 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
1140 PRINT "SPECIFICAȚI MODIFICĂRILE DORITE"
1150 INPUT "NUME BENEFICIAR"; M$
1160 INPUT "CANTITATE", C1
1170 INPUT "PREȚ", P1
1180 IF M$ <> " " THEN B$(T)=M$
1190 OPEN "BENEF. DAT" AS VIRTUAL FILE 1
1200 IF P1 <> O THEN P(T)=P1
1210 PRINT "NOUA ÎNREGISTRARE : "
1220 GO SUB 1500
1230 PRINT B$(T); TAB(21); C(T); TAB(31); P(T)
1240 GO SUB 1440 ! INCHIDERE
1250 RETURN
1260 REM SRT DE NUMĂRARE A ÎNREG. EXISTENTE IN FIȘIER
1270 T1=T1+1
1280 GO SUB 1540 ! NUMĂRARE
1290 PRINT : PRINT
1300 PRINT "FIȘIERUL CONȚINE"; I3; "ÎNREGISTRĂRI"
1310 GO SUB 1380
1320 GO SUB 1500
1330 PRINT B$(P9), C(P9), P(P9) FOR P9=1 TO I3
1340 PRINT
1350 GO SUB 1440
1360 RETURN
1370 REM SUBRUTINE
1380 REM DESCHIDERE FIȘIER
1390 OPEN "BENEF.DAT" AS VIRTUAL FILE 1
1400 PRINT
1410 PRINT "**** INCEPUT FAZA          ****"
1420 PRINT
1430 RETURN
1440 REM INCHIDERE FIȘIER
1450 CLOSE 1
1460 PRINT
1470 PRINT "**** TERMINAT FAZA          ****"
1480 PRINT
1490 RETURN
1500 REM CAP TABEL
1510 PRINT "BENEFICIAR/CANTITATE LIVRATĂ/PREȚ"
1520 PRINT "===== " : PRINT
1530 RETURN
1540 REM SUBRUTINA NUMĂRARE BENEFICIARI ÎNSCRIȘI IN FIȘIER
1550 OPEN "BENEF. DAT" AS VIRTUAL FILE 1
1560 I3=0
1570 FOR I=1 TO 30
1580 IF B$(I)=" " THEN 1600
1590 I3=I3+1
1600 NEXT I

```

```
1610 CLOSE 1
1620 RETURN
1630 STOP
1640 END
```

Să se proiecteze și să se realizeze un program BASIC-PLUS ghidat de un meniu pentru crearea și întreținerea fișierului masiv virtual JUCĂRII .DAT avind structura modificată în: cod articol, denumire articol, a1, a2, a3, b1, b2, b3, c1, c2, c3, d1, d2, d3, preț.

Să se proiecteze și să se realizeze un program BASIC-PLUS ghidat de un meniu pentru crearea și întreținerea fișierului masiv virtual CONSUMURI .DAT avind structura modificată în: cod articol, denumire articol, cod proveniență (R.S.R./IMPORT), consum normat, consum realizat, preț de achiziție.

Indicație. Se vor utiliza subrutinele din EXEMPLUL 10-M.

CONVERSAȚIA 11

Proiectarea și realizarea unui program BASIC pentru sortarea unui fișier secvențial și a unui fișier masiv virtual de livrări prin metodele: extracție, inserție, SHELL, indexată. Funcții standard pe șiruri de caractere. Instrucțiunea ON...GOSUB. Funcția SWAP. JOCURI, APLICAȚII și TESTE pentru cititor



EXEMPLELE 11-PC, m, M, F

□ De la problemă la program

Vom relua problema din conversațiile 7, 8 și 10 complicind-o sub mai multe aspecte.

Astfel, în EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F vom trata problema creerii și întreținerii fișierului secvențial de livrări cu structură modificată: denumire-beneficiar, cantitate-livrată, preț. Fișierul va fi creat în cadrul programului principal, iar setul de date privind livrările de benzină va fi supus, pe lângă operațiile de adăugare (beneficiari), listare (beneficiari), consultare (beneficiari), salvare (fișier), încărcare (fișier) și operației de **sortare**.

Sortarea este o operație foarte des întâlnită în cadrul programelor de gestiune economică, deoarece majoritatea rapoartelor presupun ca anumite informații să fie într-un anume fel aranjate, ceea ce înseamnă ca valorile diferitelor cîmpuri (numite chei) ale unui articol să fi fost deja ordonate, crescător sau descrescător.

În cadrul acestor programe (EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F) referirea la un anume beneficiar se va face nu prin cod ci prin numele beneficiarului.

EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F, puțin mai complicate decît precedentele, vor fi prevăzute cu un meniu care să afișeze pe ecran funcțiunile programului:

- 1) Adăugare beneficiari
- 2) Listare beneficiari
- 3) Consultare beneficiari
- 4) Sortare date
- 5) Salvare fișier beneficiari
- 6) Încărcare fișier beneficiari.

Sortarea (opțiunea 4), ghidată și ea de un meniu, va fi realizată prin una din metodele: BUBBLE SORT, extracție, inserție, sortare indexată, SHELL, QUICK SORT.

Pentru fiecare din opțiunile programului principal și al subprogrameelor de sortare se va scrie cîte o subrutină BASIC ce va fi prevăzută cu mesaje corespunzătoare.

Utilizatorul va tasta o cifră de la 1 la 6 pentru meniul principal și o cifră de la 1 la 8 pentru submeniul de sortare fără a mai fi obligat să acționeze tasta **[CR]**. Pentru oprirea execuției programului se va tasta cifra zero. Cit privește reîntoarcerea în meniul principal și în submeniul subrutinei de sortare, aceasta se va realiza prin acționarea tastei **[CR]**.

EXEMPLUL 11-M implementat pe minicalculatoarele INDEPENDENT și CORAL va fi în întregime consacrat sortării fișierului masiv virtual BENEF.DAT creat cu EXEMPLUL 10-M (v. Conversația 10).

Programul, pilotat de un meniu, folosește subrutinele de sortare din EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F adaptate însă pentru limbajul BASIC-PLUS.

Listele cu subrutinele de sortare vi se vor prezenta în cadrul specificațiilor de programare urmînd ca dvs. singuri, fără ajutorul nostru, să înțelegeți mecanismul de proiectare și realizare a acestora.

Rămîne să vă concentrați atenția în special asupra proiectării și realizării programelor principale ale aplicației: EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F și EXEMPLUL 11-M.

Vă invităm să testați fiecare subrutină de sortare în parte (cu **DATA** și **READ**) înainte de a o include în programul principal. De reținut că două (!) din cele opt subrutine scrise pentru EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F conțin cite o... mică eroare de logică. Pentru depanarea acestora vă recomandăm să consultați subrutinele scrise (fără erori) în BASIC-PLUS pentru EXEMPLUL 11-M. După cum remarcați, de această dată am fost mai puțin generoși! Vă rugăm să nu ne-o luați în nume de rău.

□ Specificații de programare

Schema de sistem, structura înregistrării fișierului BENE.F.DAT, tabela de variabile, specificațiile de programare și sursele subrutinelor (BASIC-80) de sortare sint prezentate în modulul de analiză structurată, figura 11.1.

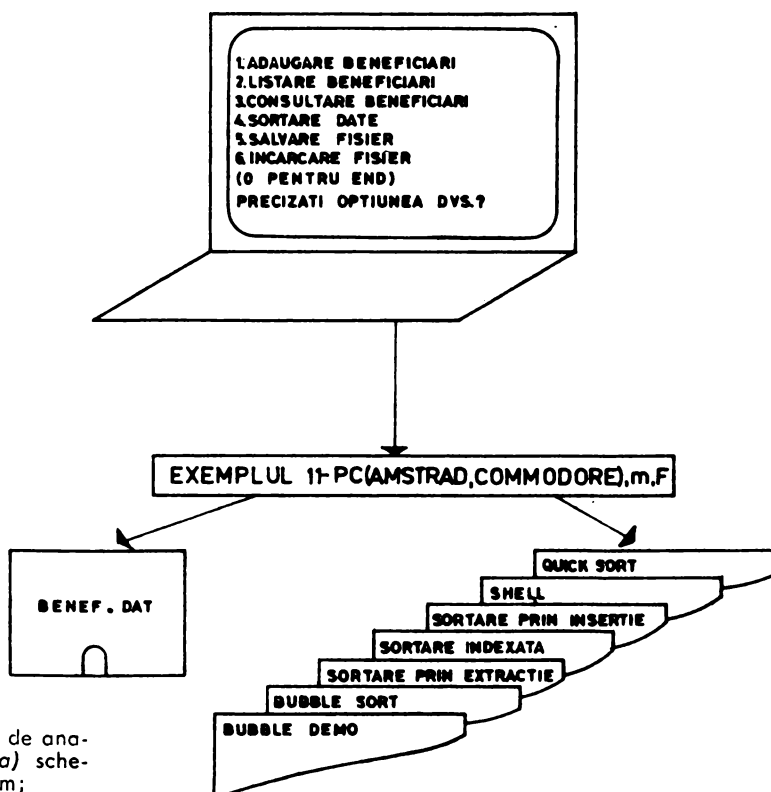


Fig. 11.1. Modulul de analiză structurată: a) schema de sistem;

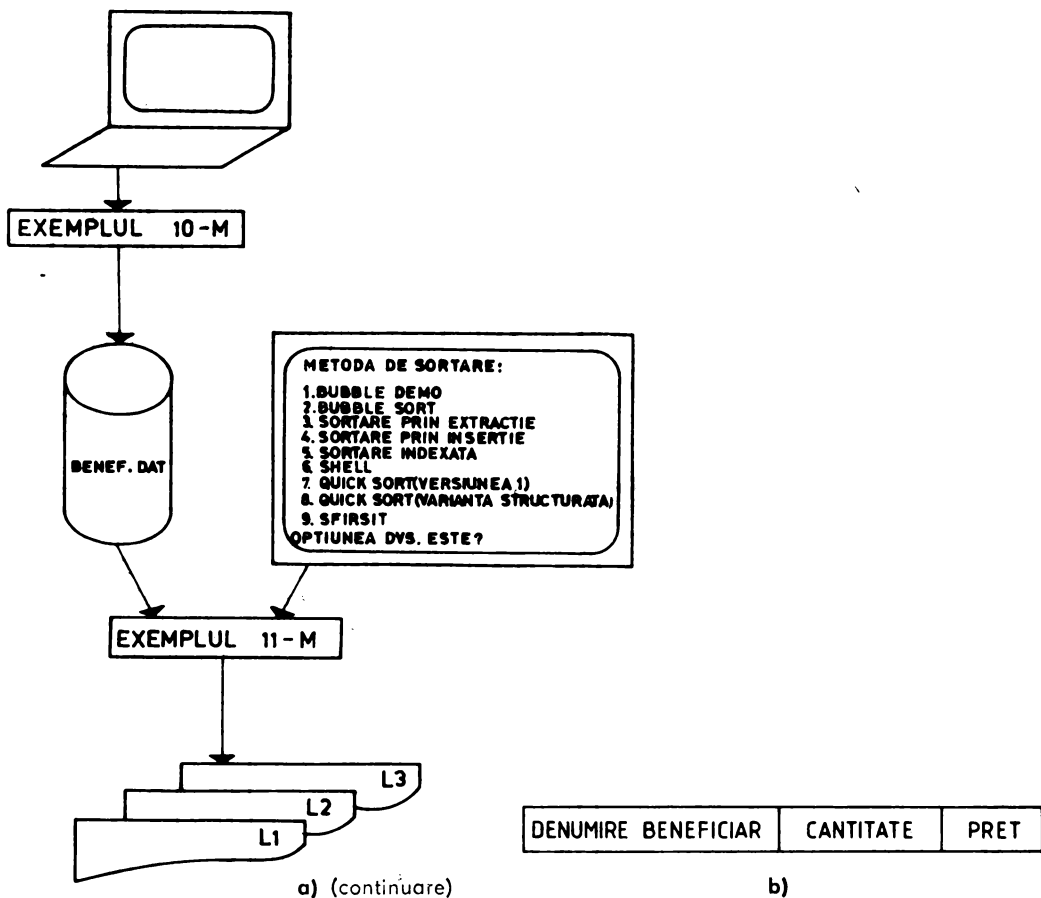


TABELA DE VARIABLE

Nume program: EXEMPLUL 11-M

Variabile de intrare

Variabile de stare

Variabile de ieșire

B\$: nume beneficiar
 C : cantitate benzină
 P : preț tonă benzină
 M : opțiune

I3 : număr beneficiari
 B, I, F, N, L\$, L, L1, J, K, S5,
 D5, D, U\$, N\$, N1, R, A\$, X, P,
 M, K\$, K, Z, G, I1, J1, S, S9 :
 variabile de lucru
 M7S : maxim
 S : suma

B\$: nume beneficiar
 C : cantitate
 P : preț

c)

Fig. 11.1. b) structura înregistrării fișierului; c) tabela de variabile pentru programul BASIC-PLUS;

TABELA DE VARIABILE

Nume program : EXEMPLELE 11-PC (AMSTRAD, COMMODORE), m, F

Variabile de intrare	Variabile de stare	Variabile de ieșire
BENS : nume beneficiar CANT : cantitate benzină PREȚ : preț tonă benzină NBENS : nume beneficiar căutat OPT : opțiune MET : metoda de sortare A\$: continuare/oprire execuție program	III, I, N, B, F, C, D D1, E, M, MINS, J, K STÎNGA, DREAPȚA, US, N1\$, R1, R2, K\$, N\$, S, P, X, G, C1, S1, I1, J1, S9 : variabile de lucru	BENS : nume beneficiar CANT : cantitate PREȚ : preț

d)

SPECIFICAȚII DE PROGRAMARE

Nume program : EXEMPLELELE 11 – PC (AMSTRAD, COMMODORE), m, F

Descriere program

Programul creează, întreține și sortează un fișier de livrări de organizare secvențială.

Intrări

Se citește de la tastatură: nume beneficiar, cantitate benzină livrată, preț tonă benzină, opțiune meniu și metoda de sortare.

Ieșiri

Listă beneficiari sortată.

Lista de funcțiuni ale programului

1. Adăugare beneficiari
2. Listare beneficiari
3. Consultare beneficiari

4. Sortare date
5. Salvare fișier beneficiari
6. Încărcare fișier beneficiari
7. Stop
8. Ștergere ecran
9. Citire opțiune meniu
10. Citire opțiune sortare
11. Sortare BUBBLE DEMO
12. Sortare BUBBLE
13. Sortare prin extracție
14. Sortare prin inserție
15. Sortare indexată
16. Sortare SHELL
17. QUICK SORT (versiunea 1)
18. QUICK SORT (variantă structurată)
19. Afișare meniu principal
20. Afișare submeniu sortare

e)

SPECIFICAȚII DE PROGRAMARE

Nume program : EXEMPLUL 11 – M

Descriere program

Programul sortează (prin opt metode) un fișier masiv virtual de livrări.

Intrări

Fișierul masiv virtual BENE.F. DAT

Ieșiri

Listă beneficiari sortată

Lista de funcțiuni ale programului

1. Numărare beneficiari
2. Deschidere-închidere fișier
3. Afișare meniu sortare

4. Citire opțiune sortare
5. Afișare listă nesortată
6. Sortare BUBBLE DEMO
7. Sortare BUBBLE
8. Sortare prin extracție
9. Sortare prin inserție
10. Sortare indexată
11. Sortare SHELL
12. QUICK SORT (versiunea 1)
13. QUICK SORT (variantă structurată)
14. Afișare listă sortată
15. Stop

f)

Fig. 11.1. d) tabela de variabile pentru programele BASIC-AMSTRAD, COMMODORE, BASIC-80, ABASIC; e) specificații de programare pentru programele BASIC-AMSTRAD, COMMODORE, BASIC-80, ABASIC; f) specificații de programare pentru programul BASIC-PLUS;

Subrutina BUBBLE DEMO (890)

```

890 REM "BUBBLE DEMO"
900 N=III
910 PRINT "Lista beneficiari (nesortată)"
920 FOR I=1 TO N
930   PRINT BENS(I), CANT(I), PREȚ(I)
940 NEXT I
950 PRINT : PRINT
960 FOR B=1 TO N-1
970   PRINT "Pasul --> "; B
980   F=0
990   FOR C=1 TO N-B
1000    FOR D=1 TO N
1010     BS(C,D)=BENS(D)
1020    NEXT D
1030    IF BENS(C+1) >= BENS(C) THEN 1080
1040     SWAP BENS(C), BENS(C+1)
1050     SWAP CANT(C), CANT(C+1)
1060     SWAP PREȚ(C), PREȚ(C+1)
1070     F=1
1080    NEXT C
1090    FOR D=1 TO N
1100     BS(N-B+1,D)=BENS(D)
1110    NEXT D
1120   FOR D=1 TO N
1130    IF D=1 THEN PRINT " * "; : D1=-1 : GOTO 1210
1140    D1=D-2
1150    IF D1 >= N-B THEN D1=N-B+1
1160    IF D1=0 THEN 1180
1170    FOR E=1 TO D1 : PRINT TAB (E * 6-3); BS(E,D); : NEXT E
1180    IF D > N-B+1 THEN 1220
1190    PRINT TAB (6 * D1+2); " * "; BS (D1+2,D); :
1200    PRINT TAB (6 * D1+8); " * ";
1210   FOR E=D1+2 TO N-B+1 : PRINT TAB (E * 6-3); BS(E,D); : NEXT E
1220   IF D < > N-B+1 THEN 1250
1230   PRINT
1240   FOR E=1 TO 6 * (N-B+1) : PRINT " - "; : NEXT E
1250   PRINT
1260   NEXT D
1270   IF F=0 THEN PRINT : PRINT "===Terminat===" : GOTO 1290
1280   PRINT : NEXT B
1290   PRINT : PRINT "Lista beneficiari (sortată)"
1300   FOR I=1 TO N
1310    PRINT BENS(I), CANT(I), PREȚ(I)
1320   NEXT I
1330   RETURN

```

g)

Fig. 11.1. g) subrutina 890;

Subrutina BUBBLE SORT (1550)

```

1550 REM "Sortare BUBBLE SORT"
1560 I=III
1570 REM
1580 FOR M=1 TO I-1
1590   F=0
1600   FOR N=1 TO I-M
1610     IF BENS(N+1) >= BENS(N) THEN 1660
1620     SWAP BENS(N), BENS(N+1)
1630     SWAP CANT(N), CANT(N+1)
1640     SWAP PREȚ(N), PREȚ(N+1)
1650     F=1
1660   NEXT N
1670   IF F=0 THEN 1690
1680 NEXT M
1690 PRINT : PRINT "Fișier beneficiari (sortat)"
1700 FOR B=1 TO I : PRINT BENS(B), CANT(B), PREȚ(B) : NEXT B
1710 PRINT
1720 RETURN

```

h)

Subrutina de sortare prin extracție (1730)

```

1730 REM "Sortare prin extracție"
1740 I=III
1750 FOR M=1 TO I-1
1760   REM "Se consideră că BENS(M) este minimul"
1770   MINS=BENS(M)
1780   J=M
1790   REM
1800   REM
1810   FOR K=M+1 TO I
1820     IF BENS(K) >= MINS THEN 1850
1830     MINS=BENS(K)
1840     J=M
1850   NEXT K
1860   REM
1870   SWAP BENS(M), BENS(J)
1880   SWAP CANT(M), CANT(J)
1890   SWAP PREȚ(M), PREȚ(J)
1900 NEXT M
1910 PRINT "Lista sortată : "
1920 FOR B=1 TO I : PRINT BENS(B), CANT(B), PREȚ(B) : NEXT B
1930 RETURN

```

i)

Fig. 11.1. h) subrutina 1 550; i) subrutina 1 730;

Subrutina de sortare prin inserție (1940)

```

1940 REM "Sortare prin inserție"
1950 STINGA=1 : DREAPTA=III
1960 I=III
1970 FOR M=STINGA+1 TO DREAPTA
1980   REM "Inserție"
1990   GOSUB 2050
2000 NEXT M
2010 PRINT "Lista sortată : "
2020 FOR B=1 TO I : PRINT BENS(B), CANT(B), PREȚ(B) : NEXT B
2030 RETURN
2040 REM "Procedura de inserție"
2050 PRINT
2060 D=M-1
2070 U$=BENS(M)
2080 C=CANT(M)
2090 P=PREȚ(M)
2100 IF D < STINGA OR U$ >= BENS(D) THEN 2160
2110 BENS(D+1)=BENS(D)
2120 CANT(D+1)=CANT(D) : PREȚ(D+1)=PREȚ(D)
2130 D=D-1
2140 GOTO 2100
2150 CANT(D+1)=C : PREȚ(D+1)=P
2160 BENS(D+1)=U$
2170 RETURN

```

j)

Subrutina SHELL

```

2520 REM "Sortare prin metoda SHELL"
2530 I=III : C=0 : S=0
2540 PRINT "Lista nesortată"
2550 GOSUB 2620
2560 REM "Sortare lista"
2570 GOSUB 2690
2580 PRINT "Lista sortată"
2590 GOSUB 2620
2600 PRINT C; "Comparații" : PRINT S; "Schimbări"
2610 RETURN
2620 REM "Tipărire listă nesortată"
2630 FOR K=1 TO I
2640   PRINT BENS(K); TAB(10); CANT(K); TAB(20); PREȚ(K)
2670 NEXT K
2680 RETURN
2690 REM "Metoda de sortare SHELL"
2700 G=I
2710 WHILE G > 1
2720   G=INT (G/2)
2730   M=I-G
2740   F=0
2750   FOR K=1 TO M
2760     P=K+G
2770     C=C+1
2780     IF BENS(K) <= BENS(P) THEN 2830
2790     S=S+1
2800     SWAP BENS(K), BENS(P)
2810     SWAP CANT(K), CANT(P) : SWAP PREȚ(K), PREȚ(P)
2820     F=1
2830   NEXT K
2840   IF F > 0 THEN 2740
2850 WEND
2860 RETURN

```

k)

Subrutina QUICK SORT (versiunea 1)

```

2870 REM "QUICK SORT (versiunea 1)"
2880 N=III : PRINT "Listă nesortată"
2890 I=N : GOSUB 2620
2900 C1=0 : S1=0
2910 I1=1 : J1=N
2920 I=I1 : J=J1 : S=-1
2930 C1=C1+1
2940 IF BENS(I) <= BENS(J) THEN 2980
2950 S1=S1+1
2960 SWAP BENS(I), BENS(J) : SWAP CANT(I), CANT(J) : SWAP PREȚ(I), PREȚ(J)
2970 S=SGN(-S)
2980 IF S=1 THEN I=I+1 ELSE J=J-1
2990 IF I < J THEN 2930
3000 IF I+1 >= J1 THEN 3020
3010 P=P+1 : S9(P,1)=I+1 : S9(P,2)=J1
3020 J1=I-1
3030 IF I1 < J1 THEN 2920
3040 IF P=0 THEN 3070
3050 I1=S9(P,1) : J1=S9(P,2) : P=P-1
3060 GOTO 2920
3070 PRINT : PRINT "Lista sortată"
3080 I=N : GOSUB 2620
3090 PRINT : PRINT N; "beneficiari"
3100 PRINT C1; "comparații"
3110 PRINT S1; "schimbări"
3120 PRINT
3130 RETURN

```

l)

Subrutina QUICK SORT (varianta structurată)

```

3200 REM "QUICK SORT (varianta structurată)"
3210 PRINT "Listă nesortată"
3215 I=III : GOSUB 2620
3220 N=III
3230 P=1 : S9(1,1)=1 : S9(1,2)=N
3240 WHILE P > 0
3250 I1=S9(P,1) : J1=S9(P,2) : P=P-1
3260 WHILE I1 < J1
3270 I=I1 : J=J1 : S90=-1
3280 WHILE I < J
3290 IF BENS(I) < BENS(J) THEN 3340
3300 SWAP BENS(I), BENS(J)
3310 SWAP CANT(I), CANT(J)
3320 SWAP PREȚ(I), PREȚ(J)
3330 S90=-S90+1 :
3340 IF NOT S90 THEN J=J-1 ELSE I=I+1
3350 WEND
3360 IF I+1 >= J1 THEN 3400
3370 P=P+1
3380 S9(P,1)=I+1
3390 S9(P,2)=J1
3400 J1=I-1
3410 WEND
3420 WEND
3430 PRINT
3440 PRINT "Listă sortată"
3450 I=N : GOSUB 2620
3460 RETURN

```

m)

Fig. 11.11. l) subrutina 2 870; m) subrutina 3 200;

Subrutina de sortare indexată

```

2180 REM "Sortare indexată"
2190 N=III
2200 FOR R=1 TO N
2210   N$(R,1)=BENS(R)
2220   R1=CANT(R) : N$(R,2)=STR$(R1)
2230   R2=PREȚ(R) : N$(R,3)=STR$(R2)
2240 NEXT R
2250 INPUT "Pentru continuare tasteți (Y/N)"; A$
2260 WHILE ASC(A$)=89
2270   INPUT "Cîmpul : 1. Beneficiar/2. Cantitate/3. Preț (1/2/3)"; J
2280   FOR R=1 TO N
2290     K$(R)=N$(R,J)
2300   NEXT R
2310   GOSUB 2430
2320   PRINT "Înregistrări sortate după cîmpul"; J
2330   FOR R=1 TO N
2340     FOR I=1 TO 3
2350       R1=X(R) : PRINT N$(R1,I); " ",
2360     NEXT I
2370   PRINT
2380 NEXT R
2390 PRINT
2400 INPUT "Pentru continuare tasteți (Y/N)"; A$
2410 WEND
2420 RETURN
2430 FOR A=1 TO N
2440   P=1
2450   FOR B=1 TO N
2460     IF K$(A)>K$(B) THEN P=P+1
2470     IF K$(A)=K$(B) AND A>B THEN P=P+1
2480   NEXT B
2490   X(P)=A
2500 NEXT A
2510 RETURN

```

n)

Fig. 11.1. n) subrutina 2180.

 Documentația de proiectare

Diagrama de structură și schemă logică sînt ilustrate în modulul de proiectare structurată, figura 11.2.

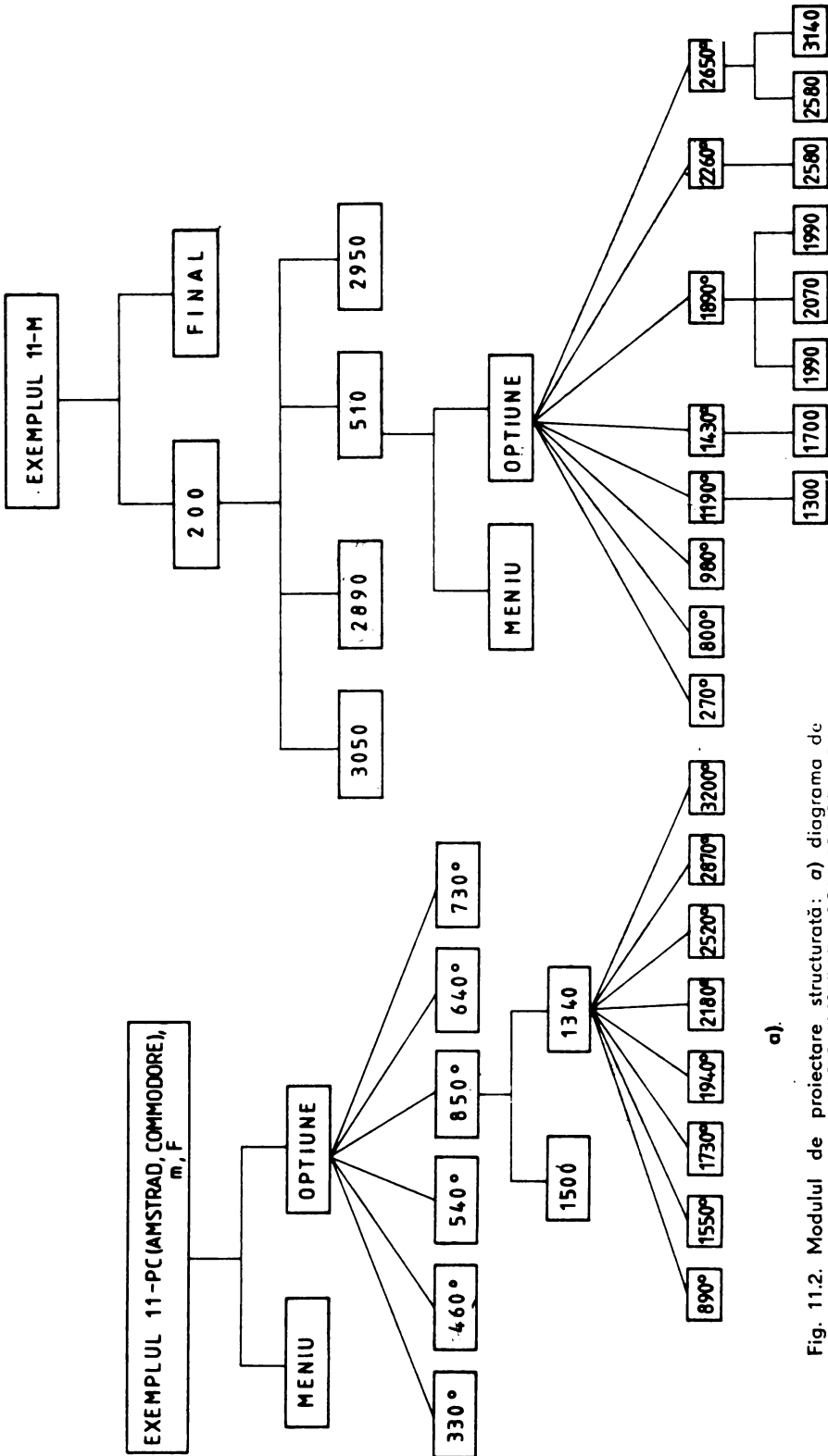
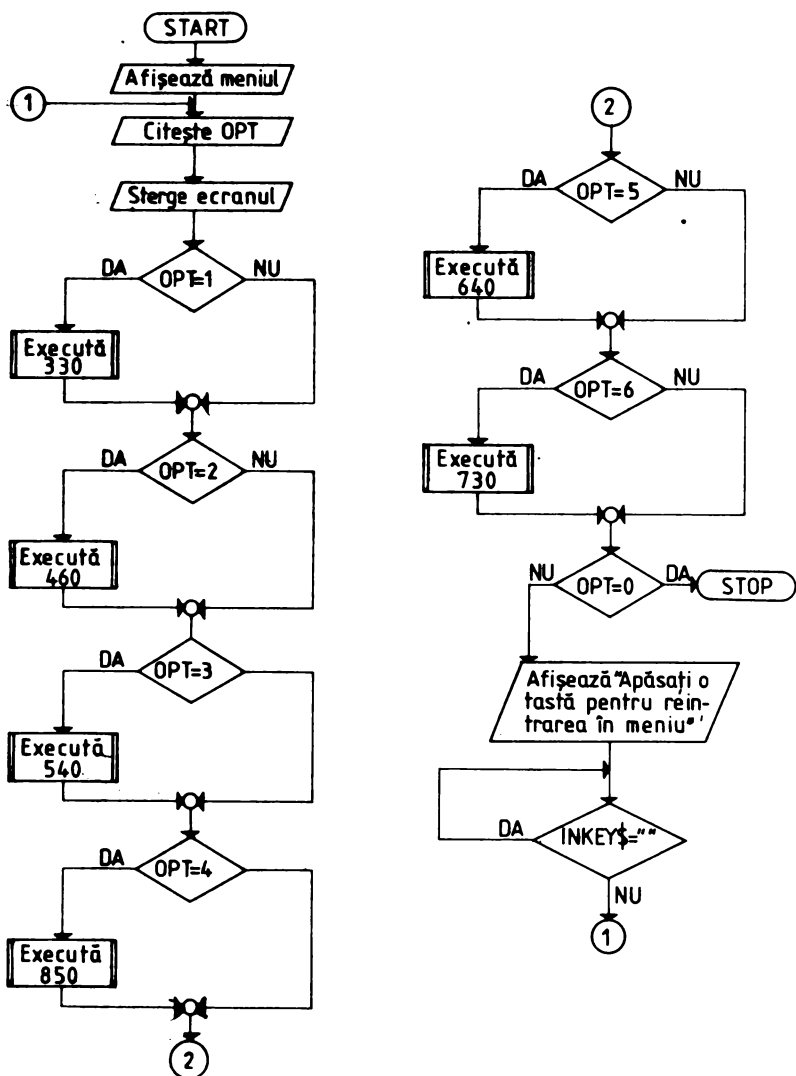


Fig. 11.2. Modulul de proiectare structurată: a) diagrama de structură a programelor BASIC-AMSTRAD, COMMODORE, BA-SIC-80, ABASIC; b) diagrama de structură a programului BA-SIC-PLUS;



c)

Fig. 11.2. c) schema logică a programelor BASIC-AMSTRAD, COMMODORE, BASIC-80, ABASIC;

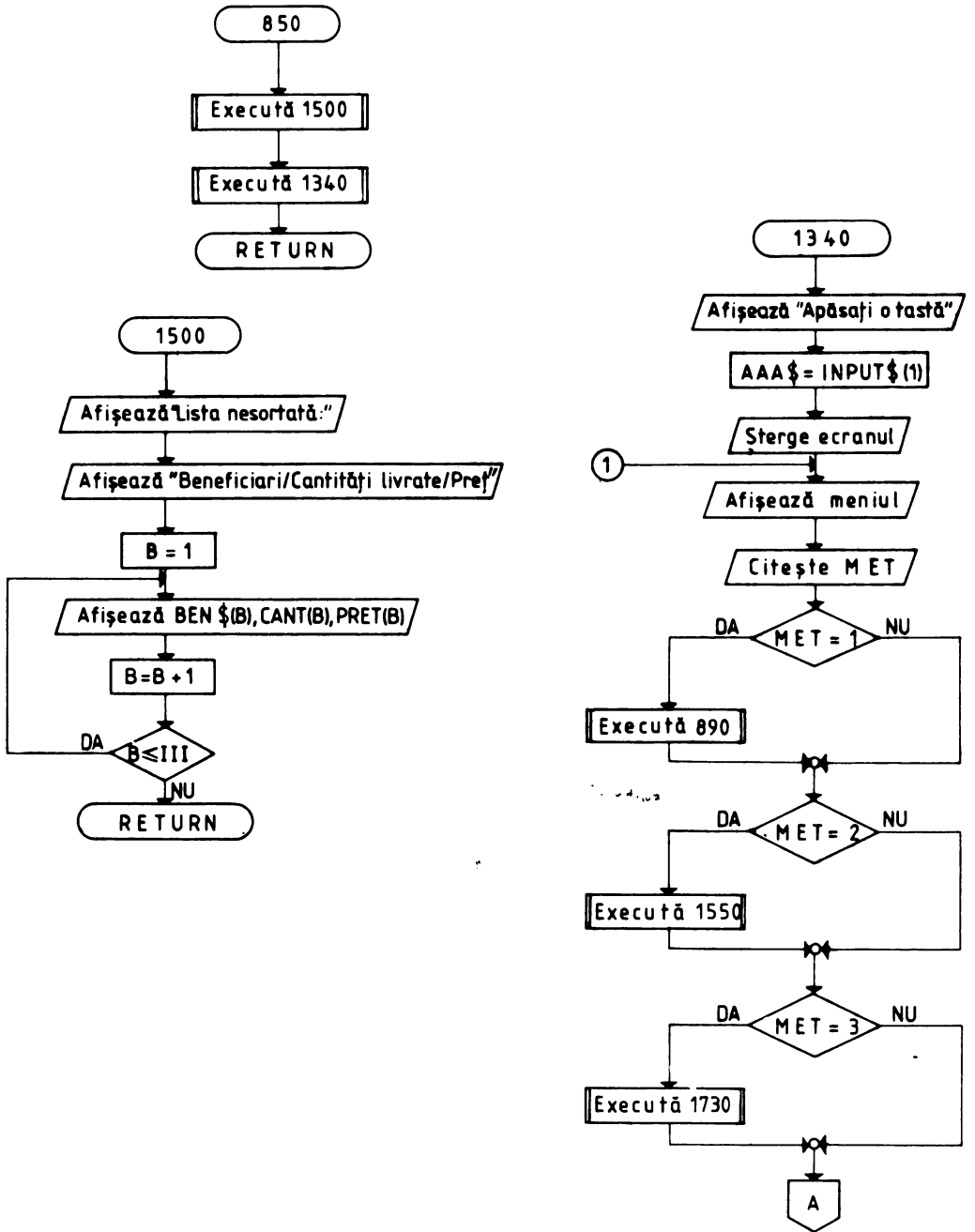


Fig. 11.2. c)

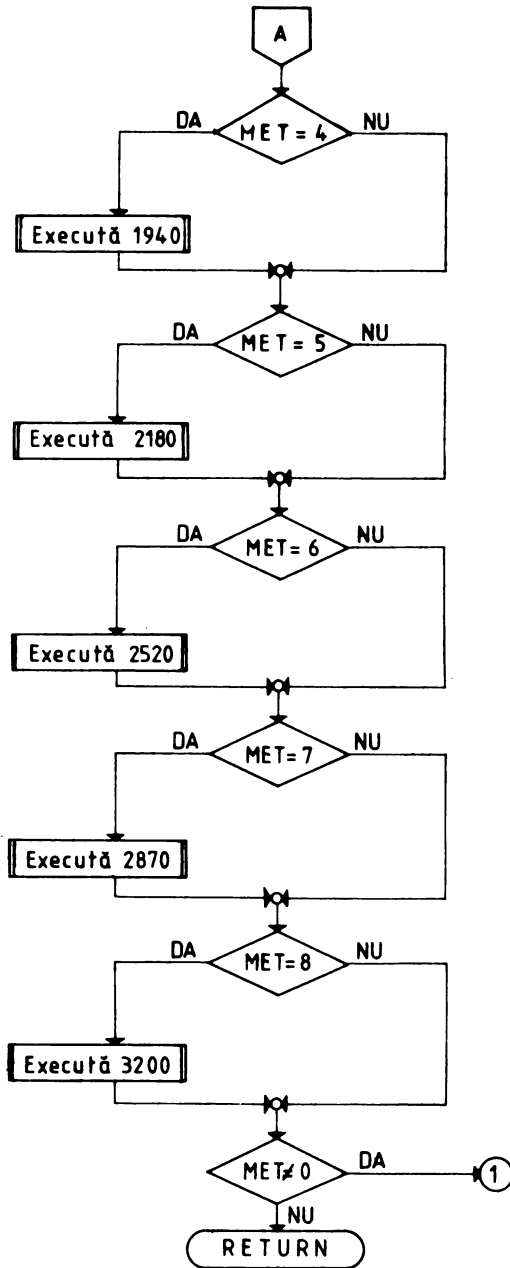
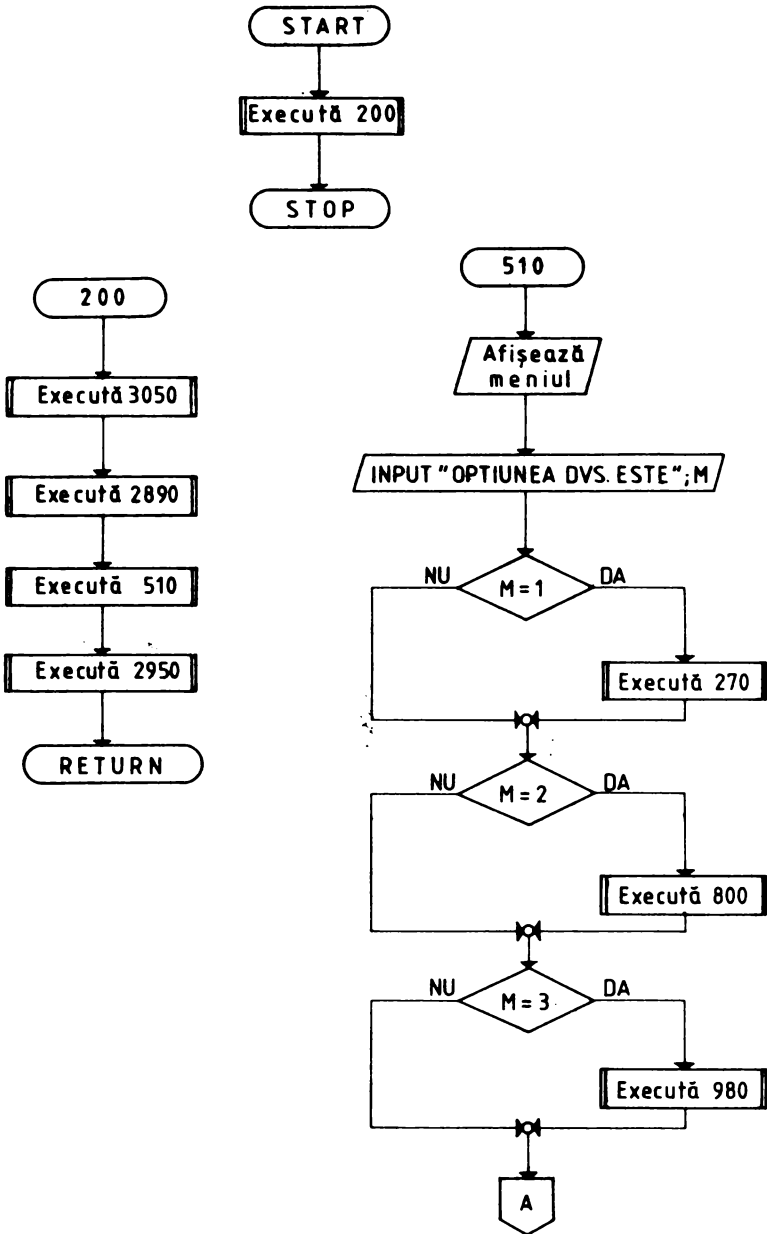


Fig. 11.2. c)



d)

Fig. 11.2. d) schema logică a programului BASIC-PLUS.

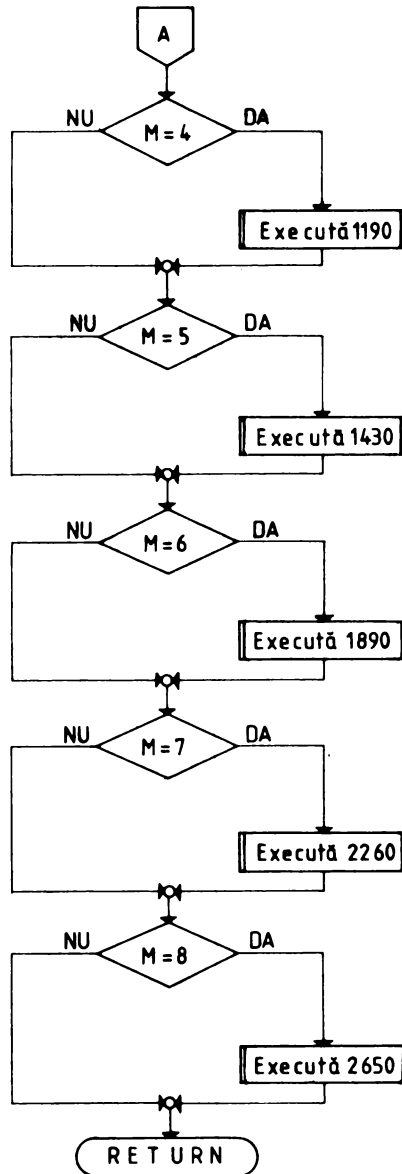


Fig. 11.2. d)

□ Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 240

Meniul conversației. Instrucțiunea ON ... GOSUB

În primul moment programul afișează pe ecran (v. liniile 160–220) meniul conversației:

1. Adăugare beneficiari.

2. Listare beneficiari
 3. Consultare beneficiari
 4. Sortare date
 5. Salvare fișier beneficiari
 6. Încărcare fișier beneficiari
- (0 pentru **END**)

Prin linia 240 sînteți invitați să tastați un număr cuprins între 1 și 6. Funcție de opțiunea dvs., prin instrucțiunea **ON ... GOSUB** se selectează una din cele șase subrutine ale programului:

270 **ON** OPT **GOSUB** 330, 460, 540, 850, 640, 730

Dacă OPT (v. linia 240) are valoarea 1, se selectează subrutina 330, iar dacă OPT are valoarea 2 se selectează subrutina 460 ș.a.m.d.

Formatul general al instrucțiunii este:

Format general
ON (expresie) GOSUB (listă nr. linii)

unde, (expresie) și (listă nr. linii) au aceeași semnificație ca la instrucțiunea **ON ... GOTO**.

Observație. Toate liniile care urmează după **GOSUB** trebuie să fie linii de început de subrutine.

Pentru valoarea zero a variabilei OPT, conversația ia sfîrșit (v. linia 280).

Secvența de instrucțiuni:

```
290 PRINT "Apăsați o tastă pentru reîntoarcerea în meniu"
300 IF INKEY$="" THEN 300
310 GOTO 150
320 PRINT
```

programează reîntoarcerea în meniu.

Pentru ieșirea din bucla 300 (v. linia 300) este obligatorie acționarea unei taste (oricare).

Opțiunea 1. Adăugare beneficiari

Prin tastarea cifrei 1 se selectează subrutina 330 – adăugare beneficiari. Un beneficiar este introdus în vectorul de date BENS\$ (v. linia 380) atunci cînd nu este respectată condiția exprimată în **IF**-ul din linia 370. Ieșirea din bucla **FOR-NEXT** (liniile 350–430) are loc cînd BENS(I)=" " (șir vid – v. linia 390).

```
330 REM Adăugare beneficiari
340 III=0
350 FOR I=1 TO 30
360   CLS
370   IF LEN (BENS(I)) > 0 THEN 430
380   INPUT "Nume beneficiar"; BENS(I)
390   IF BENS(I)=" " THEN 440
400   III=III+1
```

```

410 INPUT "Livrări"; CANT(I)
420 INPUT "Preț tonă"; PREȚ(I)
430 NEXT I
440 PRINT "Terminat introducere date"
450 RETURN

```

Funcția LEN

În linia 370 a subrutinei

```
370 IF LEN (BENS(I)) > 0 THEN 430
```

funcția **LEN(BENS(I))** calculează lungimea șirului de caractere **BENS(I)**.

Formatul funcției este:

Format general
LEN (<X\$>)

unde **X\$** reprezintă șirul de caractere.

Funcția **LEN** returnează lungimea șirului **X\$**.

Observație. Sint numărate atit caracterele netipăribile cit și spațiile.

Opțiunea 2. Listare beneficiari

Cu o buclă de tip **FOR-NEXT** se listează (v. linia 500) pentru fiecare beneficiar în parte (maximum 30) cantitatea de benzină livrată și prețul. Pentru **BENS(I)=" "** (v. linia 490) se părăsește forțat bucla.

```

460 REM Listare beneficiari
470 PRINT
480 FOR I=1 TO 30
490 IF BENS(I)= " " THEN 520
500 PRINT BENS(I); " "; CANT(I); " "; PREȚ(I)
510 NEXT I
520 PRINT "Terminat listare fișier"
530 RETURN

```

Opțiunea 3. Consultare beneficiari

Prin tastarea cifrei 3 se selectează subrutina 540 (v. liniile 540–630).

```

540 REM Consultare fișier beneficiari
550 PRINT
560 INPUT "Ce beneficiar căutați"; NBENS$
570 FOR I=1 TO 30
580 IF INSTR (BENS(I), NBENS$)=0 THEN 610
590 PRINT BENS(I); " "; CANT(I); " "; PREȚ(I)
600 RETURN
610 NEXT I
620 PRINT "Beneficiarul"; NBENS$; "nu este în fișier"
630 RETURN

```

Vom decupa în cele ce urmează din bucla **FOR-NEXT** instrucțiunea din linia 580.

```
580 IF INSTR (BEN$(I), NBEN$)=0 THEN 610
```

Funcția **INSTR** caută prima apariție a șirului **NBEN\$** (numele beneficiarului căutat – v. linia 560) în șirul **BEN\$** (nume beneficiar) și returnează poziția de la care începe corespondența.

Formatul general al instrucțiunii este:

Format general
INSTR [(<i>i</i> .)](X\$).(Y\$)

unde:

X\$ reprezintă șirul în care se caută șirul **Y\$**;

i precizează poziția de la care se începe căutarea.

Reguli

- Dacă *i* > **LEN(X\$)** sau **X\$** este nul, sau **Y\$** nu poate fi regăsit în **X\$**, funcția returnează valoarea zero.
- Dacă **Y\$** este șir nul, funcția are ca rezultat *i* (dacă există) sau 1.

Opțiunea 5. Salvare fișier

Prin tastarea cifrei 5 (am sărit intenționat opțiunea 4) se selectează subrutina 640.

```
640 REM Salvare fișier beneficiari
650 PRINT
660 OPENOUT "BENEF. DAT"
670 FOR I=1 TO 30
680   WRITE #9, BEN$(I), CANT(I), PREȚ(I)
690 NEXT I
700 CLOSEOUT
710 PRINT "==Fișier salvat=="
720 RETURN
```

După cum ați putut constata și singuri, nimic nu este complicat în această subrutină.

Încărcare fișier

Prin tastarea cifrei 6 se selectează subrutina (730) de încărcare fișier în memorie.

```
730 REM Citire fișier
740 III=0
750 PRINT
760 OPENIN "BENEF. DAT"
770 FOR I=1 TO 30
780   INPUT #9, BEN$(I), CANT(I), PREȚ(I)
790   IF BEN$(I)=" " THEN 820
800   III=III+1
810 NEXT I
```

```

820 CLOSEIN
830 PRINT "Fișier încărcat"
840 RETURN

```

De notat că subrutina încarcă și memorie cel mult 30 de beneficiari. În momentul cînd $BEN\$(I)=""$ (șir vid) fișierul BENEF.DAT este închis și pe ecran apare mesajul „Fișier încărcat”.

Opțiunea 4. Sortare date

Iată-ne ajunși la opțiunea 4. Prin tastarea cifrei 4 se selectează subrutina 850.

```

850 REM Subrutină de sortare date
860 GOSUB 1500
870 GOSUB 1340
880 RETURN

```

Subrutina 1500 se apelează pentru afișarea listei nesortate a beneficiarilor.

```

1500 REM
1510 PRINT "Lista nesortată : " : PRINT
1520 PRINT "Beneficiari/Cantități livrate/Preț" : PRINT
1530 FOR B=1 TO III : PRINT BEN$(B); TAB(21); CANT(B); TAB(31); PREȚ(B) :
NEXT B
1540 RETURN

```

Prin apelarea subrutinei 1340 se afișează (v. liniile 1360–1450) submeniuul subrutinei de sortare.

```

1340 PRINT : PRINT "Apăsați o tastă"
1345 IF INKEY$="" GOTO 1345
1350 CLS
1360 PRINT TAB(5); " Metoda de sortare : "
1370 PRINT TAB(7); " 1. BUBBLE DEMO"
1380 PRINT TAB(7); " 2. BUBBLE SORT"
1390 PRINT TAB(7); " 3. Sortare prin extracție"
1400 PRINT TAB(7); " 4. Sortare prin inserție"
1410 PRINT TAB(7); " 5. Sortare indexată"
1420 PRINT TAB(7); " 6. SHELL"
1430 PRINT TAB(7); " 7. QUICK SORT (versiunea 1)"
1440 PRINT TAB(7); " 8. QUICK SORT (variante structurată)"
1450 PRINT TAB(7); "(0 pentru END)"
1460 PRINT : PRINT TAB(5); INPUT "Metoda (1–8)"; MET
1470 ON MET GOSUB 890, 1550, 1730, 1940, 2180, 2520, 2870, 3200
1480 IF MET <> 0 THEN 1340
1490 RETURN

```

BUBBLE DEMO

Prin tastarea cifrei 1 se selectează subrutina 890 (v. liniile 890–1330) care constituie un program demonstrativ, pe pași (v. linia 970) privind modul de lucru al sortării prin metoda "bulelor de aer" (BUBBLE, în l. engleză).

Observație. Instrucțiunile 1040, 1050 și 1060 din subrutina 890 listată în fig. 11.1. g se vor înlocui cu:

1040 V\$=BEN\$(C)
 1041 BEN\$(C)=BEN\$(C+1)
 1042 BEN\$(C+1)=V\$
 1050 Z=CANT(C)
 1051 CANT(C)=CANT(C+1)
 1052 CANT(C+1)=Z
 1060 U=PREȚ(C)
 1061 PREȚ(C)=PREȚ(C+1)
 1062 PREȚ(C+1)=U

BUBBLE SORT

Prin tastarea cifrei 2 se selectează subrutina 1550 (v. liniile 1550–1720). Subrutina precedentă a fost scrisă pentru a vă ajuta să înțelegeți cum lucrează sortarea BUBBLE. Dar, într-o sortare “normală” a datelor nu este nevoie să vedeți fiecare trecere. Subrutina va imprima doar lista finală sortată (v. rezultatele execuției). Subrutina compară cheile, două câte două și schimbă elementele dacă nu este îndeplinită condiția din linia 1610.

Remarcă. Instrucțiunile din liniile 1620, 1630, 1640 se vor înlocui în aceeași manieră în care am procedat cu subrutina 890 (v. observație).

Sortare prin extracție

Prin tastarea cifrei 3 se selectează subrutina 1730 (v. liniile 1730–1930). Subrutina extrage elementul având cheia cu valoarea cea mai mică și îl schimbă cu elementul din stînga tabloului.

Remarcă. Instrucțiunile BASIC-AMSTRAD din liniile 1870, 1880 și 1890 se vor înlocui în aceeași manieră în care am procedat la subrutina 890 (v. observație).

Sortare prin inserție

Prin tastarea cifrei 4 se selectează subrutina 1940 (v. liniile 1940–2020). Subrutina inserează un element al tabloului într-o zonă deja ordonată a acestui tablou, după care prelucrează elementele care rămîn.

Sortare indexată

Prin tastarea cifrei 5 se selectează subrutina 2250 (v. liniile 2250–2510). Subrutina sortează fișierul de livrări după trei chei (la alegere): Beneficiar, Cantitate și Pret. Cele trei cîmpuri de sortare se selectează prin tastarea 1/2/3.

Funcția STR\$

În liniile 2220 și 2230 ale subrutinei

2220 R1=CANT(R) : N\$(R,2)=STR\$(R1)
 2230 R2=PREȚ(R) : N\$(R,3)=STR\$(R2)

funcția **STR\$** convertește valoarea lui R1 și R2 într-un șir de caractere. Poate vi se pare curios efectul acestei funcții, dar este mult mai ușor de manipulat un număr sub formă de șir decât sub forma lui numerică.

Formatul general al funcției este:

Format general
STR\$ ((expresie numerică))

Sortarea SHELL

Prin tastarea cifrei 6 se selectează subrutina 2520. Subrutina SHELL, numită astfel după numele inventatorului metodei de sortare – SHELL, este mult mai rapidă decât cele prezentate anterior.

SHELL a avut ideea efectuării unor salturi importante în tablou pentru plasarea mai rapidă a elementelor. Schimbarea pe o "distanță" mai mare a elementelor este mult mai eficientă decât efectuarea metodică de schimbări ale unei singure poziții deodată. "Distanța" dintre elementele comparate se numește *interval* iar în linia 2700 este readusă la $G=1$, dar schimbată în $G/2$ în linia 2720.

Deci, pentru $l=30$ elemente, intervalul de pornire este $INT(30/2)=15$. În cele din urmă, G este adus la un "mic interval" egal cu 1. Cu alte cuvinte, subrutina este identică cu BUBBLE. Dar, de data aceasta datele vor fi "aproape ordonate". Variabila F face ca programul să se oprească atunci când nu se mai pot face schimbări ($F=0$). Pentru a înțelege bine subrutina va trebui să considerați cel puțin 16 valori ale tabloului, folosind un interval de comparație de pornire – 8.

Remarcă. Instrucțiunile **SWAP** din liniile 2800 și 2810 se vor înlocui în aceeași manieră în care am procedat cu subrutina 890 (v. observație).

Subrutina afișează numărul de comparații (C) și numărul de schimbări (S) efectuate.

QUICK SORT (versiunea 1)

Prin tastarea cifrei 7 se selectează subrutina 2870 (v. liniile 2870–3130).

Metoda QUICK SORT (sortare rapidă) este foarte mult apreciată de către profesioniști (este de ordinul $n * \log n$). Metoda este foarte asemănătoare cu modul în care unii oameni efectuează sortarea natural (împărțirea unei probleme în subprobleme ș.a.m.d).

Funcția aritmetică SGN. Funcția **SGN** din linia 2970 stabilește semnul lui S .

2970 $S=SGN(-S)$

SGN returnează valorile: -1 (dacă valoarea este negativă), 0 (dacă ea este nulă) și 1 (dacă ea este pozitivă).

Formatul funcției este:

Format general
SGN ((expresie numerică))

Remarcă. Instrucțiunile **SWAP** din linia 2960 se vor înlocui în aceeași manieră în care am procedat cu subrutina 890 (v. observație).

QUICK SORT (varianta structurată)

Prin tastarea cifrei 8 se selectează subrutina 3200. Ea reprezintă varianta structurată a subrutinei QUICK SORT anterioare (2870). De remarcat prezența variabilelor S_9 , S^0_0 . Atenție la liniile 3330 și 3340!

Remarcă. Instrucțiunile **SWAP** din liniile 3300 și 3320 se vor înlocui în aceeași manieră în care am procedat cu subrutina 890 (v. observație).

TEST

Următorul program nestructurat, scris în BASIC HC-85, TIM S, SPECTRUM realizează căutarea într-un șir, prin metoda biseției, a unui anumit nume. Extindeți metoda pentru un fișier BASIC-AMSTRAD secvențial sortat după o anumită cheie.

```

10 CLEAR
20 DIM n$(20, 6)
30 DIM t$(20, 4)
40 LET i=1
50 INPUT n$(i)
60 INPUT t$(i)
65 IF n$(i)="* * " THEN GO TO 90
70 LET i=i+1
80 GO TO 50
90 REM
100 LET n=i
150 INPUT "Ce nume căutați"; q$
210 PRINT "1"; TAB 5; "h"; TAB 10;
    "m"; TAB 15; "n$(m)"
215 REM l → limita inferioară;
    h → limita superioară
216 REM m → media
220 LET l=1
225 LET h=n
230 REM
235 IF h-l=1 THEN GOTO 500
240 LET m=INT((l+h)/2)
250 PRINT l; TAB 5; h; TAB 10; m;
    TAB 15; n$(m)
260 IF q$=n$(m) THEN GOTO 320
270 IF q$<n$(m) THEN GOTO 300
280 LET l=m
290 GOTO 230
300 LET h=m
310 GOTO 230
320 REM
330 PRINT q$; t$(m)
350 GOTO 600
510 PRINT q$; "nu este în listă"
600 INPUT "Doriți să căutați alt, nume";
    Y$
610 IF Y$="DA" THEN GOTO 150
620 STOP

```

Codificarea în limbajul BASIC-COMMODORE

În comparație cu programul BASIC-AMSTRAD, diferențele de scriere a programului apar în: linia 260, 300, 360, 580 (limbajul BASIC-COMMODORE nu acceptă funcția **INSTR**), 660, 680, 700, 760, 820, 1040, 1050, 1060, (limbajul BASIC-COMMODORE nu acceptă funcția **SWAP**), 1350, 1620, 1630, 1640, 1870, 1880, 1890, 2260, 2410, 2710, 2800, 2810, 2850, 2960, 3240, 3260, 3280, 3300, 3310, 3320, 3350, 3410, 3420.

Aplicație. Se consideră un fișier dicționar (cod, denumire) secvențial cu 30 de articole. Fișierul este sortat ascendent după cod. Apelînd una din subrutinele de mai jos (verifică dacă un număr K se află în vectorul A cu 30 elemente), să se cerceteze dacă un anume beneficiar al cărui cod se citește în variabila A se găsește în fișierul dicționar.

a)

```

10 DIM A(30)
70 I=1
75 IF K=A(I) THEN 120
80 I=I+1
90 IF I<=30 THEN 75
100 PRINT "NUMĂRUL"; K; "NU
    ESTE ÎN VECTOR"
110 END
120 PRINT I, A(I)
130 RETURN

```

b)

```

60 Q=INT (SQR(30))
70 I=1-Q
80 INPUT "NR. CĂUTAT"; K
90 I=I+Q
100 IF I>=30 THEN 190
110 IF K>A(I) THEN 90
120 IF K=A(I) THEN 240
130 IF I=1 THEN 170
140 FOR J=I-1 TO I-Q+1 STEP -1
150   IF K=A(J) THEN 230
160 NEXT J
170 PRINT
180 END

```

c)

```

190 FOR J=30 TO I-Q+1 STEP -1
200   IF K=A(J) THEN 230
210 NEXT J
220 GOTO 170
230 I=J
240 PRINT I, A(I)
250 PRINT
260 RETURN

```

c)

```

70 L=1
80 U=30
90 IF U<L THEN 200
100 I=INT (L+U)/2
110 IF K<A(I) THEN 180
120 IF K=A(I) THEN 150
130 L=I+1
140 GO TO 90
150 PRINT I, A(I)
160 PRINT
170 END
180 U=I-1
190 GO TO 90
200 PRINT
210 RETURN

```

□ Particularități ale programării în limbajul BASIC-80

vol. 2, pag. 248

Funcția SWAP

Lista programului o puteți consulta în vol. 2, pag. 248. În cele ce urmează ne vom opri asupra unui singur element de... nouitate, comparativ cu programele precedente. Este vorba de funcția **SWAP** pe care ați întilnit-o în toate subrutinele de sortare, mai puțin subrutina 1730 (sortare prin extracție).

```

1040 SWAP BEN$(C), BEN$(C+1)
1050 SWAP CANT(C), CANT(C+1)
1060 SWAP PREȚ(C), PREȚ(C+1)

```

SWAP schimbă între ele valorile a două variabile. În linia 1040 valoarea lui BEN(C)$ trece în locul valorii lui BEN(C+1)$ și invers.

Formatul funcției **SWAP** este:

Format general
SWAP <variabilă>, <variabilă>

Remarcă. Cele două variabile trebuie să aibă același tip (întreg, simplă precizie, dublă precizie, șir).

Aplicație. Introduceți, executați și analizați următoarele programe:

- a) 10 **REM**
 20 **INPUT** "Introduceți un cuvint", a\$
 30 i=1
 40 na=0
 50 cont=1
 60 **WHILE** cont=1
 70 poz=INSTR (i, a\$, "a")
 80 **IF** poz>0 **THEN** na=na+1 : i=i+1 **ELSE** cont=0
 90 **WEND**
 100 **PRINT** "Litera a apare de"; na; "in cuvintul dvs."
- b) 20 **INPUT** "Introduceți data curentă zz/ll/aa"; data\$
 30 zi=VAL (data\$)
 40 mp=INSTR (data\$, "/")
 50 mh\$=MID\$ (data\$, mp+1)
 60 mh=VAL (mh\$)
 70 ypoz=INSTR (mp+1, data\$, "/")
 80 yp\$=MID\$ (data\$, ypoz+1)
 90 **FOR** i=1 **TO** mh
 100 **READ** a\$, b
 110 **NEXT**
 120 zi=zi+7
 130 **IF** zi>b **THEN** zi=zi-b : **READ** a\$, b
 140 wf\$=a\$+STR\$(zi)+" , 19"+yp\$
 150 **PRINT** : **PRINT** wf\$
 160 **DATA** Ianuarie, 31, Februarie, 28, Martie, 31, Aprilie, 30, Mai, 31, Iunie, 30
 170 **DATA** Iulie, 31, August, 31, Septembrie, 30, Octombrie, 31, Noiembrie, 30, Decembrie, 31

Observație. Funcția **MID\$** returnează un subșir de (j) caractere începând cu poziția (i).

Formatul general al funcției este:

Format general
MID\$ ((x\$), [i, j])

- c) 10 A\$="12345ABC"
 20 NR=VAL(A\$)
 30 **PRINT** NR
 40 **END**
- d) 10 **REM**
 20 **FOR** i=65 **TO** 90
 30 **PRINT** CHR\$(i)
 40 **NEXT**

Observație. Funcția **CHR** (i) generează un caracter cu codul ASCII (i).

- e) 10 **REM**
 20 **FOR** i=0 **TO** 31
 80 **PRINT** CHR\$(1); CHR\$(i);
 40 **NEXT** i
- f) 10 **REM**
 20 i=1
 30 **WHILE** i=1
 40 a\$=INKEY\$
 50 **IF** a\$ < > " " **THEN** a=ASC(a\$)
 60 **IF** a>64 **AND** a<91 **THEN**
PRINT a\$
 70 **WEND**
- g) 10 **PRINT**
 20 **INPUT** "Introduceți codul"; cod
 30 **PRINT**
 40 **LINE INPUT** "Introduceți mesaj"; mesaj\$
 50 lung=LEN (mesaj\$)
 60 codmesaj\$=" "
 70 **FOR** i=1 **TO** lung
 80 litera\$=MID\$ (mesaj\$, i, 1)
 90 literaascii=ASC (litera\$)
 100 codlitera\$=CHR\$ (literaascii+cod)

```

110 codmesaj$=codmesaj$+
    codliteră$
120 NEXT
130 PRINT

140 PRINT "Codul mesajului este:"
150 PRINT
160 PRINT codmesaj$
170 END

```

Observație. Instrucțiunea **LINE INPUT** (v. linia 40) citește o linie terminată cu [CR] în variabila de tip șir "mesaj\$" de la consolă (v. vol. 2, pag. 269).

```

h) 1000 REM "SORTARE DUPĂ TABEL"
1010 DIM BEN$(20), R(20, 20)
1020 INPUT "NUMĂR REZERVORE : ", N : INPUT "NUMĂR BENEFICIARI : ", M
1030 INPUT "NUMĂRUL REZERVORULUI DUPĂ CARE SE FACE SORTAREA", L
1040 FOR I=1 TO M
1050 PRINT "NUMELE BENEFICIARULUI NR.", I, " " : INPUT BEN$(I)
1060 NEXT I
1070 FOR I=1 TO N
1080 PRINT " *** REZERVORUL "; I
1090 FOR J=1 TO M
1100 PRINT "CANTITATEA LIVRATĂ BENEFICIARULUI";
    BEN$(J); : INPUT R(I, J)
1110 NEXT J
1120 NEXT I
1130 LPRINT "Lista nesortată : "
1140 GOSUB 1270
1150 FOR I=1 TO M
1160 FOR K=I+1 TO M
1170 IF R(I, I) >= R(I, K) THEN 1220
1180 SWAP BEN$(I), BEN$(K)
1190 FOR J=1 TO N
1200 SWAP R(J, I), R(J, K)
1210 NEXT J
1220 NEXT K
1230 NEXT I
1240 LPRINT "Sortare după rezervorul"; L
1250 GOSUB 1270
1260 END
1270 LPRINT "
1280 LPRINT " ! NUME BENEF. ! REZ1 ! REZ2 ! REZ3 ! REZ4 ! "
1290 LPRINT "
1300 FOR I=1 TO M
1310 LPRINT " ! " ; BEN$(I) ; TAB (16);
1320 FOR J=1 TO N
1330 LPRINT USING "##.# ##"; R(J, I);
1340 LPRINT " ! " ;
1350 NEXT J
1360 LPRINT
1370 NEXT I
1380 LPRINT "
1390 RETURN

```

Lista nesortată:

! NUME BENEF. !	REZ1 !	REZ2 !	REZ3 !	REZ4 !
! PECO BUZĂU	4.678 !	36.911 !	211.600 !	5.947 !
! PECO CLUJ	6.975 !	43.870 !	321.343 !	243.199 !
! COMB. BRAZI	12.457 !	24.766 !	3.203 !	56.978 !

Sortare după rezervorul 4

! NUME BENEF. !	REZ1 !	REZ2 !	REZ3 !	REZ4 !
! PECO CLUJ	6.975 !	43.870 !	321.343 !	243.199 !
! COMB. BRAZI	12.457 !	24.766 !	3.203 !	56.978 !
! PECO BUZĂU	4.678 !	36.911 !	211.600 !	5.947 !

□ **Particularități ale programării
în limbajul BASIC-PLUS**

vol. 2, pag. 255

Programul sortează fișierul masiv virtual BENEF. DAT creat în conversația 10. Pentru început, este apelată subrutina 220 (v. linia 180). La rîndul ei, subrutina apelează patru subrutine: 3050 (numărare beneficiari); 2890 (deschidere fișier); 510 (corp program+menu) și 2950 (închidere fișier).

Conversația propriu-zisă începe în subrutina 510 odată cu afișarea meniului. În variabila M se citește dinamic opțiunea utilizatorului. Se selectează pe rînd subrutinele: 270 (BUBBLE DEMO), 800 (BUBBLE SORT), 980 (sortare prin extracție), 1190 (sortare prin inserție), 1430 (sortare indexată), 1890 (SHELL), 2260 (QUICK SORT – versiunea 1), 2650 (QUICK SORT – varianta structurată).

Cînd M ia valoarea 9 se execută subrutina de închidere fișier (2950), după care conversația cu minicalculatoarele INDEPENDENT, CORAL ia sfîrșit.

În figura 11.3 puteți urmări modul în care s-a efectuat sortarea fișierului masiv virtual de livrări prin toate cele opt metode de sortare.

METODA DE SORTARE :		
1. BUBBLE DEMO		
2. BUBBLE SORT		
3. SORTARE PRIN EXTRACȚIE		
4. SORTARE PRIN INSERȚIE		
5. SORTARE INDEXATA		
6. SHELL		
7. QUICK SORT (VERSIUNEA 1)		
8. QUICK SORT (VARIANTA STRUCTURATA)		
9. SFIRȘIT		
OPTIUNEA DVS. ESTE ?1		
LISTA BENEFICIARI (NESORTATA)		
PECO TITAN	1000	7500
PECO MILITARI	30000	7500
PECO-CV	6789	7800
SUCEAVA	5600	7800
CRAIOVA	4000	7800
PLOIESTI	5655	7600
BUZAU-CENTRU	3000	7800
IASI-COPOLI	300	7890
URZICENI	1000	7000
TITU	5600	7600
CERNAVODA	5400	6000
ARAD	5555	7780
ALBA-IULIA	500	7914
VALENI	1000	7400

a)

Fig. 11.3. Opt metode de sortare: a) BUBBLE DEMO;

*** INCEPUT FAZA ***

METODA DE SORTARE :

- 1. BUBBLE DEMO
- 2. BUBBLE SORT
- 3. SORTARE PRIN EXTRACȚIE
- 4. SORTARE PRIN INSERTIE
- 5. SORTARE INDEXATA
- 6. SHELL
- 7. QUICK SORT (VERSIUNEA 1)
- 8. QUICK SORT (VARIANTA STRUCTURATA)
- 9. SFIRISIT

OPTIUNEA DVS. ESTE ?3

LISTA SORTATA :

VALENI	1000	7400
URZICENI	1000	7000
TITU	5600	7600
SUCEAVA	5600	7800
PLOIESTI	5655	7600
PECO-CV	6789	7800
PECO-TITAN	1000	7500
PECO MILITARI	30000	7500
IASI-COPOU	300	7890
CRAIOVA	4000	7800
CERNAVODA	5400	6000
BUZAU-CENTRU	3000	7800
ARAD	5555	7780
ALBA-IULIA	500	7914

**** TERMINAT FAZA ****

b)

*** INCEPUT FAZA ***

METODA DE SORTARE :

- 1. BUBBLE DEMO
- 2. BUBBLE SORT
- 3. SORTARE PRIN EXTRACȚIE
- 4. SORTARE PRIN INSERTIE
- 5. SORTARE INDEXATA
- 6. SHELL
- 7. QUICK SORT (VERSIUNEA 1)
- 8. QUICK SORT (VARIANTA STRUCTURATA)
- 9. SFIRISIT

OPTIUNEA DVS. ESTE ?2

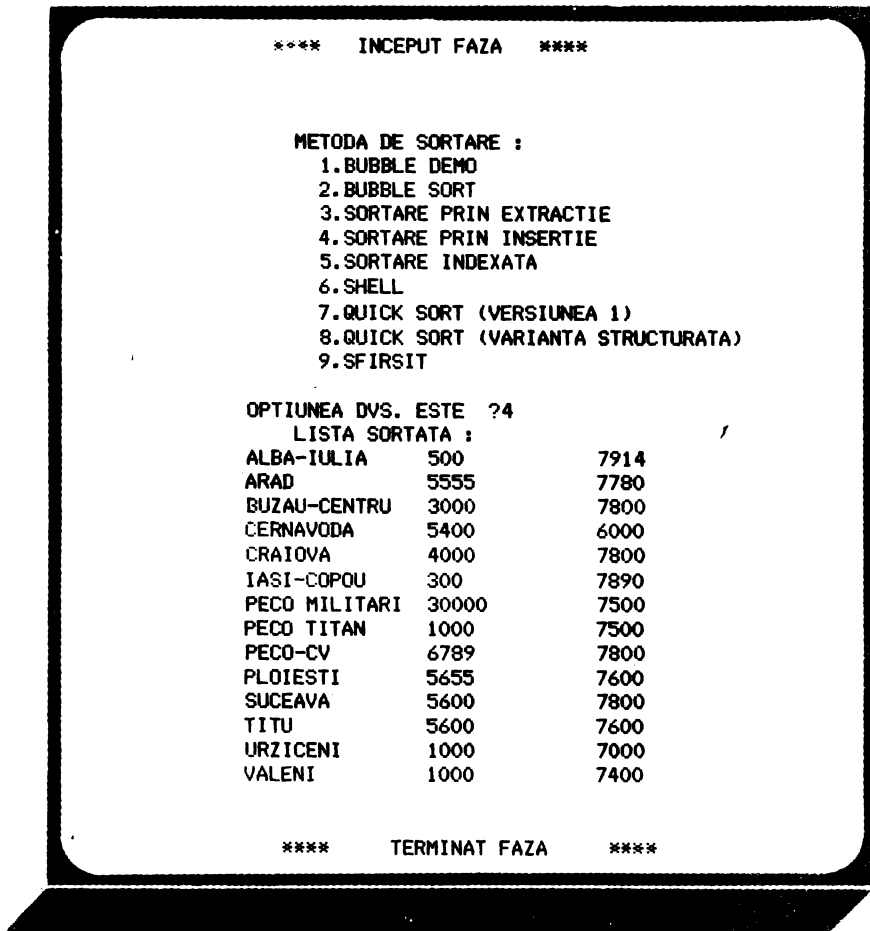
FISIER BENEFICIARI (SORTAT)

ALBA-IULIA	500	7914
ARAD	5555	7780
BUZAU-CENTRU	3000	7800
CERNAVODA	5400	6000
CRAIOVA	4000	7800
IASI-COPOU	300	7890
PECO MILITARI	30000	7500
PECO TITAN	1000	7500
PECO-CV	6789	7800
PLOIESTI	5655	7600
SUCEAVA	5600	7800
TITU	5600	7600
URZICENI	1000	7000
VALENI	1000	7400

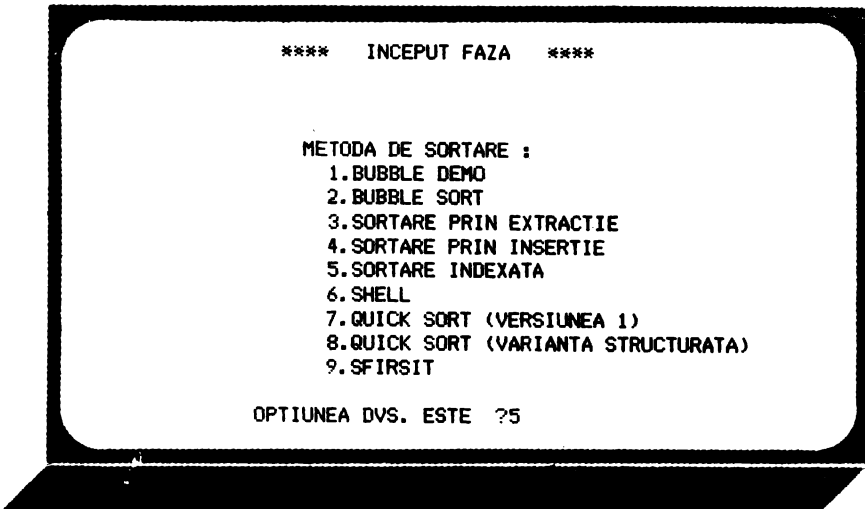
**** TERMINAT FAZA ****

c)

Fig. 11.3. b) sortare prin extracție; c) BUBBLE SORT;



d)



e)

Fig. 11.3. d) sortare prin inserție; e) sortare indexată;

PENTRU CONTINUARE TASTATI (Y/N) ?Y
 CIMPUL : 1.BENEFICIAR/2.CANTITATE/3.PRET (1/2/3) ?1
 INREGISTRARI SORTATE DUPA CIMPUL 1

ALBA-IULIA	500	7914	
ARAD	5555	7780	
BUZAU-CENTRU		3000	7800
CERNAVODA	5400	6000	
CRAIOVA	4000	7800	
IASI-COPOU	300	7890	
PECO MILITARI		30000	7500
PECO TITAN	1000	7500	
PECO-CV	6789	7800	
PLOIESTI	5655	7600	
SUCEAVA	5600	7800	
TITU	5600	7600	
URZICENI	1000	7000	
VALENI	1000	7400	

PENTRU CONTINUARE TASTATI (Y/N) ?Y
 CIMPUL : 1.BENEFICIAR/2.CANTITATE/3.PRET (1/2/3) ?2
 INREGISTRARI SORTATE DUPA CIMPUL 2

IASI-COPOU	300	7890	
ALBA-IULIA	500	7914	
VALENI	1000	7400	
URZICENI	1000	7000	
PECO TITAN	1000	7500	
BUZAU CENTRU		3000	7800
CRAIOVA	4000	7800	
CERNAVODA	5400	6000	
ARAD	5555	7780	
TITU	5600	7600	
SUCEAVA	5600	7800	
PLOIESTI	5655	7600	
PECO-CV	6789	7800	
PECO MILITARI		30000	7500

PENTRU CONTINUARE TASTATI (Y/N) ?Y
 CIMPUL : 1.BENEFICIAR/2.CANTITATE/3.PRET (1/2/3) ?3
 INREGISTRARI SORTATE DUPA CIMPUL 3

CERNAVODA	5400	6000	
URZICENI	1000	7000	
VALENI	1000	7400	
PECO TITAN	1000	7500	
PECO MILITARI		30000	7500
TITU	5600	7600	
PLOIESTI	5655	7600	
ARAD	5555	7780	
SUCEAVA	5600	7800	
PECO-CV	6789	7800	
CRAIOVA	4000	7800	
BUZAU-CENTRU		3000	7800
IASI-COPOU	300	7890	
ALBA-IULIA	500	7914	

PENTRU CONTINUARE TASTATI (Y/N) ?N

**** TERMINAT FAZA ****

Fig. 11.3. e)

**** INCEPUT FAZA ****

METODA DE SORTARE :

- 1.BUBBLE DEMO
- 2.BUBBLE SORT
- 3.SORTARE PRIN EXTRACTIE
- 4.SORTARE PRIN INSERTIE
- 5.SORTARE INDEXATA
- 6.SHELL
- 7.QUICK SORT (VERSIUNEA 1)
- 8.QUICK SORT (VARIANTA STRUCTURATA)
- 9.SFIRSIT

OPTIUNEA DVS. ESTE ?6

LISTA NESORTATA

VALENI URZICENI TITU SUCEAVA PLOIESTI
PECO-CV PECO TITAN PECO MILITARI IASI-COPOU CRAIOVA
CERNAVODA BUZAU-CENTRU ARAD ALBA-IULIA

LISTA SORTATA

ALBA-IULIA ARAD BUZAU-CENTRU CERNAVODA CRAIOVA
IASI-COPOU PECO MILITARI PECO TITAN PECO-CV PLOIESTI
SUCEAVA TITU URZICENI VALENI

73 COMPARATII

21 SCHIMBARI

**** TERMINAT FAZA ****

f)

**** INCEPUT FAZA ****

METODA DE SORTARE :

- 1.BUBBLE DEMO
- 2.BUBBLE SORT
- 3.SORTARE PRIN EXTRACTIE
- 4.SORTARE PRIN INSERTIE
- 5.SORTARE INDEXATA
- 6.SHELL
- 7.QUICK SORT (VERSIUNEA 1)
- 8.QUICK SORT (VARIANTA STRUCTURATA)
- 9.SFIRSIT

OPTIUNEA DVS. ESTE ?7

LISTA NESORTATA

VALENI URZICENI TITU SUCEAVA PLOIESTI
PECO-CV PECO TITAN PECO MILITARI IASI-COPOU CRAIOVA
CERNAVODA BUZAU-CENTRU ARAD ALBA-IULIA

LISTA SORTATA

ALBA-IULIA ARAD BUZAU-CENTRU CERNAVODA CRAIOVA
IASI-COPOU PECO MILITARI PECO TITAN PECO-CV PLOIESTI
SUCEAVA TITU URZICENI VALENI

14 BENEFICIARI

91 COMPARATII

7 SCHIMBARI

**** TERMINAT FAZA ****

g)


```

**** INCEPUT FAZA ****

METODA DE SORTARE :
1.BUBBLE DEMO
2.BUBBLE SORT
3.SORTARE PRIN EXTRACTIE
4.SORTARE PRIN INSERTIE
5.SORTARE INDEXATA
6.SHELL
7.QUICK SORT (VERSIUNEA 1)
8.QUICK SORT (VARIANTA STRUCTURATA)
9.SFIRSIT

OPTIUNEA DVS. ESTE ?8
LISTA NESORTATA
VALENI URZICENI TITU SUCEAVA PLOIESTI
PECO-CV PECO TITAN PECO MILITARI IASI-COPOU CRAIOVA
CERNAVODA BUZAU-CENTRU ARAD ALBA-IULIA
14 ARTICOLE
91 COMPARATII
7 SCHIMBARI

LISTA SORTATA
ALBA-IULIA ARAD BUZAU-CENTRU CERNAVODA CRAIOVA
IASI-COPOU PECO MILITARI PECO TITAN PECO-CV PLOIESTI
SUCEAVA TITU URZICENI VALENI

**** TERMINAT FAZA ****

```

h)

Fig. 11.3. h) QUICK SORT (VARIANTA STRUCTURATA).

Aplicație. Introduceți și executați următoarele programe. Transformați-le apoi în subrutine pentru aplicațiile dvs.

```

a)
10 INPUT "N="; N
20 DIM A(N)
30 FOR I=1 TO N
40 A(I)=RND(I)
50 NEXT I
60 FOR J=2 TO N
70 I=J-1
90 A=A(J)
100 IF K>=A(I) THEN 140
110 A(I+1)=A(I)
120 I=I-1
130 IF I>0 THEN 100
140 A(I+1)=K
150 NEXT J
160 FOR I=1 TO N
170 PRINT A(I)
180 NEXT I

60 FOR J=N TO 2 STEP -1
70 B=A(J)
80 L=J
90 FOR R=J TO 1 STEP -1
100 IF A(R)<=B THEN 130
110 L=R
120 B=A(R)
130 NEXT R
140 D=A(J)
150 A(J)=A(L)
160 A(L)=D
170 NEXT J
180 FOR I=1 TO N
190 PRINT A(I)
200 NEXT I
210 END

b)
10 INPUT "N="; N
20 DIM A(N)
30 FOR I=1 TO N
40 A(I)=RND(I)
50 NEXT I

c)
10 INPUT "N="; N
20 DIM A(N)
30 FOR I=1 TO N
40 A(I)=RND(1)
50 NEXT I
60 INPUT "T="; T
70 FOR S=T TO 1 STEP -1

```

```

80     PRINT S;
90     INPUT "H(S)="; H(S)
100    NEXT S
110    FOR S=T TO 1 STEP -1
140    H=H(S)
150    FOR J=H+1 TO N
160    I=J-H
170    A=A(J)
180    IF K>=A(I) THEN 220
190    A(I+H)=A(I)
200    I=I-H
210    IF I>0 THEN 180
220    A(I+H)=K
230    NEXT J
240    NEXT S
250    FOR I=1 TO N : PRINT A(I) :
260    NEXT I
260    END

```

Particularități ale programării în limbajul ABASIC

În comparație cu programul precedent diferențele de scriere a programului ABASIC apar în: linia 260, 300 (limbajul ABASIC nu acceptă **INKEY\$**), 360, 660, 760, 1350. De notat că limbajul ABASIC acceptă funcția **SWAP**.

Aplicație. Introduceți și executați următorul program de căutare într-un tablou a unui element prin metoda lui Fibonnacci.

```

10 READ N, L
20 DIM K(N), F(L)
30 FOR I=1 TO N
40 READ K(I)
50 NEXT I
60 F(1)=0
70 F(2)=0
80 FOR I=3 TO L
90 F(I)=F(I-1)+F(I-2)
100 NEXT I
110 INPUT "NR. CĂUTAT"; K
120 I=F(L-1)
130 P=F(L-2)
140 Q=F(L-3)
150 IF K<K(I) THEN 200
160 IF >K(I) THEN 280
170 PRINT I, K(I)
180 PRINT
190 GO TO 330
200 IF Q=0 THEN 260
210 I=I-Q
220 T1=P
230 P=Q
240 Q=T1-Q
250 GOTO 150
260 PRINT
270 GOTO 330
280 IF P=1 THEN 260
290 I=I+Q
300 P=P-Q
310 Q=Q-P
320 GOTO 150
330 END
340 DATA 12, 8
350 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
11, 12

```

TEMA 11

Răspundeți prin DA sau NU la următoarele întrebări:

- Funcția **LEN(Y\$)** returnează lungimea șirului de caractere **Y\$**.
- În instrucțiunea

```
10 IF INSTR(Y$, X$)=0 THEN 200
```

funcția **INSTR** caută prima apariție a șirului **X\$** în șirul **Y\$** și returnează poziția de la care începe corespondența.

- Un fișier masiv virtual poate fi sortat prin metoda SHELL.
- Sortarea indexată se aplică fișierele secvențiale ASCII.
- Limbajul ABASIC nu acceptă funcția **SWAP**.
- Limbajul BASIC-AMSTRAD admite funcția **SWAP**.
- În BASIC-COMMODORE, funcția **SWAP** schimbă între ele valorile a două variabile.

Înlocuiți cuvintele care lipsesc din propozițiile următoare:

a) Funcția **SWAP** este admisă de limbajul BASIC și de limbajul **BASIC**

b) În limbajul BASIC-AMSTRAD funcțiile standard pe șiruri de caractere sînt

c) Sortarea unui set de date poate fi realizată prin una din metodele:

d) Funcția **LEN** returnează iar funcția **INSTR\$**

e) Introduceți și executați următoarele două programe BASIC HC-85, TIM S, SPECTRUM.

```
10 LET a=RND * 100
20 PRINT a
30 LET a=a * 100
40 GOTO 20
```

și

```
10 LET a=RND * 100
20 PRINT TAB 15 - LEN STR$ INT a; a
30 LET a=a * 10
40 GO TO 20
```

În timp ce primul program afișează, cel de-al doilea program afișează

f) Care va fi rezultatul execuției următorului program BASIC-SPECTRUM?

```
10 PRINT 2
20 PRINT STR$ 2
30 PRINT 1/3
40 PRINT STR$ (1/3)
50 PRINT 9E15
60 PRINT STR$ 9E15
```

Scrieți cite un program BASIC pentru fiecare din problemele de mai jos:

a) Se consideră fișierul secvențial UTILAJ care conține numele și notele de la examenul de Programare ale studenților din grupa 1403. Să se listeze fișierul în ordine alfabetică și după notele obținute. Se va obține, de asemenea, o listă care să aibă structura: număr curent, nume student, notă,

rang. Prin rang se înțelege locul studentului în grupă, după nota sa; la note egale se păstrează ordinea din datele inițiale.

b) Se consideră fișierul dicționar ARMĂTURI – care conține codurile, denumirile și prețul produselor (armăturilor) fabricate de I.U.P. Tîrgoviște. Să se listeze fișierul în ordinea alfabetică a produselor și descrescător după preț.

c) Se consideră fișierul dicționar ONOMASTICI care conține numele și datele de naștere ale celor dragi. Să se listeze fișierul după data nașterii prietenilor.

d) Scrieți un program BASIC pentru interclasarea a două fișiere dicționar (cod, denumire) secvențiale sortate ascendent după cod.

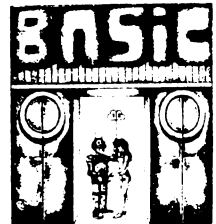
Indicație. Se va apela subrutina de interclasare (prezentată mai jos) pentru doi vectori X și Y ce conțin m, respectiv n elemente.

50	DIM X(M), Y(N), Z(M+N)	210	GOTO 290
100	I=1	220	Z(K)=Y(J)
110	J=1	230	K=K+1
120	K=1	240	J=J+1
130	IF X(I)>Y(J) THEN 220	250	IF J<=N THEN GOTO 130
140	Z(K)=X(I)	260	FOR K=1 TO M
150	K=K+1	270	Z(K+N)=X(K)
160	I=I+1	280	NEXT K
170	IF I<=M THEN 130	290	FOR I=1 TO M+N
180	FOR K=J TO N	300	PRINT Z(I)
190	Z(K+M)=Y(K)	310	NEXT I
200	NEXT K	320	RETURN

- Să se proiecteze și să se realizeze un program BASIC** ghidat de un meniu pentru sortarea fișierului secvențial JUCĂRII.DAT (v. Conversația 7) după cod, utilizând subrutinele BUBBLE SORT și SHELL (v. Conversația 11).
- Să se proiecteze și să se realizeze un program BASIC-PLUS** ghidat de un meniu pentru sortarea fișierului masiv virtual BENE.F.DAT (v. Conversația 10) după cod proveniență și preț de achiziție, utilizând subrutinele pentru sortarea indexată, extracție și inserție.

CONVERSAȚIA 12

Reprezentarea livrărilor de lichide din trei rezervoare cu ajutorul histogramelor. Tehnici de trasare a histogramelor în 2D și 3D. Programarea culorilor. Instrucțiunile grafice WINDOW, VIEWPORT, MOVE, DRAW, REMOVE, RDRAW, PLOT, POINT, CIRCLE. APLICAȚII și TESTE pentru cititor



□ Elemente de grafică interactivă

Nu știu alții cum sînt*, dar eu cînd mă gîndesc la orele de desen și la planșele (la scară și trase-n tuș) pe care trebuia să le arăt a doua zi profesorului mi se umple și-acum sufletul de neliniște. Neîndemînarea mea, lipsa de talent și sigur lipsa de efort susținut m-au ținut întotdeauna departe de blocul de desen, și mai tirziu de planșeta de la întreprindere. Din momentul cînd am descoperit că prietenul meu calculatorul știe să și deseneze vă mărturisesc că neliniștea a fost și mai mare dar de această dată bucuriile n-au întîrziat să apară.

Din dorința de a vă implica și pe dumneavoastră, stimați cititori în conversațiile de grafică cu calculatorul (aMIC, PRAE, HC-85, TIM S, SPEC-TRUM, AMSTRAD, COMMODORE, M118, TPD, JUNIOR) vă propunem să faceți o încercare desenînd pentru început figuri geometrice simple, după care veți vedea și singuri ce va urma.

Nu vă alarmați! Și noi am fost în situația dvs. nu cu mult timp în urmă. Vă vom explica tot ceea ce doream și noi să știm atunci cînd am început să desenăm cu calculatorul. Nu putem promite că veți cuceri universul după o jumătate de oră de conversație dar putem garanta că atunci cînd prietenii dvs. vă vor întreba cum le-ați luat-o înainte cu calculatorul personal îi veți uimi cu desenele, graficele ori animațiile realizate cu propriul efort.

Rezoluție, pixel, memorie ecran

În mod grafic, ecranul de vizualizare al calculatorului dvs. este „decupat” în puncte pe care le puteți aprinde sau stinge cu instrucțiuni grafice consacrate. Totalitatea punctelor distincte ce pot fi reprezentate pe un ecran (dispozitiv grafic) reprezintă **spațiul grafic**. Calitatea imaginilor obținute (pe ecran) depinde de ceea ce este cunoscut sub numele de **rezoluție** care este un atribut al spațiului grafic.

Rezoluția se măsoară prin numărul de puncte orizontale și verticale ce pot fi individualizate. Astfel, o rezoluție de 256×176 arată că există 256 de puncte pe axa orizontală a ecranului și 176 de puncte pe axa verticală.

În egală măsură este utilizat și termenul de **pixel** (prescurtare de la “picture elements”). De exemplu, ecranul TV, din punct de vedere al calculatorului PRAE este împărțit în 65 356 pixeli (256×256 puncte).

Majoritatea ecranelor pentru calculatoarele personale sînt dotate cu circuite de comandă compuse din **memorii ecran** sau **memorii video** (separate de memoria de lucru) de regulă, de tip RAM (Random Acces Memory).

* Ne amintim cu emoțiile copilăriei de marele nostru povestitor, Ion Creangă.

Utilizarea ferestrei și a vizorului

Instrucțiunile grafice... consacrate referă puncte ale spațiului grafic prin coordonatele lor (X , Y), exprimate în unități de măsură naturale, corespunzătoare mărimilor reprezentate grafic (newtoni, amperi, secunde etc.). Totalitatea valorilor de mai sus reprezintă "**spațiul utilizator**" sau virtual.

Observație. Spațiul virtual este limitat numai de precizia aritmeticii mașinii (1E-63, 8E+68) urmînd a fi reprezentat la o anumită scară, în cadrul spațiului grafic (pe suprafața display-ului).

Datorită faptului că spațiul ecranului este mult mai mic decît spațiul pe care îl are la dispoziție utilizatorul (desenatorul), se afișează pe ecran numai o parte din desen, la o scară care să permită o precizie acceptabilă. Această "fereastră" (WINDOW) permite analiza în detaliu a porțiunii alese din desen. În acest caz se au în vedere două probleme distincte: a) găsirea transformării care trebuie aplicate desenului inițial pentru a fi vizualizată pe ecran porțiunea dorită (WINDOWING); b) identificarea elementelor din desenul inițial care sînt vizibile în fereastră (CLIPPING).

Observație. Fereastra (porțiunea din desen care trebuie afișată) și vizorul (zona de pe ecran pe care se face afișarea) sînt două dreptunghiuri asemenea.

De notat că elementele care definesc fereastra (patru argumente) sînt date în unități de utilizator. În funcție de conținutul desenului inițial, ele pot fi: ani, km, lei etc. De multe ori, afișarea datelor cuprinse în fereastră nu ocupă întregul ecran, ci numai o parte a acestuia, în vederea plasării unor titluri, texte, indicații sau grafice suplimentare. Zona din ecran pe care va fi proiectat conținutul ferestrei o vom numi vizor. Elementele care definesc vizorul (patru argumente) sînt date în unitățile sistemului grafic utilizat.

Observație. Unitatea în care se exprimă limitele elementelor ce definesc vizorul s-a ales, ca fiind egală cu 1% din latura pătratului cel mai mare, ce poate fi înscris în suprafața ecranului (unitate grafică sau prescurtat U.G.).

Înaltă și joasă rezoluție

Datorită diferenței de rezoluție de la calculator la calculator și datorită faptului că fiecare sistem de calcul dispune de anumite facilități privind reprezentarea grafică se folosește și noțiunea de înaltă și joasă rezoluție. De exemplu, graficele alcătuite din segmente de dreaptă au o înaltă rezoluție pe cînd cele alcătuite din caractere semigrafice, de utilizator au o joasă rezoluție.

Sisteme de coordonate

Sistemul de coordonate carteziene. Cum o foaie de hîrtie nu are decît două dimensiuni (lungime și lățime) sînt suficiente două numere pentru a identifica toate punctele de pe suprafața acestei foi. Într-un sistem de coordonate carteziene xoy un punct M (v. figura 12.1) este reperat prin coordonatele $X=4$ și $Y=3$. Fiecare din axele OX și OY definesc atît direcția cît și

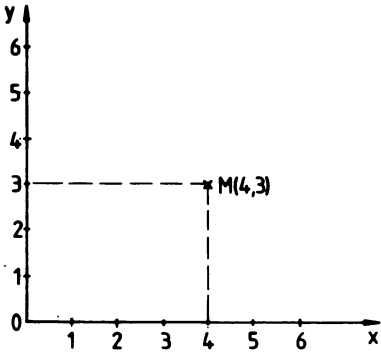


Fig. 12.1.

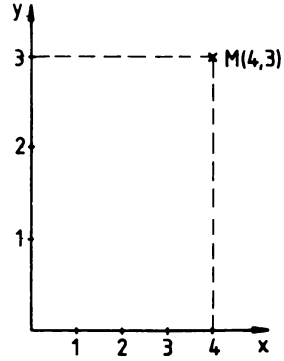


Fig. 12.2.

o unitate de măsură (v. figura 12.1 în care axele sînt perpendiculare și au aceeași unitate de măsură). Nu întotdeauna lucrurile stau la fel (v. fig. 12.2 și 12.3).

Sistemele frecvent utilizate (cu coordonate carteziene) sînt cele cu axele perpendiculare fără a avea însă mereu aceeași unitate de măsură pentru cele două axe (v. fig. 12.4 și 12.5).

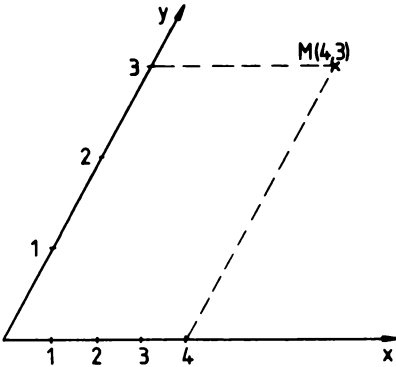


Fig. 12.3.

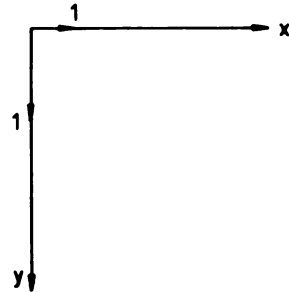


Fig. 12.4.

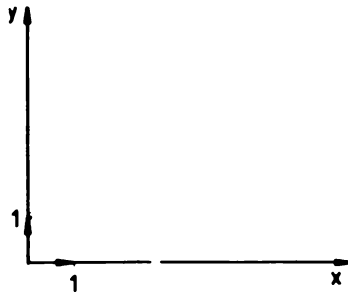


Fig. 12.5.

Sisteme de coordonate polare. Un punct $M(x, y)$ poate fi identificat (într-un sistem de coordonate polare) printr-un unghi (θ) și o lungime (ρ). În figura 12.6 este reprezentat un astfel de sistem.

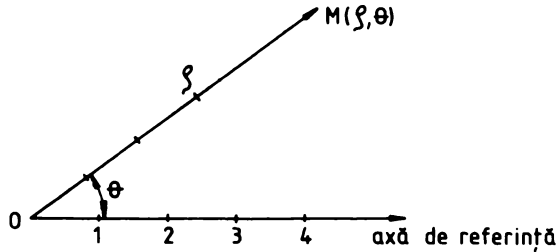


Fig. 12.6.

Trecerea de la un sistem de coordonate la altul. Să presupunem că: a) axa x , în coordonate carteziene se confundă cu axa de referință pentru coordonate polare; b) axele ox și oy sînt perpendiculare și au o unitate de măsură comună; c) originea axelor este comună.

În acest caz, punctul M (v. fig. 12.7) are coordonatele:

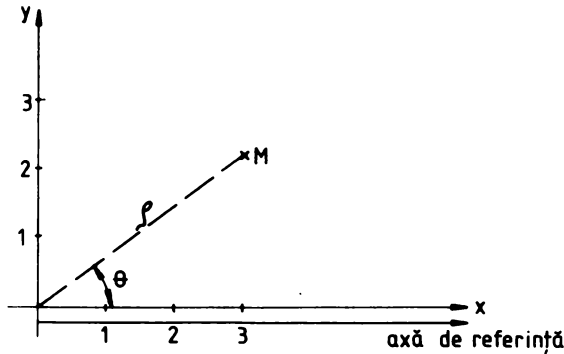


Fig. 12.7.

$X=3; Y=3$ (coordonate carteziene)

$\rho=3,5; \theta=60^\circ$ (coordonate polare).

Trecerea de la un sistem de referință la altul se obține prin relațiile:

$X = \rho \cos \theta$ (1) sau

$Y = \rho \sin \theta$ (2) $\theta = \arcsin \left(\frac{Y}{\sqrt{x^2 + y^2}} \right)$ (5)

$\rho = \sqrt{x^2 + y^2}$ (3) sau

$\theta = \arccos \left(\frac{x}{\sqrt{x^2 + y^2}} \right)$ (4) $\theta = \arctg \left(\frac{Y}{X} \right)$ (6)

□ Primitive grafice

Pentru a desena cu ... calculatorul, apare ca necesară cunoașterea și înțelegerea unor primitive (funcții) mult apropiate de limbajul matematic clasic. Cele mai utilizate primitive grafice sînt: PUNCT, LINIE, CERC, PĂTRAT etc.

În cele ce urmează le vom analiza în detaliu.

Primitiva PUNCT

Această primitivă este folosită pentru poziționarea spotului într-un punct de pe ecran.

Primitiva **PUNCT** are diverse denumiri în funcție de dialectul BASIC pe care-l utilizați.

În ceea ce ne privește vă propunem următorul format general:

Format general
PUNCT (X, Y)
Format general
RPUNCT (X, Y)

În primul format coordonatele X și Y sînt raportate la originea sistemului (coordoanate absolute) pe cînd în cel de-al doilea format coordonatele X și Y sînt raportate la poziția spotului (punctului grafic – coordonate relative).

Remarci

- Primitiva **PUNCT** are în general trei argumente: a) punct aprins/stins (sau culoare pentru sisteme policrome); b) coordonata X a punctului; c) coordonata Y a punctului.
- Cele trei valori (v. fig. 12.8) trebuie să fie definite înainte de a se apela primitiva.

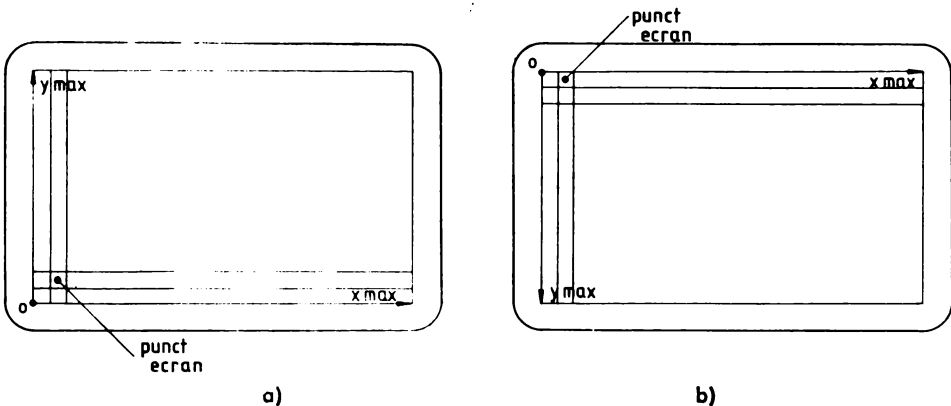


Fig. 12.8. Primitiva **PUNCT**: a) originea axelor în colțul din stînga jos; b) originea axelor în colțul din stînga sus.

- Valorile lui X și Y trebuie să satisfacă inegalitățile:

$0 \leq X \leq X_{\max}$ sau numărul maxim de puncte adresabile în direcția OX;

și

$0 \leq Y \leq Y_{\max}$ sau numărul maxim de puncte adresabile în direcția OY.

Primitiva LINIE

Pe un sistem video a cărui imagine finală este constituită dintr-o rețea de puncte, trasarea unui segment de dreaptă reprezintă o succesiune de apeluri ale primitivei **PUNCT (RPUNCT)**, simulind traseul segmentului prin "aprinderea" punctelor celor mai apropiate de segmentul adevărat (v. figura 12.9).

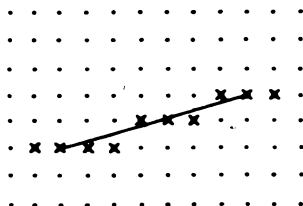


Fig. 12.9.

Ca și primitiva **PUNCT**, primitiva **LINIE** este implementată sub diverse denumiri în funcție de dialectul BASIC utilizat.

În cadrul acestei lucrări vă propunem următorul format general:

Format general
LINIE X, Y
Format general
RLINIE X, Y

În primul format coordonatele X și Y sînt raportate la originea sistemului. În cel de-al doilea format coordonatele X și Y sînt relative (fiind raportate la poziția spotului la momentul execuției primitivei – instrucțiunii).

Aplicație. Scrieți o secvență algoritmică (în pseudocod) pentru trasarea liniei din fig. 12.10.

```
AB SEQ
    PUNCT (20, 20)
    LINIE 40, 40
```

```
AB END
```

sau

```
AB SEQ
    RPUNCT (20, 20)
    RLINIE 20, 20
```

```
AB END
```

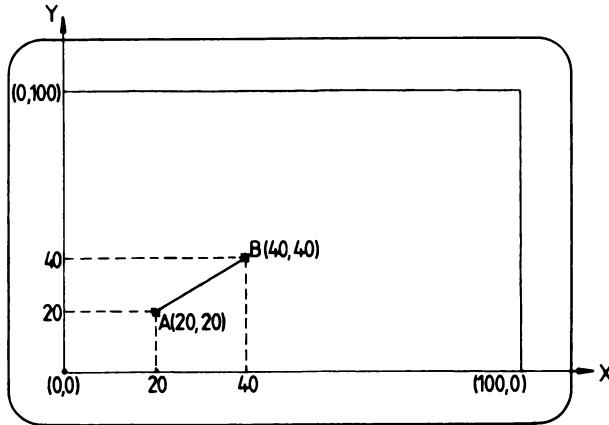


Fig. 12.10.

Primitiva CERC

Un cerc este în general definit prin coordonatele centrului și lungimea razei. De nenumărate ori ecuația sa

$$x^2 + y^2 - 2ax - 2by + c = 0 \quad (7)$$

$$(cu R^2 = a^2 + b^2 - c^2)$$

se restringe în:

$$x^2 + y^2 - R^2 = 0 \quad (8)$$

atunci când centrul cercului coincide cu originea axelor.

Primitiva **CERC** are și ea diverse denumiri în dialectele BASIC cu facilități grafice. În ceea ce ne privește vă propunem următorul format general:

Format general

CERC X, Y, R

unde X, Y reprezintă coordonatele centrului iar R este raza cercului.

Aplicație. Să se deseneze un cerc utilizând primitivele grafice **PUNCT** și **LINIE** (v. figura 12.11).

Procedura este următoarea. Vom aproxima arcul cu coarda generând arce mici (de exemplu $\frac{\pi}{10}$ radiani) după care le vom uni.

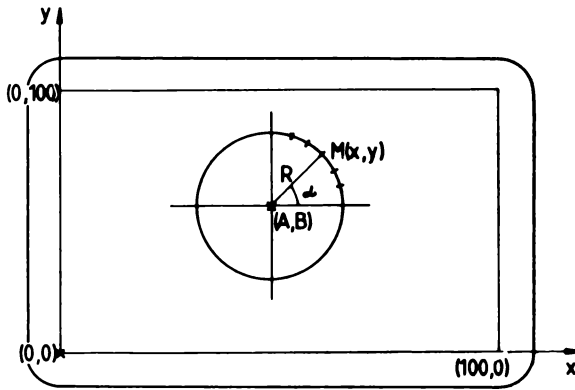
```

CERC      SEQ
              INPUT A, B, R
              PUNCT (A+R, B)

ARC      PENTRU I DE LA 0 LA 2 * PI PAS  $\frac{\pi}{10}$ 
              LINIE (A+R * COS(I), B+R * SIN (I))

ARC      SFIRȘIT
CERC      END

```



NOTA

$$x_A = R \cos \alpha$$

$$y_B = R \sin \alpha$$

$$x = A + R \cos \alpha$$

$$y = B + R \sin \alpha$$

Fig. 12.11.

Primitiva PĂTRAT

Un pătrat este în general definit prin coordonatele (x, y) centrului, lungimea laturii (A) și unghiul α (ALFA) format de o latură cu direcția axei OX (v. fig. 12.12).

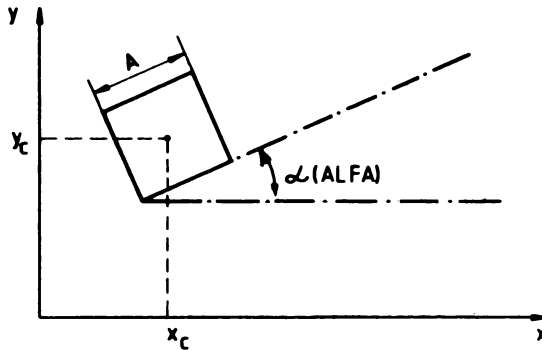


Fig. 12.12.

În cadrul acestei lucrări vom folosi următorul format general

Format general
PĂTRAT X, Y, A, ALFA

în care X, Y, A și ALFA au semnificația din figura 12.12.

TEST

Să se deseneze un pătrat definit prin: coordonatele centrului (X_c , Y_c), lungimea diagonalei (d) și unghiul β (v. fig. 12.13). Se vor da două soluții.

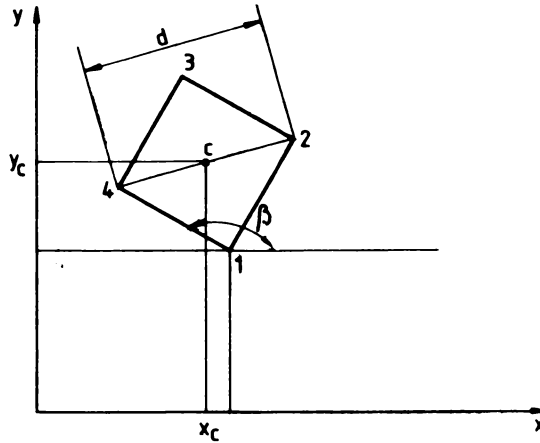


Fig. 12.13.

Aplicație. Următorul subalgoritm

```

DR      SEQ
        PUNCT (A, B)
        LINIE (A+L/2, B)
        LINIE (A+L/2, B+D)
        LINIE (A, B+D)
DR      END
  
```

desenează o jumătate de dreptunghi (v. figura 12.14). Pornind de la acest subalgoritm desenați un dreptunghi.

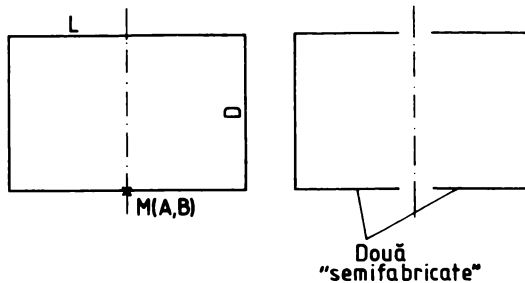


Fig. 12.14.

```

DREPT  SEQ
        INPUT A, B, D
D      PENTRU I DE LA 1 LA 3 PAS 2
        PUNCT (A, B)
        LINIE (A+J-2) * L/2, B)
        LINIE (A+J-2) * L/2, B+D)
        LINIE (A, B+D)
D      SFIRȘIT
DREPT  END
  
```

□ Să desenăm un roboțel
cu calculatoarele aMIC și PRAE

Ne propunem să desenăm un mic robot (cu calculatoarele aMIC și PRAE) a cărui înfățișare este ilustrată în figura 12.15. Poate mulți dintre dvs. zîmbesc în acest moment! Desigur orice chip poate fi și infrumusețat! În ceea ce ne privește am urmărit ca roboțelul să nu aibă capul "pătrat", și să poată fi desenat fără efort; după cum remarcăți cu multă "risipă" de dreptunghiuri simetrice față de axa verticală (!). O singură excepție face nasul care este de formă . . . triunghiulară (!)

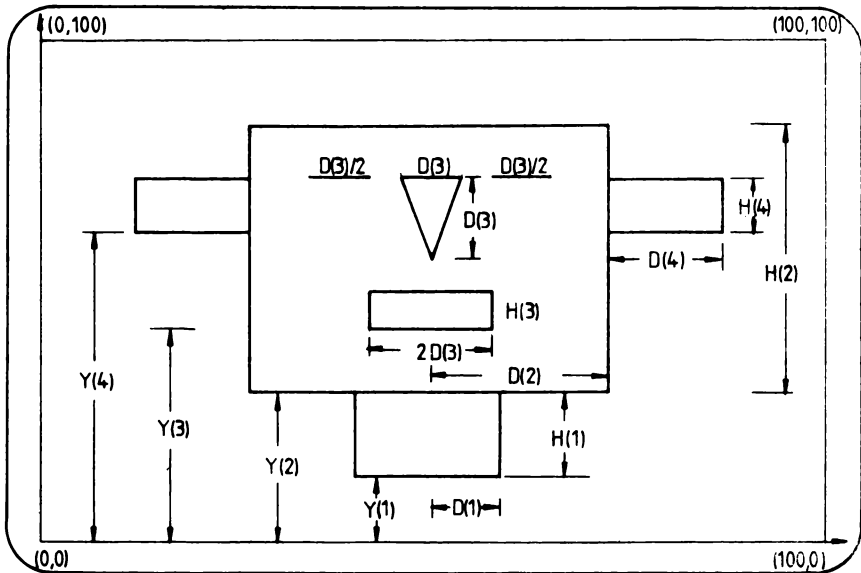


Fig. 12.15.

Și-acum, după ce ne-am amuzat puțin este momentul să ne apucăm de treabă: analiză, proiectare, codificare . . .

Tabela de variabile cuprinde *variabilele de intrare*: X (abscisa axei de simetrie a robotului), D(1), D(2), D(3) (v. figura 12.15), H(1), H(2), H(3), H(4) (v. figura 12.15), Y(1), Y(3), Y(4) (v. figura 12.15), *variabilele de stare*: S (vector de lucru), I, J (variabile de control) și *variabilele de ieșire* (Y și H).

Variabilele de intrare primesc în timpul execuției programului următoarele valori: X=50; D(1)=15; H(1)=20; Y(1)=20; D(2)=25; H(2)=40; D(3)=10; H(3)=5; Y(3)=50; H(4)=10; Y(4)=60.

Pseudocodul pentru desenarea roboțelului este prezentat în tabelul 12.1.

PSEUDOCODUL

```

ROBOT      SEQ
A          INPUT X
           PENTRU I DE LA 1 LA 4
             INPUT D(I)
             INPUT H(I)
A1         DACĂ I ≠ 2
           INPUT Y(I)
A1         SFIRȘIT
A          SFIRȘIT
           Y(2)=Y(1)+H(1)
           S(4)=D(2)
           Șterge ecran
B          PENTRU J DE LA 1 LA 3 PAS 2
B1         PENTRU I DE LA 1 LA 4
           PUNCT (x+(2-J) * S(I), Y(I))
           LINIE (x+(2-J) * (S(I)+D(I)), Y(I))
           LINIE (x+(2-J) * (S(I)+D(I)), Y(I)+H(I))
           LINIE (x+(2-J) * (S(I), Y(I)+H(I))
B1         SFIRȘIT
           PUNCT (X, Y(4)+H(4))
           LINIE (X+(2-j) * D(3)/2, Y(1)+H(4))
           LINIE (X, Y(4)+H(4)-D(3))
           PUNCT (X+D(1) * (2-j), Y(4)+H(4))
           RLINIE ((2-j) * D(3)/2, 0)
B          SFIRȘIT
ROBOT      END

```

Codificarea in limbajul BASIC-aMIC

Programul BASIC care desenează roboțelul proiectat de noi (v. fig. 12.15) are următoarea structură:

```

5 DIM S(4), Y(4), D(4), H(4)
7 PRINT "X=";
8 INPUT X
15 FOR I=1 TO 4
22 PRINT "D(" ; I ; ")=";
23 INPUT D(I)
24 PRINT "H(" ; I ; ")=";
25 INPUT H(I)
26 IF I=2 THEN 30
27 PRINT "Y (" ; I ; ")=";
30 NEXT I
35 Y(2)=Y(2)+H(1)
40 S(4)=D(2)
45 INIT
50 FOR J=1 TO 3 STEP 2
55 FOR I=1 TO 4
60 MOVE X+(2-J) * S(I), Y(I)
65 DRAW X+(2-J) * (S(I)+D(I)), Y(I)
70 DRAW X+(2-J) * (S(I)+D(I)),
  Y(I)+H(I)
75 DRAW X+(2-J) * S(I), Y(I)+H(I)
80 NEXT I
85 MOVE X, Y(4)+H(4)
90 DRAW X+(2-J) * D(3)/2,
  Y(4)+H(4)
95 DRAW X, Y(4)+H(4)-D(3)
100 MOVE X+D(1) * (2-J), Y(4)+H(4)
105 RDRAW (2-J) * D(3)/2, 0
110 NEXT j
115 STOP
120 END

```

După cum ați putut constata el conține trei elemente de noutate: instrucțiunile **MOVE** (v. liniile 60, 85, 100), **DRAW** (v. liniile 65, 70, 75, 90, 95) și **RDRAW** (v. linia 105). Cele trei instrucțiuni la care se vor mai adăuga și altele (**WINDOW**, **VIEWPORT**) formează setul minim de instrucțiuni gra-

fițe ale limbajului BASIC-aMIC. Ele codifică primitivele **PUNCT** respectiv **LINIE** și **RLINIE** cu care am făcut cunoștință nu cu mult timp în urmă. Revenind la program remarcați cum pentru $J=1$ se desenează partea stângă a roboțelului, după cum urmează (v. liniile 55–80): jumătatea stângă de gât ($I=1$), jumătatea stângă de cap ($I=2$), jumătatea dreaptă de gură ($I=3$), urechea stângă ($I=4$), jumătatea stângă de nas (v. liniile 85–95), ochiul stâng (v. liniile 100–105).

Pentru $j=2$, se desenează cu aceleași instrucțiuni partea dreaptă a roboțelului.

Vom analiza în cele ce urmează fiecare instrucțiune grafică în parte.

MOVE

Instrucțiunea **MOVE** este utilizată pentru poziționarea spotului display-ului în spațiul utilizator.

```
60 MOVE X+(2-J) * S(I), Y(I)
.
.
85 MOVE X, Y(4)+H(4)
.
.
100 MOVE X+D(1) * (2-J), Y(4)+H(4)
```

Formatul general al instrucțiunii este:

Format general
MOVE (X), (Y)

în care X și Y sînt coordonatele punctului de poziționare, exprimate în unități utilizator.

Remarci

- La calculatorul aMIC originea axelor ($X=Y=0$) se consideră în colțul din stînga jos.
- Display-ul are o suprafață de cel puțin 100/100 unități (UG).
- Instrucțiunea **MOVE** execută numai poziționarea în punctul de coordonate (X, Y) fără a marca punctul respectiv.
- Coordonatele (X, Y) pot fi constante, variabile sau expresii. Ele sînt raportate la originea sistemului.
- Instrucțiunea **RMOVE** (**MOVE** Relativ) al cărei format general este același consideră coordonatele (X, Y) raportate la poziția spotului (coordonate relative).

DRAW

Instrucțiunea **DRAW** este folosită pentru trasarea unei drepte între punctul în care se află spotul în momentul execuției instrucțiunii și punctul de coordonate (X, Y) specificat.

```
65 DRAW X+(2-J) * (S(I)+D(I)), Y(I)
70 DRAW X+(2-J) * (S(I)+D(I)), Y(I), Y(I)+H(I)
.
.
90 DRAW X+(2-J) * D(3)/2, Y(4)+H(4)
95 DRAW X, Y(4)+H(4)-D(3)
```

Formatul general al instrucțiunii este:

Format general
DRAW (X), (Y)

în care (X) și (Y) precizează capătul final al liniei. Ele pot fi constante, variabile sau expresii.

Observație. Instrucțiunea **DRAW** codifică în limbajul BASIC-aMIC primitiva **LINIE**.

Aplicație. Să se traseze graficul de producție al întreprinderii REMORCA pentru trimestrul II al anului în curs (v. figura 12.16).

Indicație. Se citesc în vectorul P valorile producției pe lunile martie, aprilie, mai, iunie: P(1), P(2), P(3), P(4). Se poziționează spotul în punctul de coordonate X=3, Y=P(1). În cadrul unei bucle **FOR-NEXT** se unesc printr-o dreaptă cele patru puncte ale graficului.

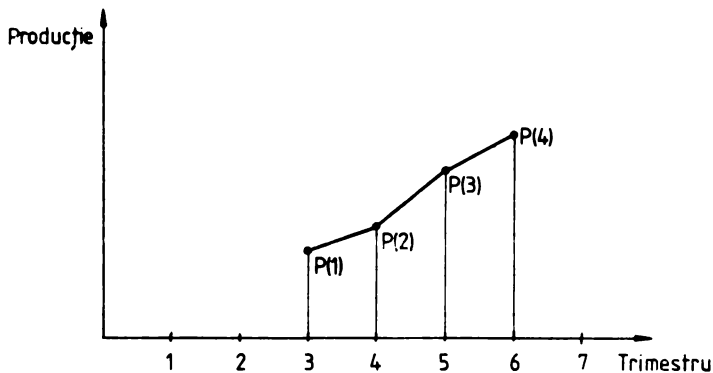


Fig. 12.16.

RDRAW

Instrucțiunea **RDRAW** (**DRAW** relativ) se deosebește de instrucțiunea **DRAW** prin faptul că coordonatele X, Y nu sînt raportate la originea sistemului, ci la poziția spotului la momentul execuției instrucțiunii.

105 **RDRAW** (2-J) * D(3)/2, 0

Formatul general al instrucțiunii este:

Format general
RDRAW (X), (Y)

Remarci

- Instrucțiunea **RDRAW** codifică în limbajul BASIC-aMIC primitiva **RLINIE**.
- Coordonatele (relative) X, Y pot fi constante, variabile sau expresii.
- Instrucțiunile **RMOVE** și **RDRAW** se folosesc în special pentru translația figurilor.

Aplicație. Să se traseze un pătrat cu latura de 10 UU avînd colțul din stînga jos în origine. Scrieți un program BASIC-aMIC care să translateze pătratul în diferite puncte din spațiul utilizatorului.

WINDOW și VIEWPORT

În limbajul BASIC-aMIC "spațiul utilizator" se declară cu instrucțiunea **WINDOW** al cărei format general este:

Format general
WINDOW <e1>, <e2>, <e3>, <e4>

Remarci

- <e1>, <e2>, <e3>, <e4> definesc un spațiu dreptunghiular (fereastră) care reprezintă limitele "spațiului utilizator", respectiv limita stîngă (e1), limita dreaptă (e2), limita inferioară (e3), limita superioară (e4). Ele pot fi constante, variabile, sau expresii.
- Instrucțiunea realizează automat trecerea de la puncte de coordonate utilizator la puncte ecran.
- Se recomandă ca laturile ferestrei declarate în instrucțiunea **WINDOW** să fie cît mai apropiate de domeniul de valori de reprezentat.
- Spațiul utilizator definit de **WINDOW** se plasează de obicei pe întreaga suprafață a ecranului.
- Instrucțiunea **WINDOW** se plasează la începutul programului (subprogramului) de reprezentare grafică.
- În absența instrucțiunii **WINDOW**, calculatorul aMIC stabilește fereastra implicit la valorile 0, 100, 0, 100.

TEST

Precizați rolul următoarelor instrucțiuni:

10 **WINDOW** – 30, 130, –30, 130

20 **WINDOW** – 6000, 6000, –1000, 1000

Nu puține sînt cazurile cînd dorim ca fereastra să fie proiectată numai pe o parte a suprafeței ecranului, cealaltă parte fiind necesară altor scopuri. În acest caz vom utiliza instrucțiunea **VIEWPORT**, al cărei format general este:

Format general
VIEWPORT <e1p>, <e2p>, <e3p>, <e4p>

unde, <e1p> – <e4p> reprezintă limitele fizice ale zonei din suprafața display-ului în aceeași ordine ca și la instrucțiunea **WINDOW**.

Remarci

- Unitatea de măsură în care se exprimă <e1p> – <e4p> s-a ales ca fiind un procent din latura pătratului cel mai mare, care poate fi înscris în suprafața display-ului. Este numită unitate grafică (UG).
- În absența instrucțiunii **VIEWPORT** calculatorul va executa implicit

10 **VIEWPORT** 0, 100, 0, 100

- Cu inițializările implicite

```
10 WINDOW 0, 100, 0, 100
20 VIEWPORT 0, 100, 0, 100
```

orice figură geometrică va apare nedeformată pe orice display, deoarece pe ambele axe de coordonate se folosește aceeași unitate de măsură (cele două instrucțiuni realizează o corespondență unu la unu între unitățile utilizator – pe orizontală și pe verticală și unitatea grafică – UG).

TEST

Precizați care este semnificația următoarelor instrucțiuni:

```
10 VIEWPORT 75, 100, 75, 100
20 VIEWPORT 0, 50, 0, 50
```

Codificarea în limbajul BASIC–PRAE

“Căutați” originea axelor de coordonate în colțul din stânga sus al ecranului. Axa Ox v-o puteți imagina ca pe o linie orizontală în partea de sus iar axa Oy ca pe o linie verticală în partea stângă a ecranului. Ecranul TV este împărțit în 256 de puncte pe orizontală și 256 de puncte pe verticală. Rezultă deci că valorile pot varia pe ambele axe de la 0 la 255.

Treziți PRAE-ul! După cum constatați, ecranul are culoarea albă. Dacă doriți un fond negru al ecranului, tastați

```
SWITCH 1 : CLS
```

Dacă considerați că ecranul s-a . . . întristat, tastați

```
SWITCH 0 : CLS
```

și ecranul va avea din nou fondul alb.

Observație. Pentru ștergerea ecranului folosiți una din metodele: a) instrucțiunea **CLS** (Clear Screen); b) apăsarea simultană a tastei **[SHIFT]** și **[S]**; instrucțiunea **PRINT CHR\$(0)**.

Pentru desenarea roboțelului pe ecranul televizorului, limbajul BASIC-PRAE vă pune la dispoziție instrucțiunile grafice **PLOT (PLOT)** și **DRAW (DRAW)**. Să le examinăm pe rând!

PLOT

Instrucțiunea **PLOT** servește la desenarea punctului (X, Y) pe ecran. Formatul general al instrucțiunii este:

Format general
PLOT (X), (Y)

unde (X) și (Y) reprezintă coordonatele punctului.

Remarci

- Punctul desenat cu **PLOT** are totdeauna culoarea opusă fondului ecranului.
- Instrucțiunea poate fi folosită și în modul imediat.

Revenind la roboțel, nu trebuie decât să înlocuiți pe **MOVE** (v. liniile 60, 85, 100) din programul BASIC-aMIC cu **PLOT**. Atenție la spațiul utilizator!

PLOT

PLOT (C de la clear) diferă de instrucțiunea de mai sus (**PLOT**) doar prin faptul că ea desenează cu puncte de culoarea ecranului. Formatul general al instrucțiunii este

Format general
PLOT (X), (Y)

unde (X), (Y) reprezintă coordonatele punctului de pe ecran.

Observație. Instrucțiunea **PLOT** poate fi folosită și în modul imediat.

DRAW

Cea de-a treia instrucțiune grafică, **DRAW** are drept efect desenarea unui segment de dreaptă care unește ultimul punct desenat pe ecran cu punctul de coordonate (X, Y).

Formatul general al instrucțiunii este:

Format general
DRAW (X), (Y)

Observații

- Pentru a fixa un capăt al segmentului de dreaptă se folosește (de obicei) instrucțiunea **PLOT**.
- Instrucțiunea **DRAW** poate fi folosită și în modul imediat.

TEST

Desenați o diagonală a ecranului

```
10 PLOT 0, 0
20 DRAW 255, 255
```

Aplicație. Introduceți și executați următoarele programe BASIC-PRAE []:

- | | |
|------------------|-------------------------|
| a) | 50 FOR I=0 TO 255 |
| 10 PLOT 20, 20 | 60 PLOT I, 128+10 * SIN |
| 20 DRAW 120, 20 | (I * 6.28/255) |
| 30 DRAW 120, 120 | 70 NEXT I |
| 40 DRAW 20, 120 | 80 FOR J=0 TO 255 |
| 50 DRAW 20, 20 | 90 PLOT J, 128+10 * SIN |
| b) | (J * 6.28/255) |
| 10 PLOT 0, 128 | 100 NEXT J |
| 20 DRAW 255, 128 | 110 SWITCH 1 |
| 30 PLOT 128, 255 | 120 CLS |
| 40 DRAW 128, 0 | |

DRAWC

DRAWC (C de la clear) diferă de instrucțiunea **DRAW** doar prin faptul că ea desenează cu puncte de culoarea ecranului.

Formatul general al instrucțiunii este:

Format general
DRAWC (X), (Y)

unde (X), (Y) reprezintă coordonatele punctului de pe ecran.

Remarcă. Instrucțiunea **DRAWC** poate fi folosită și în modul imediat.

Acum, după ce ați aflat majoritatea secretelor privind utilizarea instrucțiunilor grafice **PLOT** și **DRAW** puteți trece fără emoții la desăvârșirea robotului utilizând limbajul BASIC-PRAE. Atenție la modul cum folosiți instrucțiunea **DRAW**. Nu uitați să înlocuiți și linia 105 (v. programul BASIC-aMIC).

□ Să desenăm un cilindru cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM

Vă mai amintiți cât de prietenos este AMSTRADUL! Ca să nu mai vorbim de HC-85, TIM S sau SPECTRUM. Este momentul să le trezim și pe ele. Să le punem să deseneze un . . . cilindru, de exemplu. Da, da chiar un cilindru a cărei generatoare este înclinată la 45° față de orizontală. Nici nu bănuți cât de bine știu și ele să deseneze! Dar cum inteligența lor este funcție de . . . a noastră, nu ne mai rămîne decît să scriem programul care după cum veți vedea, nu este deloc complicat.

Ideea este următoarea. Se generează n cercuri de rază (r) constantă și se variază coordonatele centrului cercului (a, b) cu același pas (h), după cum urmează:

$$a_{k+1} = a_k + h \quad (9)$$

$$b_{k+1} = b_k + h \quad (10)$$

unde $n-1 \geq k \geq 1$, h avînd o valoare constantă.

Codificarea în limbajul BASIC-AMSTRAD

10 CLS	50 FOR i=0 TO 2 * PI STEP PI/60
20 READ a, b, r	60 DRAW a+r * COS(i),
24 DATA 50, 50, 50	b+r * SIN(i)
26 LOCATE 1,1 : PRINT "pasul este" ; :	70 NEXT i
INPUT h	72 a=a+h
27 CLS	75 b=b+h
28 n=50	80 NEXT k
30 FOR k=1 TO n	90 GOTO 90
40 MOVE a+r, b	

Programul inițializează coordonatele centrului primului cerc (v. liniile 20 și 24) fixînd pentru roză (r) valoarea 50. În ciclul **FOR-NEXT** (v. liniile

30–80) se desenează cilindrul propriu-zis după cum urmează: se trasează un cerc $C(a,b,r)$ (v. liniile 50–70) după care se modifică valorile lui a și b (v. liniile 72, 75) în conformitate cu relațiile (9) și (10). Bucla se repetă de 50 de ori (v. linia 28).

Este de datoria noastră să vă prezentăm mai întâi pe **LOCATE** (v. linia 26) și după aceea să dăm binețe unor cunoștințe e drept mai vechi, întilnite însă în alte limbaje, **MOVE** și **DRAW**.

LOCATE

Instrucțiunea **LOCATE** plasează cursorul de text într-un anumit punct al ecranului ale cărui coordonate se raportează la colțul din stînga sus (1,1) al ecranului.

Observație. În **Mode 1** există 40 de coloane și 25 de linii, în **Mode 0** nu există decît 20 de coloane dar tot 25 de linii, iar în **Mode 2** sînt 80 de coloane și 25 de linii.

Revenind la instrucțiunea **LOCATE** din program

```
26 LOCATE 1, 1 -- PRINT "pasul este"; : INPUT h
```

veți vedea în momentul execuției programului cum în colțul din stînga sus al ecranului de coordonate (1,1) în care prima cifră (1) reprezintă coordonata orizontală (cuprinsă între 1 și 40 – v. **Mode 1**), iar cea de-a doua cifră (1) reprezintă coordonata verticală (cuprinsă între 1 și 25) se afișează mesajul „pasul este” urmat de semnul întrebării (în așteptarea introducerii dinamice a valorii pasului h).

Formatul general al instrucțiunii este:

Format general
LOCATE [# <număr canal>, <X>, <Y>

în care <număr canal> se consideră # \emptyset , dacă nu îl precizați.

MOVE

Instrucțiunea **MOVE** poziționează cursorul grafic în punctul de coordonate absolute specificat, fără a marca (fizic) punctul (pixel-ul) respectiv

```
40 MOVE a+r, b
```

Formatul general al instrucțiunii este:

Format general
MOVE <X>, <Y> [,[<cerneală>] [,<tip cerneală>]

în care parametrul facultativ <cerneală> (cu valori cuprinse între 0 și 15) permite schimbarea culorii stiloului grafic. Cît privește celălalt parametru opțional <mod> (cu valori între 0 și 4–0 : normal, 1–**XOR**, 2–**AND**, 3–**OR**) el determină interacțiunea cernelii pe afișajul de pe ecran.

Remarci

- Cursorul grafic, diferit de cursorul caracter nu este vizibil.
- Coordonatele X și Y sînt definite în raport cu colțul inferior din stînga ecranului de coordonate (0,0).
- Rezoluția ecranului este de 640 (puncte orizontale)×400 (puncte verticale).
- Spre deosebire de instrucțiunea **LOCATE** aplicată caracterelor, aceste coordonate sînt aceleași în toate cele trei moduri de afișaj (**Mode** 0, 1 sau 2).
- Pentru poziționarea cursorului grafic în puncte de coordonate relative se utilizează instrucțiunea **MOVER** (v. vol. 2, pag. 39).
- Pentru afișarea în mod grafic a unui punct de coordonate (X, Y) se utilizează instrucțiunea **PLOT** (v. vol. II pag. xxx). Spre deosebire de instrucțiunea **MOVE**, **PLOT** trasează pixel-ul (punctul) în coordonatele precizate.

DRAW

Instrucțiunea **DRAW** desenează o linie pe ecran între poziția cursorului grafic și o poziție absolută specificată prin coordonatele X și Y.

```
40 MOVE a+r, b
50 FOR i=0 TO 2 * PI STEP PI/60
60 DRAW a+r * COS(i), b+r * SIN(i)
70 NEXT i
```

Remarcă. Pentru trasarea cercului am generat arce mici de $\frac{\pi}{60}$ radiani localizate prin puncte M_i de coordonate $(a+r \cos(i * \alpha), b+r \sin(i * \alpha))$. Segmentele care unesc aceste puncte aproximează cercul $C(a, b, r)$, idee care a stat la baza desenării lui cu ajutorul instrucțiunilor **MOVE** și **DRAW**.

Formatul general al instrucțiunii **DRAW** este

Format general
DRAW (X), (Y) [,{cerneală}] [,{tip cerneală}]

Remarci

- Parametrii instrucțiunii au semnificația celor întilniți la instrucțiunea **MOVE**.
- Pentru trasarea unei drepte utilizînd coordonatele relative se folosește instrucțiunea **DRAWR** (v. vol. 2, pag. 21).

Acum, după ce ați înțeles programul vă invităm să-l tastați și apoi să-l executați. Ați obținut un cilindru a cărui generatoare este înclinată la 45° față de orizontală?

Așa cum a fost scris, programul ciclează (v. linia 90) pînă cînd îl veți întrerupe dvs. tastînd de două ori **[ESC]**. De notat că instrucțiunea

```
90 GO TO 90
```

evită întreruperea automată a programului după ultima linie și afișarea mesajului **Ready**.

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

La HC-85, TIM S, SPECTRUM „căutați” punctul de coordonate (0, 0) în colțul din stînga jos al ecranului grafic (v. figura 12.17).

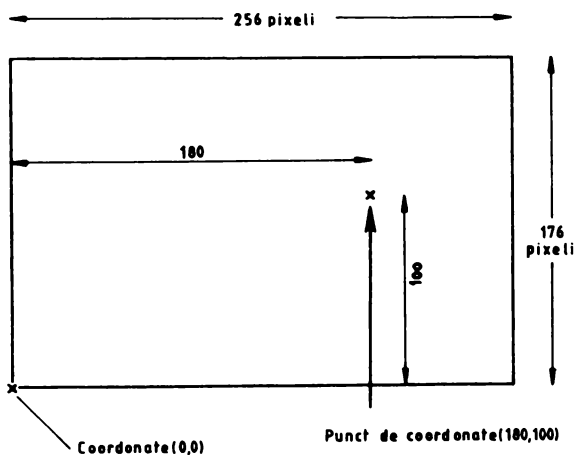


Fig. 12.17.

A nu se confunda cu punctul de coordonate (0,0) din colțul din stînga sus, al **ecranului alfanumeric** pe care l-am întilnit în momentul utilizării instrucțiunii **PRINT AT**. Vă reamintim că **ecranul alfanumeric** conține 24 de linii (liniile 22 și 23 sînt rezervate mesajelor de sistem) a cîte 32 coloane (caractere) fiecare, caracterul de adresă alfanumerică (0,0) fiind în colțul din stînga sus al ecranului. HC-85, TIM S și SPECTRUM permit însă și o înaltă rezoluție. Cum fiecare caracter alfanumeric este reprezentat pe o matrice de 8×8 pixeli rezultă că partea din ecran utilizabilă conține $(32 \times 8) \times (22 \times 8)$ pixeli adresabili. Așadar, pixelul de adresă (0,0) se află în colțul din stînga jos al **ecranului grafic**.

Imaginați-vă ecranul ca pe o casă cu mai multe nivele. Mergi la parter (X) și apoi urci la etaj (Y) (v. reprezentarea punctului de coordonate 180 și 100 din fig. 12.17).

Să ne reîntoarcem la problema noastră (trasarea unui cilindru a cărei generatoare este înclinată la 45° față de orizontală) pe care vrem s-o rezolvăm în BASIC HC-85, TIM S, SPECTRUM. Există mai multe posibilități. Fie cu instrucțiunile **PLOT** și **DRAW** fie direct cu instrucțiunea **CIRCLE**.

În BASIC HC-85, TIM S, SPECTRUM instrucțiunea **PLOT** mută cursorul grafic din poziția curentă la poziția de coordonate (X,Y) și marchează (aprinde) punctul (pixel-ul) respectiv.

Formatul general al instrucțiunii este

Format general
PLOT <X>, <Y>

în care <m> reprezintă una din instrucțiunile **OVER** sau **INVERSE**; <X>, <Y> reprezintă coordonatele punctului.

Observație. Instrucțiunea **PLOT INVERSE 1; OVER 1**; lasă pixel-ul neschimbat iar cursorul grafic rămîne poziționat în punctul de coordonate (X, Y).

Dacă dorim să desenăm o linie lungă, devine plictisitor să folosim instrucțiunea **PLOT** (în cazul cînd dreapta se trasează punct cu punct). Pu-

tem folosi în schimb instrucțiunea grafică **DRAW** al cărei format general este

Format general
DRAW $\langle X \rangle$, $\langle Y \rangle$, $\langle Z \rangle$

în care $\langle X \rangle$, $\langle Y \rangle$ reprezintă punctul final al liniei, primul punct fiind definit de poziția cursorului grafic; $\langle Z \rangle$ reprezintă numărul de radiani corespunzător circumferinței. Dacă $\langle Z \rangle$ este pozitiv curba se trasează în sens invers mișcării acelor de ceasornic. Pentru $Z=0$ se trasează o linie dreaptă între cele două puncte.

După cum ați remarcat putem utiliza instrucțiunea **DRAW** pentru a desena cercuri (problema noastră) dar este destul de complicat și nu merită efortul, deoarece instrucțiunea **CIRCLE** este mult mai comodă și, de ce să nu recunoaștem mai elegantă.

CIRCLE

Instrucțiunea **CIRCLE** trasează un cerc cu centrul în punctul (pixel-ul) de coordonate (x, y) de rază z pixeli. O vom utiliza în programul nostru.

```

10 CLS
20 READ a, b, r
24 DATA 50, 50, 50
26 PRINT AT 1, 1; "pasul este"; :
   INPUT h
27 CLS
28 LET n=50
30 FOR k=1 TO n
40 CIRCLE a, b, r
72 LET a=a+h
75 LET b=b+h
80 NEXT k
90 GOTO 90

```

Diferențele de scriere a programului apar numai în liniile 26 și 40. Cit privește cea de-a doua instrucțiune

```
40 CIRCLE a, b, r
```

ea înlocuiește instrucțiunile

```

40 MOVE a+r, b
50 FOR i=0 TO 2 * PI STEP PI/60
60 DRAW a+r * COS (i), b+r * SIN (i)
70 NEXT i

```

din programul AMSTRAD.

Instrucțiunea **CIRCLE** din linia 40 trasează 50 de cercuri (v. fig. 12.18) de rază constantă (v. linia 20) avînd însă coordonatele centrului variabile (v. liniile 72 și 75)

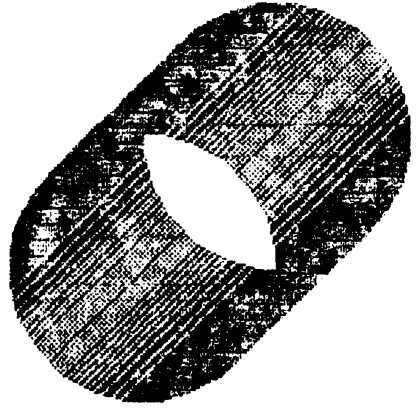
Formatul general al instrucțiunii este:

Format general
CIRCLE $\langle X \rangle$, $\langle Y \rangle$, $\langle Z \rangle$

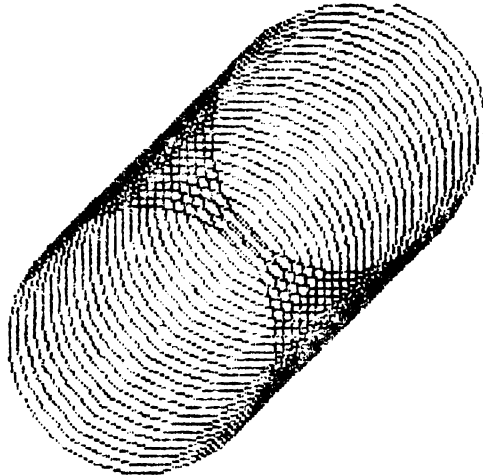
în care, $\langle X \rangle$, $\langle Y \rangle$ reprezintă coordonatele centrului;
 $\langle R \rangle$ este raza cercului.



a)



b)



c)

Fig. 12.18. Cilindru: a) pasul 2; b) pasul 1; c) pasul 3.

Remarci

- Instrucțiunea **CIRCLE** codifică în limbajul BASIC HC-85, TIM S, SPECTRUM primitiva **CERC**.
- Ca și instrucțiunile **PLOT** și **DRAW** instrucțiunea **CIRCLE** admite comenzi de modificare a culorii.

Aplicație. Introduceți și executați următorul program:

```
10 REM cercuri concentrice
20 FOR r=0 TO 60 STEP 5
30   CIRCLE 100, 75, r
40 NEXT r
50 STOP
```

□ Culori și grafice

Programarea culorilor pe calculatoarele personale HC-85, TIM S, SPECTRUM

Calculatoarele HC-85, TIM S și SPECTRUM dispun de multiple posibilități privind utilizarea culorilor și realizarea graficelor. Prin programarea culorilor, dumneavoastră puteți controla trei elemente: bordura ecranului (instrucțiunea **BORDER**), ecranul TV (instrucțiunea **PAPER**) și culoarea utilizată pentru scriere. Sînt disponibile opt culori (dacă includem și culorile alb, negru): 0 – negru, 1 – albastru, 2 – roșu, 3 – purpură (magenta), 4 – verde, 5 – albastru deschis, 6 – galben, 7 – alb. Pe un televizor alb/negru, în locul culorilor apar nuanțe gri, cifrele de cod (0–7) fiind ordonate după strălucirea nuanței de gri.

Să încercăm să utilizăm instrucțiunile privind folosirea culorilor tastînd și executînd următoarele programe:

a)

```
10 FOR i=0 TO 7
20   BORDER i
30   PAUSE 50
40 NEXT i
50 STOP
```

Observație. Instrucțiunea **PAUSE** (v. linia 30) oprește execuția programului menținînd activ display-ul pe durata a 50 perioade de baleiaj ale ecranului (20 MS pentru fiecare ecran).

b)

```
10 FOR i=0 TO 7
20   PAPER i
30   PRINT "
40 NEXT i
50 STOP
```

c)

```
10 FOR i=1 TO 10
20   FOR j=0 TO 7
30     PAPER j
40     PRINT " "
50   NEXT j
60 NEXT i
70 STOP
```

d)

```
10 REM * Demonstrație color *
20 FOR B=0 TO 7
30   FOR P=0 TO 7
40     FOR I=0 TO 7
50       BORDER B
60       PAPER P: CLS
70       INK I
80       PRINT AT 10, 10; "BORDER"; B;
          TAB 10; "PAPER"; P; TAB 10;
          "INK"; I
90       FOR U=1 TO 60 : NEXT U : BEEP
          .15, P * I + 2 * B
100      NEXT I
110     NEXT P
120    NEXT B
```

Observație. Instrucțiunea **BEEP** (v. linia 90) este utilizată pentru producerea sune-
telor.

e)

```
10 REM * cercuri concentrice *
20 FOR i=0 TO 60 STEP 5
30 CIRCLE 100, 75, r
35 INK INT (RND * 7)
40 NEXT i
50 STOP
```

Observație. Distribuția mai puțin obișnuită a culorii se datorează faptului că instruc-
țiunea **INK** este adresabilă numai la nivel de caracter, nu și de pixel.

f)

```
10 REM * Tricolorul belgian *
20 BORDER 7
30 CLS
40 FOR a=0 TO 175
50 PLOT 0, a
60 INK 0
70 DRAW 85, 0
80 INK 6
90 DRAW 85, 0
100 INK 2
110 DRAW 85, 0
120 NEXT a
```

```
60 PLOT a, b : PLOT a, d-b
70 PLOT c-a, b: PLOT c-a, d-b
80 IF RND > .5 THEN GOTO 60
90 INK RND 7
100 GOTO 50
```

j)

```
10 BORDER 2 : CLS
20 FOR x=0 TO 63 STEP .5
30 LET y=20 * SIN (x/32 * PI)
40 IF y=0 THEN GOTO 100
50 FOR n=0 TO y STEP SGN y/4
60 PLOT INK RND * 6; x * 3+30,
3 * (n+30)
```

g)

```
10 REM * Tricolorul german *
20 BORDER 7
30 CLS
40 FOR a=0 TO 58
50 INK 6
60 PLOT 0, a
70 DRAW 255, 0
80 NEXT a
90 FOR a=59 TO 117
100 INK 2
110 PLOT 0, a
120 DRAW 255, 0
130 NEXT a
140 FOR a=118 TO 175
150 INK 0
160 PLOT 0, a
170 DRAW 255, a
180 NEXT a
```

```
70 NEXT n
100 BEEP .1, x : NEXT x
```

k)

```
10 PAPER 7
20 LET a=INT (RND * 8)
30 LET b=INT (RND * 7)
40 IF a=b THEN GOTO 60
50 BORDER a
60 INK b
70 CLS
80 LET c=INT (RND * 256)-128
90 LET d=INT (RND * 172)-85
100 PLOT 128, 86
110 DRAW c, d
120 BEEP .01, RND * 100-50
130 IF RND > 0.02 THEN GOTO 110
140 RUN
```

h)

```
10 REM * Tricolorul italian *
20 BORDER 0
30 CLS
40 FOR a=0 TO 175
50 PLOT, 0, a
60 INK 4
70 DRAW 85, 0
80 INK 7
90 DRAW 85, 0
100 INK 2
110 DRAW 85, 0
120 NEXT a
130 STOP
```

l)

```
10 PAPER 7
20 LET a=INT (RND * 8)
30 LET b=INT (RND * 7)
40 IF a=b THEN GOTO 60
50 BORDER a
60 INK b
70 CLS
80 LET c=INT (RND * 256)-128
90 LET d=INT (RND * 172)-85
100 PLOT 128, 86
110 DRAW c, d, PI/2
120 BEEP .01, RND * 100-50
130 IF RND > 0.015 THEN GOTO 110
140 RUN
```

i)

```
10 PAPER 0: BORDER 0; CLS
20 LET C=255 : LET d=175
30 INK RND * 7
40 LET a=c * RND
50 LET b=d * RND
```

m)

```
10 PAPER 7
20 LET a=INT (RND * 8)
30 LET b=INT (RND * 7)
40 IF a=b THEN GOTO 60
50 BORDER a
60 INK b
```

```

70 CLS
80 LET c=INT (RND * 256)-128
90 LET d=INT (RND * 172)-85
100 PLOT 128, 86
110 IF RND > 0.5 THEN INK RND * 6
120 DRAW c, d, PI/2
130 BEEP .01, RND * 100-50
140 IF RND > 0.015 THEN GOTO 110
150 RUN

```

n)

```

10 BORDER 5: PAPER 7: CLS
20 CIRCLE INK RND * 6;
   128+RND * 10-RND * 10,
   85+RND * 7-RND * 7, RND * 65
30 IF RND > .92 THEN CLS
40 BEEP RND/3, RND * 100-30
50 GOTO 20

```

o)

```

10 PAPER 7: CLS
20 INK 0
30 BORDER 7
40 FOR a=1 TO 10
50 FOR J=1 TO 7 STEP .1
60 PLOT a * J * COS (J)+110,
   a * J * SIN(J)+100

```

```

70 NEXT J
80 INK a/2
90 NEXT a

```

p)

```

10 PAPER 7: CLS
20 INK 0
30 BORDER 7
40 FOR a=5 TO 19
50 FOR J=1 TO 7 STEP .3
60 PLOT a * J * COS (J)/2+110,
   a * J * SIN (J)+100
70 NEXT J
80 NEXT a

```

r)

```

10 PAPER 7: CLS
20 INK 0
30 BORDER 7
40 FOR a=2 TO 19
50 FOR J=1 TO 7 STEP .01
60 PLOT a * (J+4 * SIN (J)) *
   COS (J)+110, a * (J+2 * SIN
   (J)) * SIN (J)+100
70 NEXT J
80 INK a/3.5
90 NEXT a

```

Programarea culorilor pe AMSTRAD

În BASIC-AMSTRAD aveți de ales între 27 de culori! În **Mode 0** puteți avea în același timp pe ecran 16 culori, în **Mode 1-4** culori iar în **Mode 2** numai 2 culori. Pentru programarea culorilor utilizați instrucțiunile **BORDER, PAPER, PEN, INK** al căror format general este prezentat în volumul 2, pag. 14, 44, 30.

Aplicație. Introduceți și executați următorul program:

```

10 MODE 1 : INK 2, 10 : INK 3, 4
20 ORIGIN 440, 100, 440, 640, 100, 300
30 WINDOW 1, 26, 1, 25
40 CLG 2 : GRAPHICS PAPER 0
50 DRAW 200, 200, 3
60 MOVE 2,0 : FILL 3
70 ORIGIN 440, 0, 440, 640, 0, 400
80 GRAPHICS PEN, 1 : MASK, 0
90 FOR y=60 TO 318 STEP 2
100 GOSUB 220
110 FRAME : FRAME
120 GOSUB 220
130 NEXT
140 TAG
150 FOR y=60 TO 318 STEP 2
160 MOVE 96, y : PRINT CHR$ (224) ;
170 FRAME : FRAME
180 MOVE 96, y : PRINT CHR$ (224) ;
190 NEXT
200 END
210 MOVE 90, y, 1
220 DRAWR 20, 0 , , 1
230 DRAWR 0,20
240 DRAWR -20, 0
250 DRAWR 0, -20
260 RETURN

```

Observație. Pentru înțelegerea instrucțiunilor folosite de acest program consultați volumul 2, paginile 43, 28, 58.

Aplicație. Introduceți și executați următoarele programe:

a)

```

20 ON BREAK GOSUB 220
30 FOR i=1 TO 15 : INK i, 26 : NEXT
40 m(1)=1 : m(2)=2 : m(3)=4 : m(4)=8
50 MODE 0 : PRINT CHR$(23); CHR$(3); : TAG

```

```

60 FOR p=1 TO 4
70 GRAPHICS PEN m(p), 1
80 LOCATE #1, 1, 25 : PRINT #1, CHR$(48+p);
90 FOR x=0 TO 7
100 FOR y=0 TO 14 STEP 2
110 IF TEST (x * 4, y)=0 THEN 140
120 MOVE (x+6) * 32, (y+6) * 16 : PRINT CHR$(143);
130 MOVE (x+6) * 32, (y+7) * 16 : PRINT CHR$(143);
140 NEXT y, x, p
150 LOCATE #1, 1, 25 : PRINT #1, " " ;
160 FOR p=1 TO 4
170 FOR i=1 TO 25 : FRAME : NEXT
180 FOR i=0 TO 15
190 IF (i AND m(p))=0 THEN INK i, 0 ELSE INK i, 26
200 NEXT i, p
210 GOTO 160
220 INK 1, 26

```

RUN

```

b) 20 DEFINT a-z
30 INK 0,1 : INK 1,26
40 INK 2,6 : INK 3,6
50 FOR i=4 TO 7 : INK i, 9 : NEXT
60 FOR i=8 TO 15 : INK i, 20; NEXT
70 MODE 0 : DEG : ORIGIN 0,150 : CLG : MOVE 0,150
80 FOR x=16 TO 640 STEP 16
90 DRAW x, COS (x) * 150+RND * 100,4
100 NEXT
110 MOVE 0,0 : FILL 4
120 cx=175 : GOSUB 320
130 cx=525 : GOSUB 320
140 SYMBOL 252, 0, 0, &C, &1F, &30, &7F, &FF
150 SYMBOL 253, 0, 6, &E, &F2, 2, &F2, &FF
160 SYMBOL 254, 0, &60, &70, &7F, &7F, &7F, &7F
170 SYMBOL 255, 0, 0, 0, &F8, &EC, &FE, &FF
180 pr$=CHR$(254)+CHR$(255)
190 pl$=CHR$(252)+CHR$(253)
200 TAG : t ! = TIME
210 FOR x=-32 TO 640 STEP 4
220 x2=((608-x) * 2) MOD 640 : h1=RND * 10 : hr=50 * SIN(x)
230 GRAPHICS PEN 8, 1 : MOVE x, 100+hr, , 3 : PRINT pr$ ;
240 GRAPHICS PEN 2, 1 : MOVE x2, 115+h1, , 3 : PRINT pl$ ;
250 IF (TEST (x2-2, 115+h1-12) AND 8)=8 THEN 380
260 IF TIME-t ! < 30 THEN 260
270 FRAME : t ! = TIME
280 GRAPHICS PEN 7, 1 : MOVE x, 100+hr, , 2 : PRINT pr$ ;
290 GRAPHICS PEN 13, 1 : MOVE x2, 115+h1, , 2 : PRINT pl$ ;
300 NEXT
310 GOTO 210
320 MOVE cx, 100
330 FOR x=0 TO 360 STEP 10
340 DRAW cx+SIN(x) * 50+10 * RND, 100+COS(x) * 25+10 * RND, 1
350 NEXT
360 DRAW cx, 100 : MOVE cx, 90 : FILL 1
370 RETURN
380 ENT -1, 1, 1, 1
390 SOUND 1, 25, 400, 15, , , 1, 15
400 FOR y=100+hr TO -132 STEP -2
410 GRAPHICS PEN 7, 1 : MOVE x, y, , 2 : PRINT pr$ ;
420 GRAPHICS PEN 8, 1 : MOVE x, y-2, , 3 : PRINT pr$ ;
430 NEXT
440 GOTO 70

```

RUN

□ **Generarea unui rezervor-display cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM.
Trasarea histogramelor cu ajutorul caracterelor mozaic**

Să nu credeți că am uitat de problema rezervoarelor cu benzină care ne-a însoțit pe tot parcursul lucrării. Nu poate fi vorba de așa ceva. Am abandonat-o numai pentru o clipă spre a vă putea prezenta facilitățile grafice de care dispun calculatoarele cu care am lucrat. O întrebare. Vă mai amintiți de listele cu livrările de benzină, generate în conversațiile precedente? Din păcate acestea nu sînt de mare ajutor atunci cînd este vorba de interpretarea rezultatelor. În foarte multe cazuri sîntem interesați de modul cum se schimbă valorile și nu de precizia obținerii lor.

O metodă eficientă, pe care v-o propunem în această conversație, constă în vizualizarea rezultatelor folosind un *grafic* sau o *histogramă*. Astfel de imagini sînt mult mai sugestive chiar dacă de multe ori precizia are de suferit.

Să începem prin a genera din caractere mozaic un *rezervor-display* căruia să-i alăturăm șapte histograme reprezentînd cantitățile de benzină livrate în fiecare zi a săptămîinii din acel rezervor.

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 262

Caractere mozaic

Pentru a „produce” *rezervorul-display* vom utiliza simbolurile grafice mozaic ale calculatorului AMSTRAD, după cum urmează:

Baza *rezervorului-display* o vom construi prin tipărirea a trei caractere mozaic avînd codurile 130, 131 și 129. Peretii îi vom realiza prin tipărirea a două caractere mozaic cu codurile 138 (pentru peretele din stînga) și 133 (pentru peretele din dreapta). În sfîrșit, capacul *rezervorului-display* îl vom „pune” cu codurile 136, 140 și 132.

Putem reprezenta cantitatea totală de combustibil livrată prin desenaarea unei linii verticale a cărei lungime este proporțională cu valoarea livrărilor efectuate în cursul unei săptămîni. Înălțimea *rezervorului-display* va fi suficientă pentru a permite coloanei să atingă valoarea maximă. În vederea citirii cantităților livrate vom desena o scară chiar lîngă linia măsurătoare. Cit privește desenaarea histogramelor (pentru cele șapte zile) vom utiliza, de asemenea, caractere semigrafice (pline și semipline).

Specificațiile de programare și documentația de proiectare sînt ilustrate în modulele de analiză și proiectare structurată, figura 12.19.

Acum, este momentul să vă prezentăm lista programului (v. vol. 2, pag. 262), pe care vă invităm să-l tastați și după aceea să-l și executați.

După ce în linia 140 s-a inițializat abscisa de unde va începe trasarea *rezervorului-display*, în liniile următoare (150–170) se generează *rezervorul-display* propriu-zis. Scalarea *rezervorului-display* se realizează cu secvența de instrucțiuni 175–190, iar afișarea inițialelor (L, M, . . . , D) zilele săptămîinii se face cu instrucțiunile 192–196. Vom introduce dinamic (v. li-

TABELA DE VARIABILE

Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Variabile de intrare	Variabile de stare	Variabile de ieșire
t: cantitatea (totală) de combustibil livrată	z, i, n, j, y: variabile de lucru t: cantitatea de combustibil livrată zilnic	t: cantitatea de combustibil livrată

a)

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Descrierea programului

Programul desenează cu ajutorul caracterelor mozaic un rezervor-display și o familie de histograme ce reprezintă cantitatea de combustibil (benzină) livrată în cursul unei săptămâni (defalcată pe fiecare zi în parte).

Intrări

Se introduce de la tastatură cantitatea de combustibil livrată într-o săptămână.

Ieșiri

Un rezervor-display și șapte histograme.

Lista de funcțiuni ale programului

1. Inițializare abscisă generare rezervor-display.
2. Reprezentare bază.
3. Reprezentare corp.
4. Reprezentare capac.
5. Scalare rezervor-display.
6. Afișare inițiale zile săptămână.
7. Citire cantitate totală de combustibil livrată.
8. Citire cantitate de combustibil livrată zilnic.
9. Calcul număr întreg de unități de măsură prin care se reprezintă o cantitate t de benzină livrată.
10. Reprezentare printr-un număr de Y/2 caractere "pline" (CHR\$(143)).
11. Reprezentare printr-un număr de (y-1)/2 caractere "pline" plus un caracter "semiplin" (CHR\$(140)).

b)

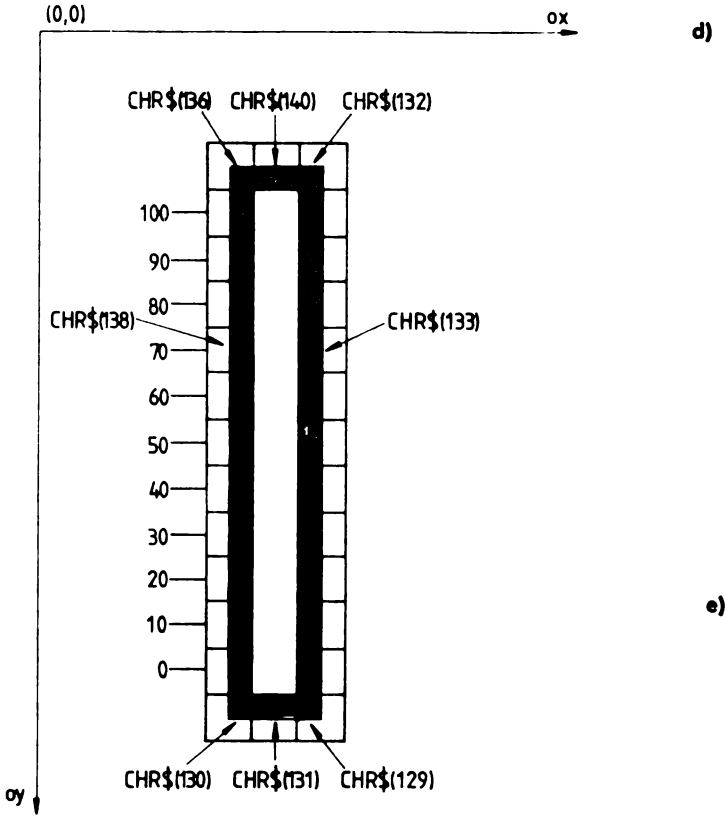
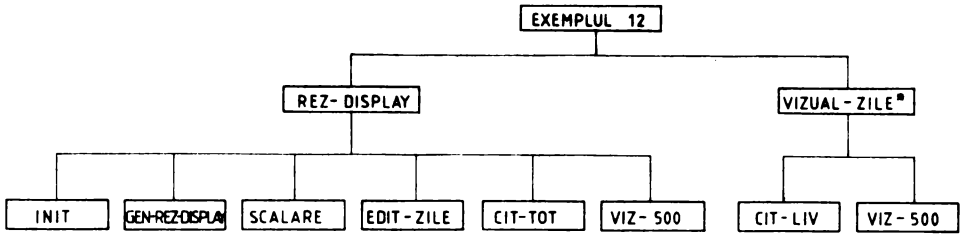
ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Modul	Funcțiuni
INIT	1
GEN-REZ-DISPLAY	2, 3, 4
SCALARE	5
EDIT-ZILE	6
CIT-TOT	7
CIT-LIV	8
VIZ-500	9, 10, 11

c)

Fig. 12.19. Modulele de analiză și proiectare structurată: a) tabela de variabile a programelor BASIC-AMSTRAD și BASIC HC-85, TIM S, SPECTRUM; b) specificații de programare ale programelor BASIC-AMSTRAD și BASIC HC-85, TIM S, SPECTRUM; c) alocarea funcțiunilor de prelucrare;



PSEUDOCODUL

Nume program: EXEMPLUL 12 - PC (AMSTRAD)

```

MOZAMS  SEQ
        CLS
        Z=8
        I=0
        LOCATE Z, 20
        PRINT CHR$(130); CHR$(131); CHR$(129)
    
```

Fig. 12.19 f)

Fig. 12.19. d) diagrama de structură a programelor BASIC-AMSTRAD și BASIC HC-85, TIM S, SPECTRUM; e) rezervorul-display AMSTRAD; f) pseudocodul programului BASIC-AMSTRAD.

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (AMSTRAD)

```

A      PENTRU N DE LA 1 LA 11
        LOCATE Z-1, 20-N
        PRINT "-" CHR$(138); CHR$(128); CHR$(133)
A      SFÎRȘIT
        LOCATE Z, 8
        PRINT CHR$(136); CHR$(140); CHR$(132)
B      PENTRU N DE LA 1 LA 11
        LOCATE (Z-6), 20-N
        PRINT I
        I=I+10
B      SFÎRȘIT
C      PENTRU J DE LA Z+1 LA 38
        LOCATE J, 20
        PRINT CHR$(131)
C      SFÎRȘIT
        LOCATE 15, 22
        PRINT "L M M J V S D"
        LOCATE 1, 1
        PRINT "Cantitatea de combustibil rezervor"
        INPUT T
        LOCATE 1, 1
        PRINT "
        DO 500
D      PENTRU Z DE LA 14 LA 32 PAS 3
        READ T
        DO 500
D      SFÎRȘIT
MOZAMS END
VIZ-500 SEQ
A      PENTRU N DE LA 1 LA 11
        LOCATE Z+1, 20-N
        PRINT CHR$(128)
A      SFÎRȘIT
        Y=INT((T+5)/5+0.5)
B      PENTRU N DE LA 1 LA INT (Y/2)
        LOCATE Z+1, 20-N
        PRINT CHR$(143)
B      SFÎRȘIT
C      DACĂ INT (Y/2) <> Y/2
        Y=INT (Y/2)
        LOCATE Z+1, 19-Y
        PRINT CHR$(140)
C      SFÎRȘIT
VIZ-500 END

```

Fig. 12.19.f)

niile 205–206) cantitatea totală de combustibil livrată în cursul unei săptămîni după care se vizualizează cu subrutina 500. Cele șapte histograme reprezentînd livrările zilnice de combustibil se tipăresc cu instrucțiunile 210–215.

Să nu părăsim programul înainte de-a înțelege bine modul în care a fost realizată subrutina de vizualizare livrări (500).

Histograme

În instrucțiunea 530 cantitatea de combustibil (benzină) ce urmează a fi vizualizată se consideră $t+5$ avînd în vedere că originea (punctul 0) se află la o „înălțime” de 5 unități de măsură față de baza *rezervorului-display* (înălțimea unui caracter se consideră egală cu 10 unități de măsură – v. figura 12.19.e). Cantitatea $Y = \text{INT}((t+5)/5 + 0.5)$ semnifică numărul întreg de cîte 5 unități de măsură prin care se reprezintă o cantitate de combustibil t .

Exemplele următoare de conversie (v. tabelul 12.2) în jumătăți de caracter (5 unități de măsură) vă vor ajuta să înțelegeți mai bine mecanismul de reprezentare.

Tabelul 12.2

t	y
0	$y = \text{int}((0+5)/5+0.5) = \text{int}(1.+0.5) = \text{int}(1.5) = 1$
2	$y = \text{int}((7)/5+0.5) = \text{int}(1,4+0,5) = \text{int}(1,1) = 1$
3	$y = \text{int}(8/5+0,5) = \text{int}(1,6+0,5) = \text{int}(2,1) = 2$
5	$y = \text{int}(10/5+0,5) = \text{int}(2+0,5) = \text{int}(2,5) = 2$
7	$y = \text{int}(12/5+0,5) = \text{int}(2,4+0,5) = \text{int}(2,9) = 2$
8	$y = \text{int}(13/5+0,5) = \text{int}(2,6+0,5) = \text{int}(3,1) = 3$
12	$y = \text{int}(17/5+0,5) = \text{int}(3,4+0,5) = \text{int}(3,9) = 3$

Remarcați o rotunjire a valorii cantităților de combustibil livrate în jumătăți de caracter, după regula: **o cantitate t se convertește în același număr de jumătăți de caracter ca și cea mai apropiată cantitate divizibilă cu 5.**

Astfel, cantitățile $t=73$, $t=74$, $t=75$, $t=76$, $t=77$ vor avea aceeași reprezentare grafică. Analog, cantitățile $t=78$, $t=79$, $t=80$, $t=81$, $t=82$. Rezultă, după cum ați putut constata o anumită imprecizie. Deoarece în reprezentarea cantităților nu putem folosi decît caracterul „plin” – CHR\$(143) sau caracterul „semiplin” – CHR\$(140) apare ca o necesitate utilizarea instrucțiunilor 530–565. Ele au următoarea semnificație: în cazul cînd numărul Y rezultat prin conversia lui t este par, reprezentarea acestei cantități se va face printr-un număr de $Y/2$ caractere „pline” – CHR\$(143), în caz contrar reprezentarea făcîndu-se printr-un număr de $(Y-1)/2$ caractere pline plus un caracter semiplin.

TEST

Modificați programul anterior astfel încît să se afișeze (cu ajutorul caracterelor mozaic) cantitățile de combustibil (defalcate pe zile) din trei rezervoare. Cele trei cantități se vor introduce de la tastatură.

```

200 FOR d=1 TO 3
205 LOCATE 1,1 : PRINT "Cantitatea de combustibil rezervor"; d
206 INPUT t
.
.
.
215 NEXT Z
300 NEXT d

```

Aplicație. Introduceți și executați următorul program:

```

10 MODE 1
20 x=20 : y=12
22 c=249
30 LOCATE x, y : PRINT CHR$ (c)
40 a$=INKEY$
45 IF ASC (a$) < 240 OR ASC (a$) > 243 THEN 40
50 IF a$=CHR$(240) THEN yadd=-1 : xadd=0 : c=248
60 IF a$=CHR$(241) THEN yadd=+1 : xadd=0 : c=249
70 IF a$=CHR$(242) THEN xadd=-1 : yadd=0 : c=251
80 IF a$=CHR$(243) THEN xadd=+1 : yadd=0 : c=250
85 q1dx=x : q1dy=y
90 x=x+xadd : y=y+yadd
100 IF x>40 THEN x=40
110 IF x<1 THEN x=1
120 IF y>25 THEN y=25
130 IF y<1 THEN y=1
135 LOCATE q1dx, q1dy : PRINT " "
140 GO TO 30

```

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

Caracterele mozaic cu ajutorul cărora puteți construi atît *rezervorul-display* cît și cele șapte histograme sînt ilustrate în figura 12.20.

















	128		136
	129		137
	130		138
	131		139
	132		140
	133		141
	134		142
	135		143

Fig. 12.20.

Aplicație. Introduceți și executați următorul program. Comparați rezultatele cu fig. 12.20.

```

10 CLS
20 PRINT
30 FOR i=32 TO 164
40 PRINT CHR$ i ; " " ;
50 NEXT i
60 STOP

```

Odată cunoscute: setul caracterelor mozaic, rezoluția ecranului, instrucțiunile semigrafice nu vă rămîine decît să adaptați programul de pe AMSTRAD pe calculatoarele HC-85, TIM S, SPECTRUM. Luați această invitație ca pe o provocare la programare!

Aplicație. Introduceți și executați următoarele programe:

```

a)
10 DIM a(7)
20 DIM c(7)
30 FOR s=1 TO 30
40 CLS
50 FOR p=1 TO 7
60 LET a(p)=122+INT
   (RND * 22)
70 LET c(p)=INT (7 * RND)
80 NEXT p
90 LET k=3+INT (5 * RND)
100 FOR n=1 TO 672 STEP k
110 FOR j=1 TO k
120 PRINT INK c(j); CHR$ a(j)
130 NEXT j
140 NEXT n
150 PAUSE 200
160 NEXT s

120 LET c=15
130 FOR j=1 TO 4
140 FOR i=1 TO 4
150 PRINT AT r+j-1, c+i-1;
   CHR$ a(i, j);
160 NEXT i
170 NEXT j
180 STOP

b)
10 DIM a(4,4)
20 FOR j=1 TO 4
30 FOR i=1 TO 4
40 READ a(i, j)
50 NEXT i
60 NEXT j
70 DATA 128, 133, 143, 128
80 DATA 132, 140, 142, 140
90 DATA 128, 132, 142, 128
100 DATA 128, 138, 128, 138
110 LET r=10

c)
10 DIM a(4,4)
20 FOR j=1 TO 4
30 FOR i=1 TO 4
40 READ a(i, j)
50 NEXT i
60 NEXT j
70 DATA 128, 133, 143, 128
80 DATA 132, 140, 142, 140
90 DATA 128, 132, 142, 128
100 DATA 128, 138, 128, 138
110 FOR r=0 TO 16 STEP 5
120 FOR c=0 TO 28 STEP 4
130 FOR j=1 TO 4
140 FOR i=1 TO 4
150 PRINT AT r+j-1, c+i-1;
   CHR$ a(i, j);
160 NEXT i
170 NEXT j
180 NEXT c
190 NEXT r
200 STOP

```

□ **Generarea a trei rezervoare-display cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM. Trasarea histogramelor prin metoda "punct cu punct"**

Programul a fost conceput (v. modulele de analiză și proiectare structurată, fig. 12.21) să deseneze, în prima fază, trei rezervoare-display (v. figura 12.22) gradate din 10 în 10, de la 0 la 50 (tone combustibil). În cea de-a doua fază am programat conversația propriu-zisă. După ce comunicați

TABELA DE VARIABILE

Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Variabile de intrare	Variabile de stare	Variabile de ieșire
t: cantitatea de combustibil (in tone) din rezervor	d: număr rezervor x, y: coordonate poziționare cursor i, k, q, r, y, x, p, j, q, l: variabile de lucru	t: cantitatea de combustibil (in tone) din rezervor

a)

Fig. 12.21. a) tabela de variabile;

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLELE 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Descrierea programului

Programul desenează pe ecranul monitorului trei rezervoare-display. Fiecare rezervor-display ilustrează cantitatea de combustibil (în tone) livrată la un moment dat. Modul de construire a imaginilor este "punct cu punct".

Intrări

Se introduc de la tastatură cantitățile (în tone) de combustibil livrate (pentru trei rezervoare).

Ieșiri

Trei rezervoare-display.

Lista de funcțiuni ale programului

1. Inițializare x, y
2. Curățire ecran
3. Poziționare cursor grafic
4. Trasare cadru rezervor-display
5. Trasare bază rezervor-display
6. Gradare rezervor-display
7. Citire dinamică valoare cantitate combustibil rezervor
8. Reprezentare parte activă rezervor-display

b)

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 12 – PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Modul	Funcțiuni
INIT	1, 2
CADRU-HIST	3, 4
RA-340	5, 8
GRAD-HIST	6
CIT-CANT	7

c)

PSEUDOCODUL

Nume program: EXEMPLUL 12-PC (AMSTRAD)

PUNCT	SEQ
	READ Y
	CLS
A	PENTRU X DE LA 100 LA 630 PAS 200
	MOVE X, Y
	DRAWR 70, 0
	DRAWR 0, 300
	DRAWR -70, 0
	DRAWR 0, -300
	Q=50
	K=0
	DO 340
A1	PENTRU I DE LA Y+50 LA Y+250 PAS 8
	MOVE X, I
	R=(I-Y-50)/40

d)

Fig. 12.21. b) specificații de programare; c) alocarea funcțiilor de prelucrare; d) pseudocodul pentru programul BASIC-AMSTRAD;

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (AMSTRAD)

```

A2          DACA INT (R)=R
           DRAWR -20, 0
           LOCATE (X-70)/16, (400-I)/16
           PRINT K
           K=K+10
A2          IN CAZ CONTRAR
           DRAWR -10, 0
A2          SFIRȘIT
A1          SFIRȘIT
A           SFIRȘIT
           Q1=30
B           PENTRU X DE LA 100 LA 630 PAS 200
           D=D+1
           LOCATE 1, 1
           PRINT "CANTITATEA IN TONE REZERVOR" D
           INPUT T
           P=50
           Q=T*4
           DO 340
           LOCATE 1, 1
           PRINT "
B           SFIRȘIT
PUNCT      END
RA-340     SEQ
           MOVE X, Y+P
A           PENTRU I DE LA Y+P LA Y+P-1+Q
A1         PENTRU J DE LA X-1+Q1 LA X+70-Q1
           DRAWR 0, 1
           MOVE J+1, I
A1         SFIRȘIT
A           SFIRȘIT
RA-340     DRAWR 0, 1
           END

```

d)

PSEUDOCODUL

Nume program: EXEMPLUL 12-PC (HC-85, TIM S, SPECTRUM)

```

HCPUNCT    SEQ
           READ Y
           CLS
           P=0
           Q1=0
A           PENTRU X DE LA 40 LA 200 PAS 80
           PLOT X, Y
           DRAW 30, 0
           DRAW 0, 130
           DRAW -30, 0
           DRAW 0, -130
           Q=20
           K=0
           Q1=28
           DO 340

```

e)

Fig. 12.21. e) pseudocodul pentru programele BASIC HC-85, TIM S, SPECTRUM;

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (HC-85, TIM S, SPECTRUM)

```

A1          PENTRU I DE LA Y+20 LA Y+120 PAS 4
            PLOT X, I
            R=(I-Y-20)/20
A2          DACA INT(R)=R
            DRAW -20, 0
            PRINT AT (175-I)/8, (X-30)/8; K
            K=K+10
A2          IN CAZ CONTRAR
            DRAW -10, 0
A2          SFIRȘIT
A1          SFIRȘIT
A          SFIRȘIT
            Q1=2
            D=0
B          PENTRU X DE LA 40 LA 200 PAS 80
            D=D+1
            PRINT AT 1, 1; "cantitatea in tone rezervor"; d
            INPUT t
            P=20
            Q=T * 2
            DO 340
            PRINT AT 1, 1; "

B          SFIRȘIT
HCPUNCT    END
340        SEQ
A          PLOT X, Y+P
A1         PENTRU I DE LA Y+P LA Y+P-1+Q
            PENTRU J DE LA X+(30-Q1)/2 LA X+(30-Q1)/2+Q1
            DRAW 0, 1
            PLOT J, I
A1         SFIRȘIT
A          SFIRȘIT
340        DRAW 0, 1
            END
    
```

e)

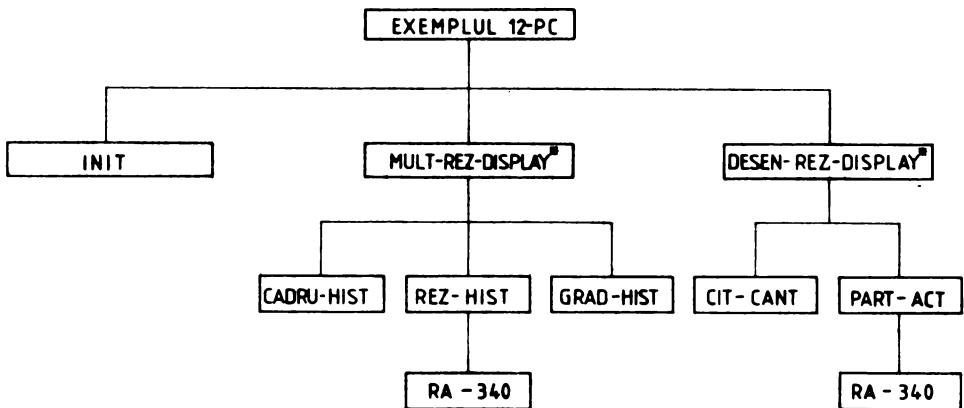


Fig. 12.21. f) diagrama de structură.

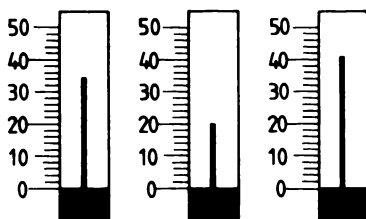


Fig. 12.22. Rezervoare-display generate prin metoda punct cu punct.

calculatorului, pentru fiecare rezervor în parte, cantitățile de combustibil livrate puteți admira pe ecranul monitorului histogramele corespunzătoare, trasate prin metoda „punct cu punct”.

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 263

Lista programului se află în vol. 2, pag. 263. Înainte de a-l tasta, vă sugerăm să-l analizați cu multă atenție.

- 40–240 Multiplică pe ecranul monitorului trei rezervoare-display gradate de la 0–50 (tone combustibil)
- 50– 90 Desenează cadrul rezervorului-display
- 120 Trimite la subrutina 340 care pentru $q=50$, $p=0$, $q1=0$ desenează baza rezervorului-display
- 130–230 Gradează rezervorul-display. Gradarea se face din 5 în 5. Numărul de tone este vizualizat din 10 în 10 (v. linia 190); instrucțiunea 170 utilizează formula de trecere de la sistemul de coordonate pentru **DRAW** la sistemul de coordonate pentru **LOCATE**.
- 260–320 Desenează „partea activă” a rezervorului-display corespunzătoare celor trei cantități de combustibil livrate ale căror valori se introduc de la tastatură (v. linia 280).
- 340–420 Subrutina construiește „punct cu punct” cu ajutorul a două bucle **FOR-NEXT** (350–400) și (360–390) un dreptunghi de lungime.
 $L=x+70-q1-x-1-q1=69-2q1$ puncte, și de lățime
 $l=y+p-1+q-y-p=q-1$ puncte.
 Un punct este desenat cu ajutorul instrucțiunilor 370, 380. Variabilele q și $q1$ reglează lățimea și lungimea acestui dreptunghi, fapt ce permite atât desenarea bazei rezervorului-display cât și reprezentarea „părții active” a acestuia.

Observație. Datcrită modulul de lucru „punct cu punct” viteza de reprezentare este mică.

TEST

Simulați cu creionul și hirtia mod I de funcționare a subrutinei 340 de trasare „punct cu punct” a părții active a rezervorului-display (v. liniile 290, 300, 310 din programul principal).

```
340 MOVE x, y+p
350 FOR i=y+p TO y+p-1+q
```

```

360 FOR j=x-1+q1 TO x+70-q1
370   DRAW 0,1
380   MOVE J+1, i
390 NEXT j
400 NEXT i
410 DRAW 0,1
420 RETURN

```

Aplicație. Introduceți și executați următorul program:

```

10 MODE 0 : BORDER 13
20 MOVE 0, 200 : DRAW 640, 200
30 FOR x=80 TO 560 STEP 80
40 MOVE x, 0 : DRAW x, 400
50 NEXT : MOVE - 40, 300
60 FOR C=0 TO 7
70 MOVER 80, 0 : FILL c
80 MOVER 0, - 200 : FILL c+8
90 MOVER 0, 200 : NEXT
100 GO TO 100

```

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 266

Pentru scrierea programului am avut în vedere atât rezoluția ecranului cit și setul de instrucțiuni grafice disponibile pe HC-85, TIM S și SPECTRUM. Am procedat prin a revedea specificațiile de programare ale exemplului precedent după care am întocmit o nouă documentație de proiectare (v. pseudocodul din fig. 12.21.e). Lista programului este prezentată în vol. 2, pag. 266. Vă rugăm să lecturați în paralel cele două programe, ca apoi, singuri, fără ajutorul nostru să descoperiți... diferențele. Atenție la parametrii care se transmit subrutinei 340!

TEST

În urma execuției uneia din versiunile programului BASIC HC-85, TIM S, SPECTRUM de mai sus s-au obținut histogramele din figura 12.23. Precizați cărei instrucțiuni i se datorează această imagine... neterminată!

R.

380 PLOT j+1, i

în loc de

380 PLOT j, i

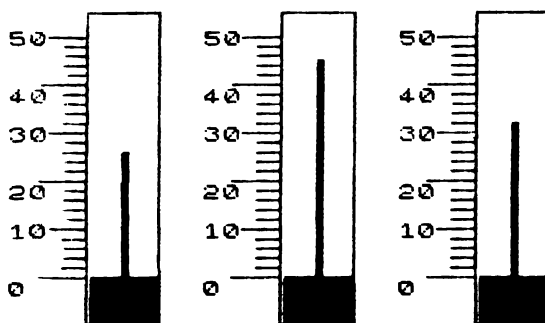


Fig. 12.23. Histograme cu ... greșeli.

□ **Trasarea histogramelor în 2D prin metoda
„linie cu linie” cu calculatoarele
AMSTRAD și HC-85, TIM S, SPECTRUM**

Vom relua problema din exemplul precedent complicînd-o după cum urmează.

Pentru trei rezervoare, se citesc de la tastatură cantitățile de combustibil livrate zilnic beneficiarilor. Se dorește ca, pentru fiecare rezervor în parte, să se traseze, printr-o metodă mai rapidă – linie cu linie, o histogramă reprezentînd cantitățile de combustibil livrate în fiecare din cele șase zile lucrătoare ale săptămîinii.

Specificațiile de programare și documentația de proiectare sînt ilustrate în modulele de analiză și proiectare structurată, figura 12.24.

Nume program: EXEMPLELE 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)		
Variabile de intrare	Variabile de stare	Variabile de ieșire
	t : livrări benzină, pe zile d : număr rezervor x, y : coordonate vîrf k, i, r, q1, q, p : variabile de lucru	t : livrări benzină

a)

SPECIFICAȚII DE PROGRAMARE	
Nume program: EXEMPLELE 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)	
Descrierea programului	Ieșiri
<p>Programul afișează succesiv trei familii de histograme (alcătuite din linii), fiecare familie reprezentînd cantitățile de benzină livrate dintr-un anumit rezervor în fiecare din cele șase zile (lucrătoare) ale săptămîinii.</p> <p>Intrări</p> <p>Se citesc (de la tastatură), pentru fiecare rezervor în parte, livrările zilnice de benzină efectuate.</p>	<p>Trei histograme</p> <p>Lista de funcțiuni ale programului</p> <ol style="list-style-type: none"> 1. Inițializare origine sistem axe xoy 2. Trasare axe cu săgeți în vîrf 3. Gradare axă oy 4. Tipărire zile săptămîină (sub axa ox) în clar 5. Citire livrări combustibil, pe zile 6. Acționare orice tastă 7. Construire histogramă din linii

b)

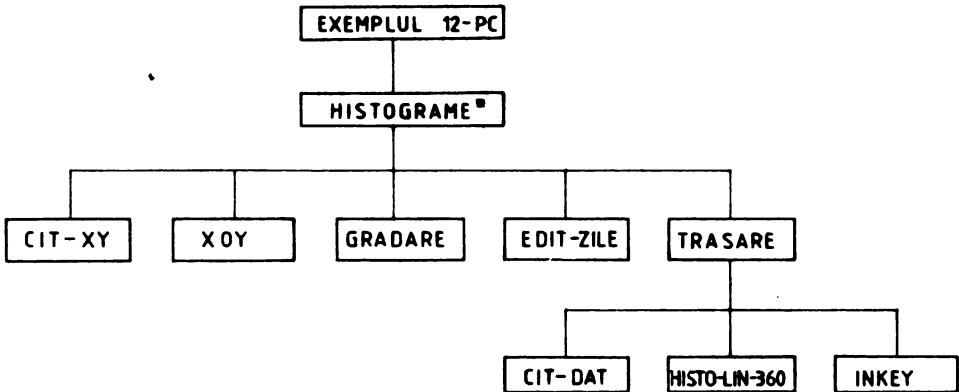
Fig. 12.24. Modulele de analiză și proiectare structurată: a) tabela de variabile; b) specificații de programare;

ALOCAREA FUNCȚIUNILOR DE PRELUCRARE

Nume program: EXEMPLELE 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Modul	Funcțiuni
CIT - XY	1
XOY	2
GRADARE	3
EDIT - ZILE	4
CIT - DAT	5
HISTO-LIN-360	7
INKEY	6

c)



d)

PSEUDOCODUL

Nume program: EXEMPLUL 12-PC (AMSTRAD)

```

2DHISTA  SEQ
A        READ X, Y
        PENTRU D DE LA 1 LA 3
        CLS
        K=0
        X=80
        MOVE X, 350
        DRAW X, 100
        DRAW 640, 100
        MOVE X -10, 340
        DRAW X, 350
        .DRAW X +10, 340
        MOVE 630, Y+10
        DRAW 640, Y
        DRAW 630, Y-10
    
```

e)

Fig. 12.24. c) alocarea funcțiilor de prelucrare; d) diagrama de structură; e) pseudocodul programului BASIC-AMSTRAD.

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (AMSTRAD)

```

A1      PENTRU I DE LA Y LA Y+200 PAS 8
        MOVE X, I
        R=(I-Y)/40
A2      DACA INT(R)=R
        DRAWR -20, 0
        LOCATE (X-70)/16, (400-I)/16
        PRINT K
        K=K+10
A2      IN CAZ CONTRAR
        DRAWR -10, 0
A2      SFIRȘIT
A1      SFIRȘIT
        Q1=10
        LOCATE
        PRINT "LIVRĂRI BENZINA REZERVOR "D" PE ZILE"
        LOCATE (X+16)/16, 22
        PRINT "Luni, Marți, Miercuri, Joi, Vineri, Simbătă"
A3      PENTRU X DE LA 80 LA 620 PAS 95
        READ T
        Q=T*4
        DO 360
A3      SFIRȘIT
A4      CIT TIMP INKEY$=" "
A4      SFIRȘIT
A       SFIRȘIT
2DHISTA  END
360     SEQ
        MOVE X, Y+P
A       PENTRU I DE LA Y+P LA Y+P-1+Q
        MOVE X-1+Q1, I
        DRAW X+70-Q1, I
A       SFIRȘIT
360     END

```

Fig. 12.24. e)

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 264

Lista programului este prezentată în volumul 2, pag. 264). De urmărit în cadrul programului: trasarea (cu **MOVE** și **DRAW**) sistemului de coordonate xoy (v. liniile 70–116); scalarea axei oy (v. liniile 120–220); tipărirea (sub axa ox) numelor zilelor săptămînii (v. liniile 240–250); introducerea dinamică a livrărilor (v. liniile 260–330); acționarea oricărei taste în vederea trasării histogramei pentru rezervorul următor (v. linia 335); construirea propriu-zisă din linii, a histogramelor (v. liniile 360–440).

Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 267

Diferențele de scriere a programului apar la utilizarea setului de instrucțiuni grafice (**PLOT**, **DRAW**) specifice calculatoarelor personale HC-85, TIM S și SPECTRUM.

După ce consultați specificațiile de programare și documentația de proiectare a programului încercați să continuați, (singuri) cu codificarea programului. Dacă nu reușiți consultați lista programului din vol. 2, pag. 267.

În figurile 12.25, 12.26 și 12.27 sînt ilustrate histogramele realizate „linie cu linie” pentru livrările de benzină din cele trei rezervoare.

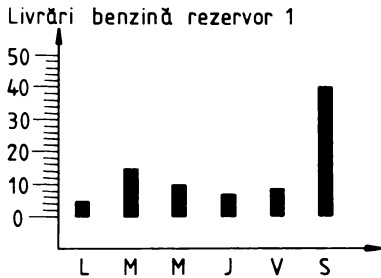


Fig. 12.25.

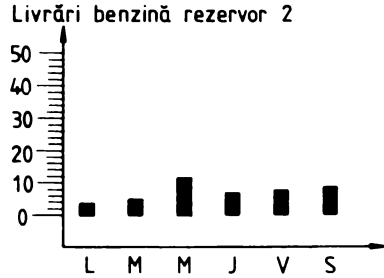


Fig. 12.26.

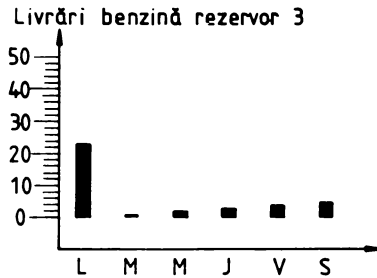


Fig. 12.27.

Aplicație. Introduceți și executați următoarele programe:

```

a) 5 BORDER 7 : PAPER 7 : CLS : IN
    K 0
    10 LET x=INT (RND*256)
    20 LET y=INT (RND*176)
    30 LET l=INT (RND*256)
    40 LET m=INT (RND*176)
    50 LET u=15 : LET v=7
    60 DEF FN r(x)=INT (RND * x)
    100 GO SUB 1000
    115 LET num=num-1
    116 IF num=0 THEN GO SUB 1000
    120 PLOT x, y
    130 DRAW 1-x, m-y
    140 IF x+a>255 OR x+a<0 THEN
        LET a=-a
    150 IF y+b>175 OR y+b<0 THEN
        LET b=-b
    160 IF l+c>255 OR l+c<0 THEN
        LET c=-c
    170 IF m+d>175 OR m+d<0 THEN
        LET d=-d
    180 LET x=x+a : LET y=y+b
    
```

```

    190 LET l=l+c : LET m=m+d
    200 LET cont=FN r(200)
    220 IF cont=l THEN RUN
    230 GO TO 115
    1000 LET a=FNr(u)-v
    1010 LET b=FNr(u)-v
    1020 LET c=FNr(u)-v
    1030 LET d=FNr(u)-v
    1040 LET num=FNr(20)+10
    1050 RETURN
    
```

```

b) 10 READ y
    20 DATA 20
    30 CLS
    35 LET p=0; LET q1=0
    40 FOR x=40 TO 200 STEP 80
    50 PLOT x, y
    60 DRAW 30,0
    70 DRAW 0,130
    80 DRAW -30,0
    90 DRAW 0,-130
    100 LET q=20
    
```

```

110 LET k=0
115 LET q1=28
120 GO SUB 340
130 FOR i=y+20 TO y+120 STEP 4
140 PLOT x, i
150 LET r=(i-y-20)/20
160 IF INT (r) < > r THEN GO TO 22
170 DRAW -20,0
180 PRINT AT (175-i)/8, (x-30)/8
200 LET k=k+10
210 GO TO 230
220 DRAW -10,0
230 NEXT i
240 NEXT x
250 LET q1=8
255 LET d=0
260 FOR x=40 TO 200 STEP 80
262 LET d=d+1
265 PRINT AT 1,1; "cantitatea in tone
rezervor"; d;
280 INPUT t
290 LET p=20
300 LET q=t*2
310 GO SUB 340
315 PRINT AT 1,1;"
320 NEXT x "
325 GO TO 325
330 STOP
340 PLOT x, y+p
350 FOR i=y+p TO y+p-1+q
360 PLOT x+(30-q1)/2, i
370 DRAW q1, 0
400 NEXT i
420 RETURN

```

□ Trasarea histogramelor în 3D cu calculatoarele AMSTRAD și HC-85, TIM S, SPECTRUM

Vom relua problema livrărilor de benzină din exemplul precedent propunându-ne de această dată construirea de histograme spațiale pentru vizualizarea și interpretarea livrărilor zilnice.

Specificațiile de programare și documentația de proiectare sînt ilustrate în modulele de analiză și proiectare structurată figura 12.28.

SPECIFICAȚII DE PROGRAMARE

Nume program: EXEMPLUL 12-PC (AMSTRAD, HC-85, TIM S, SPECTRUM)

Descrierea programului

Programul afișează succesiv trei familii de histograme spațiale (alcătuite din linii), fiecare familie reprezentînd cantitățile de benzină livrate dintr-un anumit rezervor în fiecare din cele șase zile (lucrătoare) ale săptămîinii.

Intrări

Se citesc (de la tastatură), pentru fiecare rezervor în parte, livrările zilnice de benzină efectuate.

Ieșiri

Trei histograme spațiale.

Lista de funcțiuni ale programului

1. Inițializare origine sistem axe xoy
 2. Trasare axe cu săgeți în virf
 3. Gradare axă Oy
 4. Tipărire nume zile săptămîină (sub axa Ox) în clar
 5. Citire livrări combustibil, pe zile
 6. Acționare orice tastă
 7. Trasare linie orizontală de lățime 50 puncte
 8. Trasare linie înclinată la 45° față de linia orizontală
 9. Traducere linie frîntă pe o direcție paralelă cu Oy
 10. Completare ultima linie frîntă trasată pînă la un paralelogram.
-

a)

Fig. 12.28. Modulele de analiză și proiectare structurată: a) specificații de programare;

PSEUDOCODUL

Nume program: EXEMPLUL 12-PC (HC-85, TIMS, SPECTRUM)

```

HC30      SEQ
A         READ X, Y
          PENTRU D DE LA 1 LA 3
          X=40
          CLS
          K=0
          P=0
          Q1=0
          PLOT X, Y
          DRAW 200, 0
          DRAW -5, 5
          PLOT X+200, Y
          DRAW -5, -5
          PLOT X, Y
          DRAW 0, 137
          DRAW -5, -5
          PLOT X, 137+Y
          DRAW 5, -5
          Q=20
          K=0
          Q1=28
A1        PENTRU I DE LA Y+20 LA Y+120 PAS 4
          PLOT X, I
          R=(I-Y)/20
B         DACA INT(R)=R
          DRAW -20, 0
          PRINT AT (175-I)/8, (X-30)/8; K
          K=K+10
B         IN CAZ CONTRAR
          DRAW -10, 0
B         SFIRȘIT
A1        SFIRȘIT
          Q1=8
          PRINT AT 1, 1; "LIVRĂRI BENZINĂ REZERVOR";
          PRINT AT (175-Y)/8+1, X/8; "
          L M M J V S "
C         PENTRU X DE LA 40 LA 190 PAS 30
          READ T
          P=20
          Q=T*2
          DO 340
C         SFIRȘIT
D         CIT TIMP INKEY$=" "
D         SFIRȘIT
HC3D     END
340      SEQ
A         PENTRU I DE LA Y+P LA Y+P-1+Q
          PLOT X+(30-Q1)/2, I
          DRAW Q1, 0
          DRAW 5, 5
A         SFIRȘIT
          DRAW -Q1, 0
          DRAW -5, -5
340      END

```

b)

Fig. 12.28 b) pseudocodul programului BASIC HC-85, TIM S, SPECTRUM.

PSEUDOCODUL

Nume program: EXEMPLUL 12 – PC (AMSTRAD)

```

3D          SEQ
A          READ X, Y
          PENTRU D DE LA 1 LA 3
          CLS
          K=0
          X=80
          MOVE X, 350
          DRAW X, 100
          DRAW 640, 100
          MOVE X-10, 340
          DRAW X, 350
          DRAW X+10, 340
          MOVE 630, Y+10
          DRAW 640, Y
          DRAW 630, Y-10
A1         PENTRU I DE LA Y LA Y+200 PAS 8
          MOVE X, I
          R=(I-Y)/40
A2         DACĂ INT(R)=R
          DRAWR -20, 0
          LOCATE (X-70)/16, (400-I)/16
          PRINT K
          K=K+10
A2         IN CAZ CONTRAR
          DRAWR -10, 0
A2         SFIRȘIT
A1         SFIRȘIT
          Q1=10
          LOCATE, 1, 1
          PRINT "LIVRĂRI BENZINĂ REZERVOR "D" PE ZILE"
          LOCATE (X+16)/16, 22
          PRINT "Luni Marți Mierc. Joi Vineri Simb."
B          PENTRU X DE LA 80 LA 620 PAS 95
          READ T
          Q=T * 4
          DO 360
          DRAW X+Q1+9, I+9
          DRAW X-1+Q1, I-1
B          SFIRȘIT
C          CIT TIMP INKEY$=""
C          SFIRȘIT
A          SFIRȘIT
3D         END
360        SEQ
          MOVE X, Y+P
A          PENTRU I DE LA Y+P LA Y+P-1+Q
          MOVE X-1+Q1, I
          DRAW X+70-Q1, I
          DRAW X+70-Q1+10, I+10
A          SFIRȘIT
360        END

```

c)

Fig. 12.28. c)

Codificarea în limbajul BASIC-AMSTRAD

vol. 2, pag. 265

Lista programului se găsește în volumul 2, pag. 265.

De urmărit în cadrul programului:

- 10– 20 inițializare origine sistem axe
- 70–116 trasare axe (cu săgeți în vîrf)
- 120–220 construirea unei scări de măsură
- 260–330 generare histograme în 2D pentru cele trei rezervoare
- 360–440 generare histogramă spațială (3D)

Scalarea

Deși problema v-a mai fost prezentată (niciodată însă cu explicațiile noastre) considerăm că este util să aducem câteva precizări.

Cunoașteți rezoluția AMSTRAD-ului! 640×400 puncte cu punctul de coordonate (0, 0) în colțul din stînga jos al ecranului. Totodată, ecranul mai poate fi privit și ca un sistem $x'o'y'$ (punctul o' de coordonate (0, 0) se află în colțul din stînga sus) mărimea ecranului fiind de 40×25 caractere. Rezultă că, un caracter este alcătuit din $(640/40) \times (400/25) = 16 \times 16$ pixeli (puncte). Pe baza acestor considerente putem stabili următoarele formule de trecere de la un sistem de coordonate la altul.

$$y' \cdot 16 = 400 - y \quad (11)$$

$$x' \cdot 16 = x \quad (12)$$

de unde, rezultă

$$y' = \frac{400 - y}{16} \quad (13)$$

$$x' = \frac{x}{16} \quad (14)$$

Relațiile (13) și (14) justifică utilizarea instrucțiunilor din liniile 180 și 190.

Cît privește reprezentarea livrărilor de combustibil am făcut următoarele ipoteze: a) cantitatea maximă de combustibil ce poate fi reprezentată este de 50 tone; b) pentru o tonă de combustibil folosim un spațiu de patru puncte ecran (v. linia 310). Deci, pentru a reprezenta 10 tone avem nevoie de un spațiu de 40 puncte (v. ciclul de instrucțiuni din liniile 120–220).

Cu instrucțiunea

150 IF INT (r) <> r THEN 210

selectăm punctele a căror diviziune este multiplu de 10 (exemplu: 0, 10, 20, 30, 40, 50) semnalate printr-o liniuță mai lungă (v. linia 160) față de cele care semnalează cantitățile de combustibil divizibile cu 2 (v. linia 210).

Observație. Deoarece scalarea s-a realizat din două în două tone ciclul **FOR-NEXT** din liniile 120–220 are pasul 8 (o tonă se reprezintă prin 4 puncte).

Histograma spațială

Cit privește subrutina 360 aducem următoarele precizări suplimentare. Ea reprezintă varianta unei subrutine cunoscute de desenare a histogramelor. De notat că în acest program paramentul p nu este inițializat în mod explicit, deci implicit va lua valoarea zero.

Instrucțiunile din liniile 380–390 trasează o linie de lățime $x+70-q_1-x+1-q_1=70-2q_1$ puncte situate la o înălțime de i puncte de baza ecranului respectind condiția:

$$y+p \leq i \leq y+p-1+q \quad (15)$$

Deoarece $q_1=10$ (v. linia 230) în cazul de față lățimea histogramei va fi de 50 puncte ($70-2 \times 10$ puncte).

Instrucțiunea din linia 395 desenează în continuarea acestei linii o altă linie înclinată față de aceasta cu 45° .

În figura 12.29 am reprezentat linia frântă rezultată, indicind coordonatele absolute ale punctelor ce o determină.

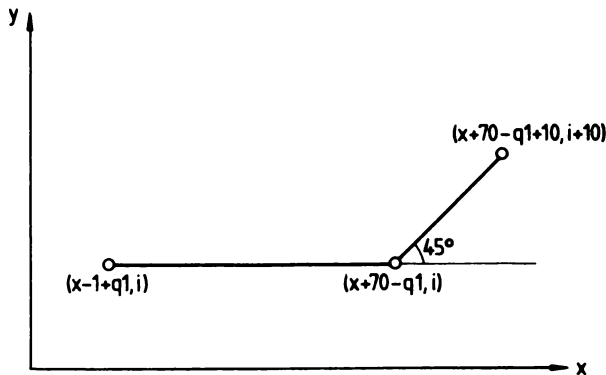


Fig. 12.29.

Ciclul **FOR-NEXT** cuprins între liniile 370 și 420 nu face decât să translateze această linie frântă pe o direcție paralelă cu axa Oy rezultind pe ecran o figură asemănătoare cu cea din fig. 12.30 (a).

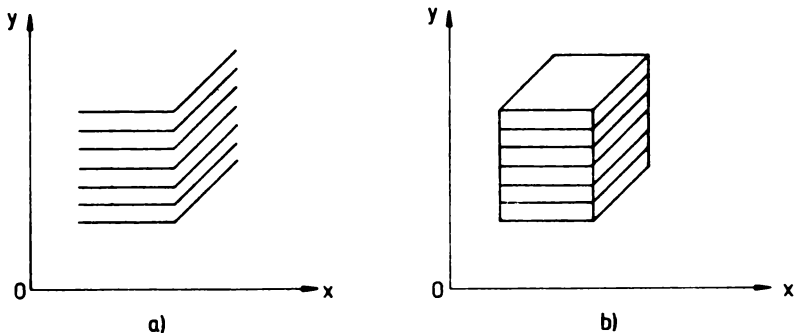


Fig. 12.30.

Instrucțiunile din liniile 325 și 326 pun „capac” acestei figuri; de fapt se completează pînă la un paralelogram ultima linie frîntă (v. fig. 12.30 (b) trasată cu ciclul **FOR-NEXT** din liniile 370–420.

Aplicație. Introduceți și executați următoarele programe BASIC-AMSTRAD:

```

a) 100 REM
110 MODE 2
120 DIM upper (640), lower (640)
130 INPUT "Continuați/Stop (Y/N)",
ans$
140 IF LEFT$(ans$, 1)="y" THEN
lne=-1 ELSE lne=0
150 centx=320 : centy=200
160 viewx=275 : viewz=120
170 REM
180 ht=40 : ohm=0.043
190 :
200 FOR i=1 TO 640
210 upper (i)=0
220 lower (i)=1000
230 NEXT i
240 :
250 FOR z=viewz-1 TO-viewz+1
STEP -5
260 lowx=INT(viewx*SQR (1-z*z/
viewz/viewz)+0.5)
270 x=-lowx
280 y=ht*SIN (ohm*SQR(x*x+z*z))
290 x1=x+centx+z
300 y1=INT (400-(centy+y+z/2)--0.5)
310 FOR x=-lowx+1 TO lowx-1
320 y=ht*SIN(ohm*SQR(x*x+z*z))
330 x2=centx+x+z
340 y2=INT (400-(centy+y+z/2)+0.5)
350 IF lne=0 THEN GOTO 390
360 IF y2<lower (x2) THEN GOSUB
460
370 IF y2>upper (x2)
THEN upper (x2)=y2 : MOVE
x1, y1 : DRAW x2, y2
380 GOTO 400
390 MOVE x1, y1 : DRAW x2, y2
400 x1=x2
410 y1=y2
420 NEXT x
430 NEXT z
440 END
450 :
460 lower (x2)=y2
470 IF upper (x2)=0 THEN upper
(x2)=y2
480 RETURN

b) 100 REM Histograme 3D
110 MODE 0
120 k1=5 : k2=20 : k3=10
130 REM
140 p=100 : q=100
150 MOVE p, q : DRAWR 300,
300*k1/k3,1 : DRAWR 200,
-200*k1/k2,1
160 DRAWR-300, -300*k1/k3,1 :
DRAWR -200, 200*k1/k2,1
170 FOR i=1 TO 3
180 MOVE p, i*50+q : DRAWR
300, 300*k1/k3,1 : DRAWR 200,
-200*k1/k2,1
190 NEXT i
200 MOVE p, q : DRAWR 0, 150,1
210 MOVER 300, 300*k1/k3 : DRAWR
0, -150,1
220 MOVER 200, -200*k1/k2 : DRAWR
0,150,1
230 :
240 REM
250 ik=2 : off=80
260 READ h : x=p+40 : y=q-8 :
GOSUB 450
270 FOR j=1 TO 3
280 READ h : x=x+off : y=x+off*
k1/k3 : GOSUB 450
290 NEXT i
300 REM
310 ik=3 : off=80
320 READ h : x=p+190 : y=q-20 :
GOSUB 450
330 FOR j=1 TO 3
340 READ h : x=x+off : y=y+off*
GOSUB 450
350 NEXT j
360 REM
370 ik=4 : off=80
380 READ h : x=p+240 : y=q-32 :
GOSUB 450
390 FOR j=1 TO 3
400 READ h : x=x+off : y=y+off*
k1/k3 : GOSUB 450
410 NEXT j
420 :
430 z$=INKEY$ : IF z$="" THEN 430
440 END
450 REM MOVE x, y : DRAWR 0, h, ik
460 FOR p=0 TO k2 STEP 2
470 MOVE p+x, y-p*k1/k2
480 DRAWR 0, h, ik
490 DRAWR k3, k1, ik
500 NEXT p
510 :
520 FOR p=k3 TO 0 STEP -2
530 MOVE x+k2+p, y-(k3-p)*k1/k3
540 DRAWR 0, h, ik
550 NEXT p
560 :
570 MOVE x+k2, y-k3*k1/k3
580 DRAWR 0,h+2,0
590 DRAWR k3, k1, 0
600 MOVE x, y+h : DRAWR k2, -k1,0
610 RETURN

```

```

620 DATA 100, 80, 120, 100, 150, 60,
100
630 DATA 100, 140, 75, 126, 150
c) 100 REM Histograme
110 MODE 1
120 SYMBOL 244, 126, 126, 126, 126,
126, 126, 126, 126
130 SYMBOL 245, 24, 24, 24, 24, 24,
24, 24, 24
140 SYMBOL 246, 0, 0, 0, 0, 0, 0, 0,
255
150 DEF FNprt$(a)=RIGHT$( "
+STR$(a), 3)
160 INPUT "Nr. bare "<30)", bars
170 DIM value (bars)
180 maximum=0
190 FOR i=1 TO bars
200 LOCATE 1,5
210 INPUT "Valori", value (i)
220 IF value (i) > maximum THEN
maximum=value (i)
230 NEXT i
240 :
250 CLS
260 scale=1
270 IF maximum>20 THEN scale=
=maximum/20
280 :
290 FOR i=1 TO bars
300 value (i)=INT (value(i)/scale)
310 NEXT i
320 :
330 FOR i=1 TO 20 STEP 2
340 LOCATE 1, 22-i : PRINT FNprt$(
INT(i*scale))
350 NEXT i
360 :
370 FOR i=1 TO 23
380 LOCATE 4, i : PRINT CHR$( 245);
390 NEXT i
400 :
410 FOR i=1 TO bars+4
420 LOCATE 4+i, 22 : PRINT CHR$(
(246));
430 NEXT i
440 :
450 FOR i=1 TO bars
460 FOR j=1 TO value(i)
470 LOCATE 7+i,22-j : PRINT
CHR$( 244);
480 NEXT j
490 NEXT i
500 z$=INKEY$ : IF z$=""
THEN 500

```

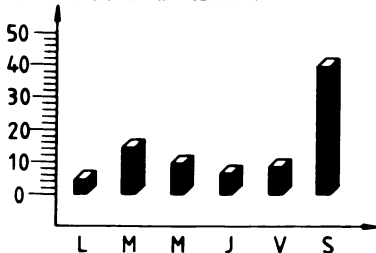
Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM

vol. 2, pag. 267

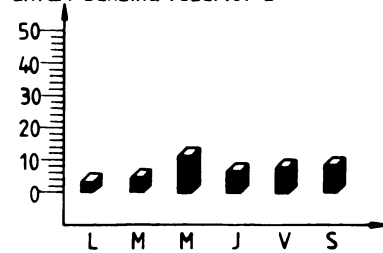
Diferențele privind realizarea histogramelor spațiale pe calculatoarele HC-85, TIM S și SPECTRUM țin atât de rezoluția ecranului cât și de setul de instrucțiuni grafice disponibile pe aceste tipuri de calculatoare.

În urma execuției programului s-au generat următoarele histograme:

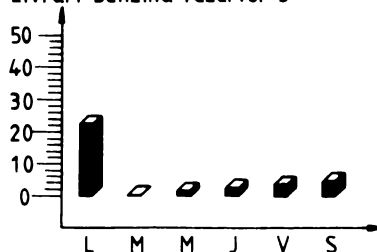
Livrări benzină rezervor 1



Livrări benzină rezervor 2



Livrări benzină rezervor 3



Aplicație. Introduceți și executați următoarele programe:

```
a) 10 FOR y=10 TO-10 STEP-1
    20 IF y<>10 AND y<>-10 AND y>-1 THEN PRINT " " ;
    30 PRINT y; TAB 4;
    40 FOR x=-10 TO 10
    50 LET k=y-x*x/2+7
    60 PRINT ("0" AND k>=.5)+("." AND k<.5);
    70 NEXT x
    80 PRINT
    90 NEXT y
    100 PRINT TAB 4; ".9.7.5.3.1.1.3.5.7.9."
```

b) Înlocuiți liniile 60 și 70 cu instrucțiunile

```
60 LET k=SQR (ABS(y*x*x2))-x
70 PRINT ("■" AND k<=.5)+("." AND k>.5);
```

c) Înlocuiți liniile 60 și 70 cu instrucțiunile

```
60 LET k=ABS (y+x)-x*x
70 PRINT ("■" AND k>=.5)+("." AND k<.5);
```

□ Trasarea histogramelor în 2D și 3D cu microcalculatoarele M118 și TPD

vol. 2, pag. 268

Particularități BASIC-80

Microcalculatoarele M118 și TPD de construcție românească, spre deosebire de microcalculatorul JUNIOR sînt prevăzute și cu opțiune grafică (rezoluția ecranului este de 512×256 puncte). Setul de instrucțiuni grafice disponibile pe cele două tipuri de microcalculatoare este următorul: **VIEWPORT** (coordonatele au valori cuprinse între 0–511 și 0–255), **WINDOW**, **MOVE**, **RMOVE**, **RDRAW**, **ROTATE** (rotește o linie trasată pe ecran în coordonate relative).

Pregătirea ecranului (ștergerea altor caractere și comutarea din mod defilare în mod pagină) se realizează cu instrucțiunile BASIC-80:

```
SIRS=CHR$(27)+"|" +CHR$(27)+"2"
PRINT SIRS
```

pentru varianta de monitor MON V3.6, și cu instrucțiunile:

```
SIRS=CHR$(27)+"0"
PRINT SIRS
```

pentru varianta de monitor V4.2.

Observație. Codurile "1" și "2" realizează respectiv ștergerea ecranului și comutarea din mod defilare (ecranul defilează) în mod pagină (ecran fix).

Grafica realizată pe cele două microcalculatoare permite utilizarea modurilor de lucru **ALFA** și **GRAFIC**. Reprezentarea informațiilor în modul de lucru **ALFA** se realizează sub formă alfanumerică (v. caracterele alfanu-

merice) pe 24 de linii ecran a câte 80 caractere fiecare. De notat că poziționarea cursului se face în raport cu colțul din stînga sus.

Reprezentarea informațiilor în modul de lucru ALFA se realizează cu instrucțiunea **PRINT** al cărei format general este:

Format general

PRINT CHR\$(27)+"1"+CHR\$(X+32)+CHR\$(Y+32); "(text alfa)"

în care $\langle X \rangle$ și $\langle Y \rangle$ reprezintă coordonatele cursorului la un moment dat. Valorile celor două coordonate se supun următoarelor restricții: $x \leq 80$; $y \leq 24$.

Reprezentarea în modul de lucru GRAFIC se realizează la nivel de bit.

Remarci privind utilizarea versiunii 5.2 a interpretorului BASIC-80.

- Pentru a schimba modul de generare a vectorilor (numai pentru versiunea 5.2 a interpretorului BASIC-80) puteți utiliza instrucțiunile:

POKE &H520A, &HC1 (funcționare normală)

POKE &H520A, &HC8 (ștergerea vectorului)

- În mod normal funcția **GCLEAR** are ca efect ștergerea ferestrei fizice selectate. Se poate însă umple fereastra logică! Utilizați următoarea secvență de instrucțiuni

POKE &H5268, &H81

POKE &H52A7, &H41

POKE &H52B4, &H81

- Pentru readucerea la normal a instrucțiunii **GCLEAR** utilizați secvența de instrucțiuni

POKE &H5268, &H38

POKE &H52A7, &H48

POKE &H52B4, &H88

Înapoi la program

Vom relua și în cadrul acestui program (operațional pe microcalculatoarele M118 și TPD) problema livrărilor de benzină afișind de această dată sub forma unor histograme „Situția livrărilor zilnice la trei rezervoare”. De notat că programul generează într-un mod care cu siguranță o să vă încinte ochiul, atât histograme în spațiul 2D cit și histograme în spațiul cu trei dimensiuni (3D).

De această dată vă punem la dispoziție numai două „piese” de analiză-proiectare și anume: ieșirile programului (histogramele în 2-D și 3-D (v. fig. 12.31, fig. 12.32) și pseudocodul (v. tabelul 12.3) după care am realizat codificarea.

Ne permitem să vă atragem atenția asupra acestui program (realizat cu versiunea 5.2), puțin mai dificil decât precedentele.

Lista programului se află în vol. 2, pag. 268. Pentru înțelegerea lui revedeți și particularitățile instrucțiunilor grafice expuse mai sus.

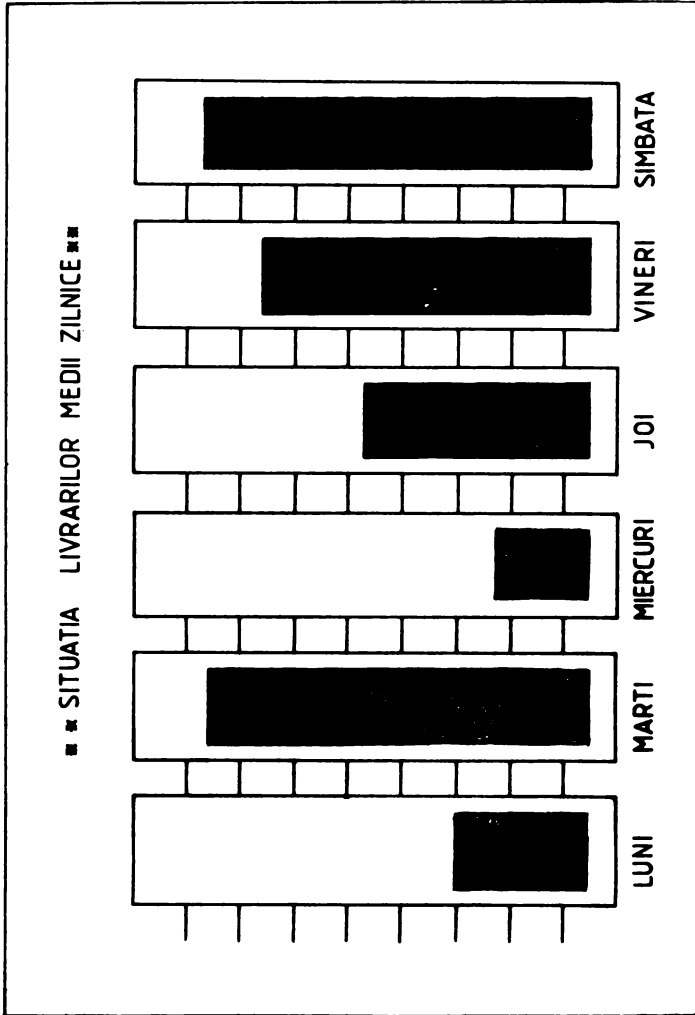


Fig. 12.31.

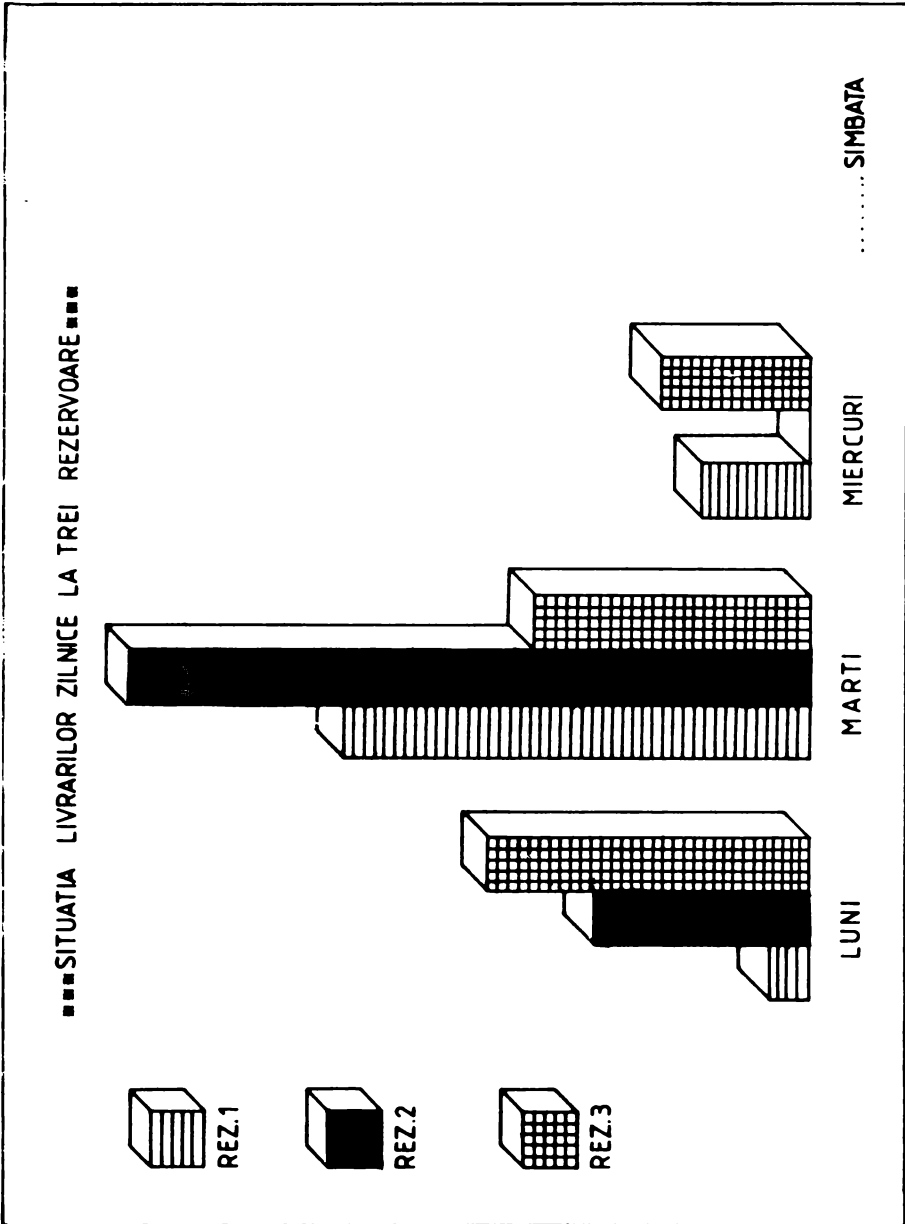


Fig. 12.32.

PSEUDOCODUL

Nume program: EXEMPLUL 12 - m

EX13M118	SEQ				RDRAW 40, 0
A	PENTRU I DE LA 1 LA 6	A		SFIRȘIT	
	READ ZIL\$(I)			IX=X	
A	SFIRȘIT			IY1=Y1	
B	PENTRU MM DE LA 1 LA 25			IY2=Y2	
	PRINT			DO 680	
B	SFIRȘIT			MOVE 40+X, Y	
	MAX=0			RDRAW 0, Y2	
	MAXB=0			RDRAW 40, 0	
A1	PENTRU I DE LA 1 LA 6			RDAW 0, -Y2	
B1	PENTRU J DE LA 1 LA 3			RDRAW -40, 0	
	PRINT "*" ; ZIL\$(I) ; TAB(13) ;			X1=X+40	
	"R"	B		CIT TIMP X1<X+80	
	PRINT USING "#"; J			X1=X1+2	
	PRINT "..."			MOVE X1, Y	
	INPUT A(I, J)			RDRAW 0, Y2	
B2	DACA MAX<A(I, J)	B		SFIRȘIT	
	MAX=A(I, J)			IX=X+40	
B2	SFIRȘIT			IY1=Y2	
B1	SFIRȘIT			IY2=Y3	
	B(I)=(A(I,1)+A(I,2)+A(I,3))/3			DO 680	
B3	DACA MAXB<B(I)			MOVE 80+X, Y	
	MAXB=B(I)			RDRAW 0, Y3	
B3	SFIRȘIT			RDRAW 40, 0	
A1	SFIRȘIT			RDRAW 0, -Y3	
	PRINT CHR\$(27)+"1"+			RDRAW -40, 0	
	+CHR\$(27)+"2"			X1=X+80	
	VIEWPORT 0, 132, 0, 99	C		PENTRU MM DE LA 1 LA 3	
	WINDOW 0, 1320, 0, 1300			X1=X1+10	
	MOVE 0, 0			MOVE X1, Y	
	DRAW 0, 1000			RDRAW 0, Y3	
	DRAW 1320, 0	C		SFIRȘIT	
	DRAW 0, 0			YY1=Y+15	
	X0=120	D		CIT TIMP YY1<Y3+100	
D	PENTRU K DE LA 1 LA 6			MOVE X+80, YY1	
	X=X0+(K-1)*200			RDRAW 40, 0	
	DO 340			YY1=YY1+15	
D	SFIRȘIT	D		SFIRȘIT	
	DO 840			DO 800	
	DO 950	340		END	
	DO 1230	680		SEQ	
	U\$=INPUT\$(1)	A		DACA IY1<IY2-20	
	DO 1370			MOVE IX, Y+IY1	
EX13M118	END			RDRAW 30, 20	
340	SEQ			RDRAW 10, 0	
	Y=100	A		IN CAZ CONTRAR	
	Y1=INT(A(K,1)*800/MAX)	A1		DACA (IY1>=IY2-20)	
	Y2=INT(A(K,2)*800/MAX)			OR (IY1<=IY2)	
	Y3=INT(A(K,3)*800/MAX)			YY1=INT(37-(IY2-IY1))	
	MOVE X, Y			MOVE IX, Y+IY1	
	RDRAW 0, Y1			RDRAW 30, 20	
	RDRAW 40, 0			RDRAW YY1, 0	
	RDRAW 0, -Y1	A1		IN CAZ CONTRAR	
	RDRAW -40, 0			YY1=IY1-IY2	
	YY1=Y			MOVE IX, Y+IY1	
A	CIT TIMP YY1<Y1+99			RDRAW 30, 20	
	YY1=YY1+10			RDRAW 40, 0	
	MOVE X, YY1			RDRAW -30, 20	

Tabelul 12.3 (continuare)

PSEUDOCODUL

Nume program: EXEMPLUL 12 – m

	RMOVE 30, 20		RDRAW 30, 20
	RDRAW 0, -YY1		RDRAW 40, 0
A1	SFIRȘIT		RDRAW -30, -20
A	SFIRȘIT		RMOVE 30, 20
680	END		RDRAW 0, -40
840	SEQ		RDRAW -30, -20
	PRINT CHR\$(27)+"1"+CHR\$(45)+		YY1=2
	+CHR\$(33)		XX1=42
	YY1=41	B	CIT TIMP XX1<79
	DO 930		MOVE XX1, 550
	PRINT "Luni"		RDRAW 0, 40
	YY1=53		XX1=XX1+2
	DO 930	B	SFIRȘIT
	PRINT "Marți"		MOVE 40, 250
	YY1=63		RDRAW 0, 40
	DO 930		RDRAW 40, 0
	PRINT "Miercuri"		RDRAW 0, -40
	YY1=78		RDRAW -40, 0
	DO 930		MOVE 40, 290
	PRINT "Joi"		RDRAW 30, 20
	YY1=88		RDRAW 40, 0
	DO 930		RDRAW -30, -20
	PRINT "Vineri"		RMOVE 30, 20
	YY1=100		RDRAW 0, -40
	DO 930		RDRAW -30, -20
	PRINT "Sâmbătă"	C	PENTRU MM DE LA 1 LA 3
840	END		MOVE 40+MM*10, 250
930	SEQ		RDRAW 0, 40
	PRINT CHR\$(27)+"1"+	C	SFIRȘIT
	+CHR\$(YY1)+CHR\$(54)		YY1=265
930	END	D	CIT TIMP YY1<285
950	SEQ		MOVE 40, YY1
	MOVE 40, 850		RDRAW 40, 0
	RDRAW 0, 40		YY1=YY1+15
	RDRAW 40, 0	D	SFIRȘIT
	RDRAW 0, -40	950	END
	RDRAW -40, 0	1230	SEQ
	MOVE 40, 890		YY1=37
	RDRAW 30, 20		DO 1270
	RDRAW 40, 0		PRINT "Rez. 1"
	RDRAW -30, 20		YY1=44
	YY1=10		DO 1270
	XX1=850+YY1		PRINT "Rez. 2"
	RMOVE 30, 20		YY1=51
	RDRAW 0, -40		DO 1270
	RDRAW -30, -20		PRINT "Rez. 3"
A	CIT TIMP XX1<889		
	MOVE 40, XX1	1230	END
	RDRAW 40, 0	1270	SEQ
	XX1=XX1+YY1		PRINT CHR\$(27)+"1"+
A	SFIRȘIT		+CHR\$(33)+CHR\$(YY1)
	MOVE 40, 550	1270	END
	RDRAW 0, 40	1290	SEQ
	RDRAW 40, 0		POKE &H5268, &H81
	RDRAW 0, -40		POKE &H52A7, &H41
	RDRAW -40, 0		POKE &H52B4, &H81
	MOVE 40, 590	1290	END

Tabelul 12.3 (continuare)

PSEUDOCODUL

Nume program: EXEMPLUL 12 – m

	1330	SEQ	C	SFIRȘIT
		POKE H5268, H88		PRINT CHR\$(27)+"1"+
		POKE H52A7, H48		+CHR\$(52)+CHR\$(33)
		POKE H52B4, H88		PRINT "Situția livrărilor medii
	1330	END		zilnice"
	1370	SEQ		PRINT CHR\$(14)
		PRINT CHR\$(27)+"1"+		YY1=43
		+CHR\$(27)+"2"		DO 930
		VIEWPORT 0, 133, 0, 99		PRINT "Luni"
		WINDOW 0, 1320, 0, 1000		YY1=55
		DO 1290		DO 930
		GCLEAR		PRINT "Marți"
		DO 1330		YY1=65
		Y=10		DO 930
		X=13		PRINT "Miercuri"
A		PENTRU MM DE LA 1 LA 6		YY1=80
		VIEWPORT X, X+16, Y, Y+80		DO 930
		GCLEAR		PRINT "Joi"
		X=13+MM * 20		YY1=90
A		SFIRȘIT		DO 930
		YY1=15		PRINT "Vineri"
		X1=10		YY1=101
		X2=125		DO 930
B		PENTRU MM DE LA 1 LA 8		PRINT "Sâmbătă"
		VIEWPORT X1, X2, YY1, YY1+1		PRINT CHR\$(15)
		GCLEAR		PRINT CHR\$(27)+"1"+
		YY1=YY1+10		+CHR\$(33)+CHR\$(35)
B		SFIRȘIT		PRINT USING "##.##"; MAXB
		DO 1290		PRINT CHR\$(27)+"1"+CHR\$(34)+
C		PENTRU MM DE LA 1 LA 6		+CHR\$(52); "0"
		X1=15+(MM-1) * 20		US=INPUT\$(1)
		X2=X1+12		PRINT CHR\$(27)+"2"
		Y1=15	1370	END
		Y2=INT(70 * B(MM)/MAXB)+15		
		VIEWPORT X1, X2, Y1, Y2		
		GOLEAR		

Aplicație: Introduceți și executați următorul program:

```

10 W=.0314159265
20 D=3.14159265/4
30 DIM A$(1, 61), Z$(100), X(100), Y(100)
40 FOR I=1 TO 100 : T=I*8.6
50 X(I)=(100-T/10)*COS(W*T+D)+125
60 Y(I)=20*SIN(62*W*T)+20+T/5
70 NEXT I
90 J=29 : GOSUB 300
100 FOR I=1 TO 61 : AS(1,I)=" " : NEXT I
105 J=J-1
110 FOR I=1 TO 100
120 IF Y(I) > (J+.5)*256/36 THEN 200
130 NEXT I
135 IF J < 6 THEN AS(1,30)="H"
140 FOR I=1 TO 61 : PRINT A$(1,I) : NEXT I : PRINT
150 IF J < 1 THEN 500
160 GO TO 100
200 K=INT(X(I)/256*60+.05) : A$(1,K)="*"; Y(I)=Y(I)-300
220 GOTO 130

```

```

300 PRINT TAB (28) "*"
310 PRINT TAB (27) "*" :;"
320 PRINT TAB (25) "*" :;"
330 PRINT TAB (27) "*" :;"
340 PRINT TAB (28) "*"
350 RETURN
500 END

```

Observație. Programul desenează un brad înalt.

TEMA 12

Răspundeți prin DA sau NU la următoarele întrebări:

– Instrucțiunile **MOVE** și **DRAW** se deosebesc de instrucțiunile **RMOVE** și **RDRAW** prin faptul că X și Y nu sînt raportate la origine.

– O imagine monocromă (alb-negru) poate fi produsă cu ajutorul unei rețele rectangulare de puncte ale unui ecran prin iluminarea sau neiluminarea unor puncte.

– Instrucțiunile pentru poziționarea spotului sînt aceleași pentru toate versiunile (dialectele) BASIC.

– Pentru a programa o imagine grafică pe ecran cu caractere mozaic nu este necesar să se precizeze codul caracterului grafic ce urmează a fi utilizat.

– Reprezentarea unui caracter grafic pe ecran se face cu ajutorul unei instrucțiuni

POKE X, Y

– Instrucțiunea **WINDOW** definește spațiul utilizator.

– Cu instrucțiunea **CIRCLE** se pot genera cercuri în limbajul BASIC-aMIC.

– Setul de instrucțiuni grafice corespunzătoare limbajului BASIC-SPECTRUM nu cuprinde instrucțiunea **MOVE**.

– Histogramele constituie un prețios ajutor în special pentru interpretarea rezultatelor.

Înlocuiți cuvintele care lipsesc din următoarele propoziții:

a) Histogramele _____ se trasează mult mai încet decît histogramele trasate prin metoda _____.

b) Microcalculatoarele _____, _____ sînt prevăzute cu opțiune grafică.

c) În BASIC-aMIC instrucțiunile **WINDOW** și _____ sînt executate implicit.

- d) Instrucțiunea **MOVE** este folosită pentru în timp ce instrucțiunea **RMOVE**
- e) Instrucțiunea **DRAW** este folosită pentru , ,
- f) Instrucțiunea **RDRAW** este folosită pentru
- g) Instrucțiunea prin care utilizatorul specifică limitele „spațiului utilizator” în care sînt cuprinse datele de reprezentat este
- h) Instrucțiunea **VIEWPORT** specifică
- i) Instrucțiunile , , , , formează setul minim de instrucțiuni grafice în limbajul BASIC-AMSTRAD.
- j) Pentru programarea culorilor în BASIC TIM S se utilizează ,
- k) Instrucțiunea **PLOT** nu aparține limbajului BASIC implementat pe calculatoarele , , ,

Scrieți cite un program BASIC pentru fiecare din problemele de mai jos:

- a) Să se genereze un con cu vîrfurile în punctul de coordonate (a, b).
- b) Să se genereze n dreptunghiuri incluse unul într-altul, primul dreptunghi trasat avînd dimensiunile cele mai mari.
- c) Să se genereze n triunghiuri isoscele cu laturile paralele și incluse unul într-altul.
- d) Să se scrie un program BASIC HC-85, TIM S, SPECTRUM care selectează aleator codul unui caracter și-l afișează pe ecran într-o poziție aleatoare.
- e) Să se deseneze un triunghi definit prin coordonate generate aleator.
- f) Să se genereze un cerc prin mai multe metode.
- g) Să se genereze un octogon.
- h) Să se genereze mai multe poligoane concentrice.
- i) Să se genereze mai multe figuri stelate.
- j) Se consideră o placă plană a cărei suprafață poate fi împărțită în dreptunghiuri (pătrate) și triunghiuri. Să se scrie un program BASIC-80 care:
- trasează conturul plăcii plane;
 - determină poziția centrului de greutate al plăcii.
- Să se traseze în 2D și 3D** histograma corespunzătoare cantităților lunare de jucării fabricate de Întreprinderea **URSULEȚUL**.
- Să se reprezinte procentele** corespunzătoare consumurilor trimestriale ale unui articol cu ajutorul unei histograme (linie) în 3D.

SOLUȚIA TEMEI 12

- Nu răspundem.
- Nu răspundem.
- Programele BASIC sînt:

a1) Lista programului BASIC-AMSTRAD

```

10 CLS
20 READ a, b
24 DATA 100, 100
26 LOCATE 1,1 : PRINT "pasul este"; : INPUT h
27 CLS
28 n=400
30 FOR r=1 TO n STEP 5
40   MOVE a+r, b
50   FOR i=0 TO 2*PI STEP PI/60
60     DRAW a+r*COS (i), b+r*SIN (i)
70   NEXT i
72   a=a--h
75   b=b--h
80 NEXT r
90 GOTO 90

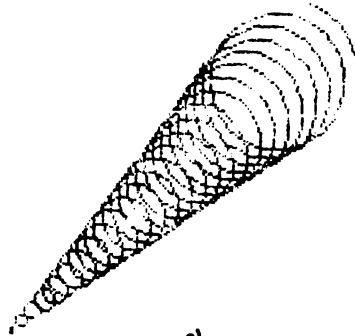
```



a)



b)



c)

Fig. 12.33. Con: a) pasul 2; b) pasul 3; c) pasul 4.

a2) Lista programului BASIC HC-85, TIM S, SPECTRUM

```

10 CLS
20 READ a, b
24 DATA 50, 50
26 PRINT AT 1,1; "pasul este"; : INPUT h
27 CLS
28 LET n=50
30 FOR r=1 TO n
40   CIRCLE a, b, r
72   LET a=a+h
75   LET b=b+h
80 NEXT r
90 GOTO 90
    
```

b1) Lista programului BASIC-AMSTRAD

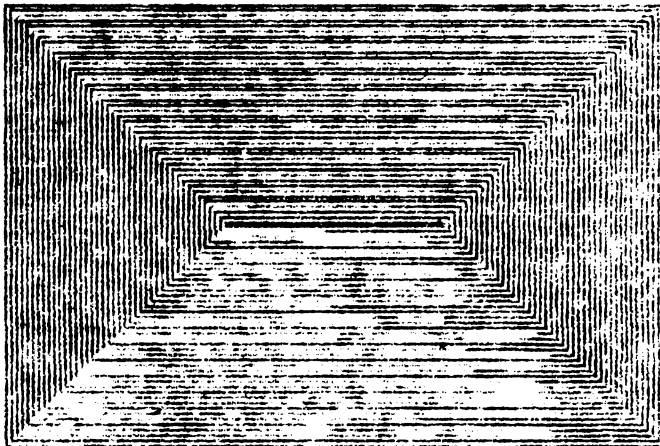
```

10 CLS
20 n=1000
30 READ x, y, l, d
40 DATA 20, 20, 360, 600
50 LOCATE 1,1 : PRINT "pasul
este"; : INPUT h
60 CLS
70 FOR i=1 TO n
80   MOVE x+h*(i-1),
      y+h*(i-1)
90   l1=l-2*h*(i-1)
100  IF l1<0 THEN 170
110  DRAW x+h*(i-1),
      y+h*(i-1)+l1
120  d1=d-2*h*(i-1)
130  DRAW x+h*(i-1)+d1,
      y+h*(i-1)+l1
140  DRAW x+h*(i-1)+d1,
      y+h*(i-1)
150  DRAW x+h*(i-1),
      y+h*(i-1)
160 NEXT i
170 GOTO 170
    
```

b2) Lista programului BASIC HC-85, TIM S, SPECTRUM

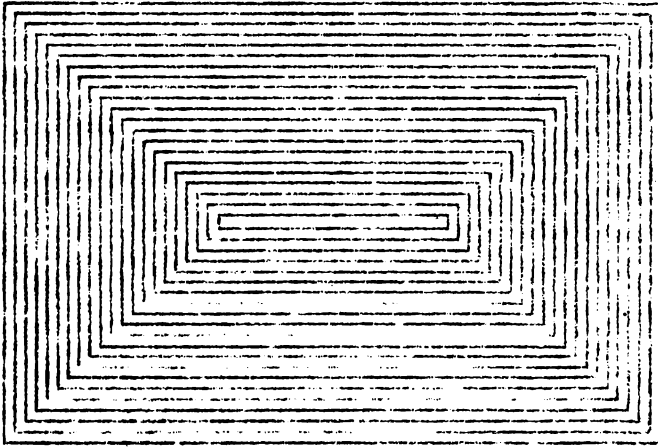
```

10 CLS
20 LET n=1000
30 READ x, y, l, d
40 DATA 5, 5, 245, 165
50 PRINT AT 1,1; "pasul este": INPUT h
60 CLS
70 FOR i=1 TO n
80   PLOT x+h*(i-1), y+h*(i-1)
90   LET l1=l-2*h*(i-1)
95   LET d1=d-2*h*(i-1)
100  IF l1<0 THEN GO TO 170
110  IF d1<0 THEN GO TO 170
120  DRAW l1, 0
125  IF d1<0 THEN GO TO 170
130  DRAW 0, d1
140  DRAW -l1, 0
150  DRAW 0, -d1
160 NEXT i
170 GO TO 170
    
```

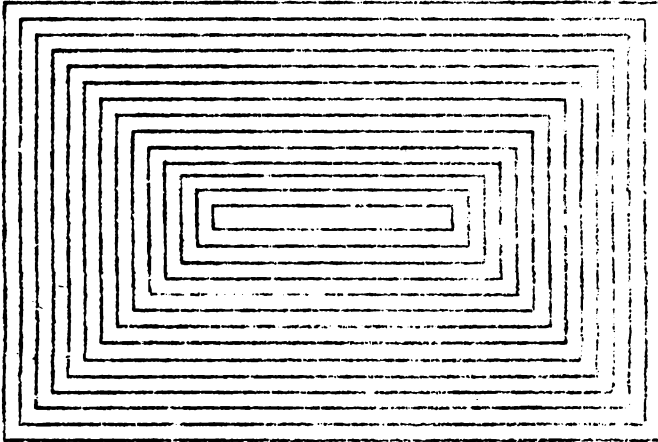


a)

Fig. 12.34. Dreptunghiuri imbricate (a, b, c).



b)



c)

Fig. 12.34.

c1) Lista programului BASIC-AMSTRAD

```

10 CLS
20 READ y, x1, x2, h
24 DATA 20, 100, 490, 375
26 LOCATE 1,1 : PRINT "pasul este";
   : INPUT h1
27 CLS
40 t=(2 * h)/(x2-x1)
50 t2=(-1+SQR (t ↑ 2+1))/t
60 d=h1/t2
70 n=(x2-x1)/(2 * d)+1
80 FOR i=1 TO n
90   MOVE x1+d * (i-1), y+h1 * (i-1)
100  DRAW x2-d * (i-1), y+h1 * (i-1)
110  DRAW (x1+x2)/2, y+h1 * (i-1)+
      t * ((x2-x1)/2-d * (i-1))
120  DRAW x1+d * (i-1), y+h1 * (i-1)
130 NEXT i
140 GOTO 140

```

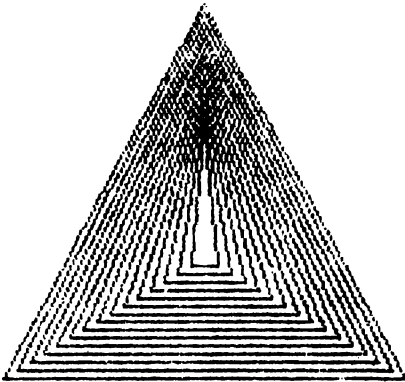
c2) Lista programului BASIC HC-85, TIM S, SPECTRUM

```

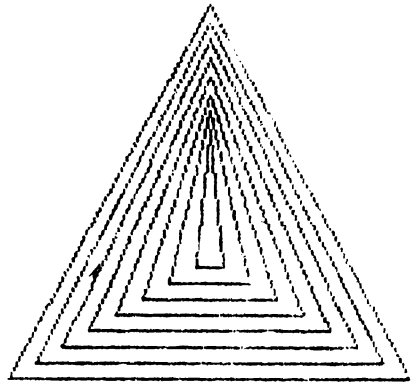
10 CLS
20 READ y, x1, x2, h
25 DATA 10, 50, 200, 140
26 PRINT AT 1,1; "pasul este"; :
  INPUT h1
27 CLS
40 LET t=(2 * h)/(x2-x1)
50 LET t2=(-1+SQR (t ↑ 2+1))/t
60 LET d=h1/t2
70 LET n:=(x2-x1)/(2 * d)+1
    
```

```

80 FOR i=1 TO n
90 PLOT x1+d * (i-1), y+h1 * (i-1)
95 LET l=x2-x1-2 * d * (i-1)
100 DRAW l, 0
105 LET a=h1 * (i-1)+t * ((x2-x1)/2
  -d * (i-1))
110 DRAW -l/2, a
115 DRAW -l/2, -a
130 NEXT i
    
```

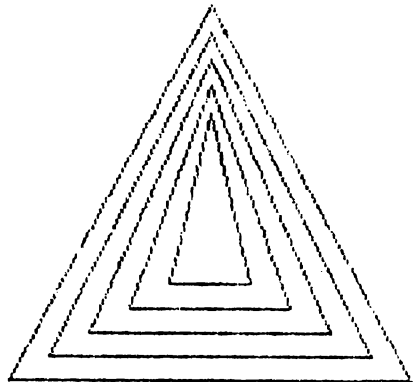


a)



b)

Fig. 12.35. Triunghiuri imbricate (a, b, c).



c)

```

d) 10 FOR n=1 TO 100
20 LET r=INT (RND * 22)
30 LET c=INT (RND * 32)
40 INK INT (RND * 7)
50 LET s:=96+INT (RND * 48)
60 PRINT AT r, c : CHR$ s;
70 NEXT n
    
```

```

e) 10 FOR s=1 TO 20
20 BORDER INT (RND * 8)
30 LET p=INT (RND * 8)
40 PAPER p : CLS
50 FOR n=1 TO 15
60 LET x1=INT (RND * 255)
70 LET y1=INT (RND * 175)
    
```

```

100 LET c = INT (RND * 255)
110 LET x2 = INT (RND * 175)
120 LET y2 = INT (RND * 255)
130 LET x1 = INT (RND * 175)
140 LET c1 = INT (RND * 8)
150 LET c = p THEN GO TO 80
160 INK c
170 PLOT x1, y1
180 DRAW x2-x1, y2-y1
190 DRAW x2-y2, y3-y2
200 DRAW x1-x3, y1-y3
210 NEXT n
220 PAUSE 200
230 NEXT s
240 INK 0

```

```

f1) 10 FOR s=1 TO 10
20 CLS
30 PLOT 0,0
40 DRAW 255,0
50 DRAW 0,175
60 DRAW -255,0
70 DRAW 0, -175
80 FOR n=1 TO 15
90 LET r = 5 * RND * 50
100 LET x = r * RND * (255-2 * r)
110 LET y = r * RND * (175-2 * r)
120 CIRCLE x, y, r
130 NEXT n
140 PAUSE 200
150 NEXT s
160 STOP

```

```

f2) 10 LET cx=128
20 LET cy=96
30 LET r=50
40 FOR x=-r TO r
50 LET y = SQR (r * r - x * x)
60 IF cx - x > 255 THEN LET
x = 255 - cx
70 IF cx + x < 0 THEN LET
x = 0 - cx
80 IF cy + y > 175 THEN LET
y = 175 - cy
90 IF cy - y < 0 THEN LET
y = 0 - cy
100 PLOT cy + x, cy - y
110 PLOT cx - x, cy + y
120 NEXT x

```

```

f3) 10 FOR n = 1 TO 15
20 LET r = 10 * INT (RND 40)
30 LET x = r * INT (RND +
(255-2 * r))
40 LET y = r * INT (RND
(175-2 * r))

```

```

50 GOSUB 500
60 NEXT n
70 STOP
500 LET th = 2 * PI / r
510 LET x1 = r
520 LET y1 = 0
530 PLOT x1, y1
540 FOR i = 1 TO r
550 LET x2 = x1 * COS th -
y1 * SIN th
560 LET y2 = x1 * SIN th + y1 *
COS th
570 DRAW x2-x1, y2-y1
580 LET x1 = x2
590 LET y1 = y2
600 NEXT i
610 RETURN

```

```

g) 10 LET xc=128
20 LET yc=88
30 LET r=50
40 LET dt=2 * PI/8
50 LET th=0
60 LET x1 = xc + r
70 LET y1 = yc
80 PLOT x1, y1
90 FOR n=1 TO 8
100 LET th = dt * n
110 LET x2 = xc + r * COS th
120 LET y2 = yc + r * SIN th
130 DRAW x2-x1, y2-y1
140 LET x1 = x2
150 LET y1 = y2
160 NEXT n

```

```

h) 10 LET r=12
20 LET xc=128
30 LET yc=88
40 FOR k=3 TO 9
50 LET ns = k
60 LET DT = 2 * PI / ns
70 LET x1 = xc + r
80 LET y1 = yc
90 PLOT x1, y1
100 FOR n=1 TO ns
110 LET th = n * dt
120 LET x2 = xc + r * COS th
130 LET y2 = yc + r * SIN th
140 DRAW x2-x1, y2-y1
150 LET x1 = x2
160 LET y1 = y2
170 NEXT n
180 LET r = r + 12
190 NEXT k

```

j) Lista programului BASIC-80

```

100 'Inițializare grafică'
110 PRINT CHR$(27) + "1" + CHR$(27) + "2"
120 'Introducerea datelor și calculul centrului de greutate'
130 GOSUB 390
140 VIEWPORT 0, 133, 0, 100 ' stabilire fereastră fizică '

```

```

150 WINDOW -30, 1330, -20, 1000 ' stabilire fereastră logică '
160 ' Trasarea axelor '
170 ROTATE 0
180 MOVE 0, 0
190 DRAW 1300, 0
200 RDRAW -30, 30
210 RMOVE 0, -60
220 RDRAW 30, 30
230 MOVE 0, 0
240 RDRAW 0, 980
250 RDRAW -30, -30
260 RMOVE 60, 0
270 RDRAW -30, 30
280 ' Trasare contur placă '
290 MOVE 0, H1
300 DRAW L1, H1
310 DRAW L1, H2
320 DRAW L1+L2, H2
330 DRAW L1+L2+L3, 0
340 GOSUB 540 ' Vizualizarea centrului de greutate '
350 ' Testarea condiției de continuare cu un nou calcul '
360 IF DNS="D" THEN 110
370 GOSUB 790
380 END
390 INPUT " Date de intrare (H1, H2, L1, L2, L3) "; H1, H2, L1, L2, L3
400 H1=H1 * 10 : H2=H2 * 10
410 L1=L1 * 10 : L2=L2 * 10 : L3=L3 * 10
420 XG1=L1/2
430 YG1=H1/2
440 XG2=L1+L2/2
450 YG2=H2/2
460 XG3=L1+L2+L3/3
470 YG3=H2/3
480 A1=L1 * H1
490 A2=L2 * H2
500 A3=L3 * H2/2
510 XG=(A1 * XG1+A2 * XG2+A3 * XG3)/(A1+A2+A3)
520 YG=(A1 * YG1+A2 * YG2+A3 * YG3)/(A1+A2+A3)
530 RETURN
540 ' Localizarea centrului de greutate al plăcii (un punct pe ecran) '
550 MOVE INT(XG), INT(YG)
560 RDRAW 0, 0
570 ' Trasarea unui cerc în jurul centrului de greutate găsit '
580 N=30
590 IU=8 * ATN(1)/N
600 U=0
610 FOR I=1 TO N
620 MOVE INT(XG), INT(YG)
630 ROTATE U
640 RMOVE 30, 0
650 RDRAW 0, 3
660 U=U+IU
670 NEXT I
680 XG=XG/10 : YG=YG/10
690 PRINT CHR$(14) ' Video invers '
700 ' Tipărirea pe ecran a coordonatelor centrului de greutate '
710 PRINT CHR$(27)+"1"+CHR$(35)+CHR$(54); "XG="; XG; "YG="; YG;
720 PRINT CHR$(15) ' Revenire în video normal '
730 PRINT CHR$(27)+"1"+CHR$(45)+CHR$(33);
740 GOSUB 810 'Tipărirea la imprimantă a rezultatelor '
750 PRINT "Doriți alt calcul (D/N)";
760 DNS INPUT$(1)
770 RETURN

```

```
780 ' Revenire in modul de lucru defilare '  
790 PRINT CHR$(27)+"2"  
800 RETURN  
810 LPRINT " Determinarea centrului de greutate al unei plăci "  
820 LPRINT " avind dimensiunile : "  
830 LPRINT " H1=" ; H1/10  
840 LPRINT " H2=" ; H2/10  
850 LPRINT " L1=" ; L1/10  
860 LPRINT " L2=" ; L2/10  
870 LPRINT " L3=" ; L3/10  
880 LPRINT " Centrul de greutate are coordonatele : "  
890 LPRINT " XG=" ; XG ; " YG=" ; YG  
900 RETURN  
RUN
```

Determinarea centrului de greutate al unei plăci
avind dimensiunile:

```
H1=53  
H2=40  
L1=25  
L2=46  
L3=36
```

Centrul de greutate are coordonatele:

```
XG=42.379 YG=20.9813
```

Nu răspundem.

Nu răspundem.

CONVERSAȚIA 13

Simularea livrărilor de lichide din trei rezervoare cu ajutorul imaginilor animate. Simboluri grafice definite de utilizator. Sprite-uri COMMODORE. Simboluri grafice în mișcare. JOCURI, APLICAȚII ȘI TESTE pentru cititor



EXEMPLELE 13-PC

☐ Caractere definite de utilizator pe calculatoarele HC-85, TIM S, SPECTRUM și AMSTRAD

Pînă acum am desenat cu calculatorul folosindu-ne numai de setul de caractere standard sau de caracterele mozaic. Ce facem însă dacă dorim să reproducem o literă grecească ori un alt simbol și mai complicat? O soluție salvatoare ar fi utilizarea pe cît posibil a instrucțiunilor **PLOT** și **DRAW** dar, nu întotdeauna metoda este și practică. Ceea ce dorim să vă prezentăm în cadrul acestei conversații este o metodă pentru generarea caracterelor (simbolurilor) nestructurată care pot fi tipărite la fel ca și simbolurile obișnuite.

Și ceea ce este minunat este faptul că atât HC-85, TIM S, SPECTRUM cît și calculatorul personal AMSTRAD dispun de o astfel de facilitate.

Metodă pentru programarea simbolurilor grafice definite de utilizator în BASIC HC-85, TIM S și SPECTRUM

1. Căutarea modelului necesar.
2. Desenarea simbolului dorit pe o grilă de 8×8 pixeli (puncte reprezentate prin pătrate). Pătratele se vor înnegri pentru a da forma simbolului dorit.
3. Definirea valorilor binare cu ajutorul funcției **BIN** corespunzătoare simbolului definit. „Zero” semnifică un pătrat alb iar „unu” semnifică un pătrat „negru”.
4. Introducerea simbolului în memorie cu ajutorul instrucțiunii **POKE** al cărei format general este:

Format general
POKE <m>, <n>

unde <m> reprezintă adresa de memorie;
<n> valoarea pe care dorim s-o înscriem.

Observație. Această înscriere nu se realizează prin intermediul procedurilor BASIC obișnuite ($0 \leq m \leq 65535$; $0 \leq n \leq 255$). Opusul lui **POKE** este **PEEK** care citește din memorie octetul de la adresa <m>).

Din fericire pentru noi, nu este necesar să știm adresa de memorie (<m>) la care dorim să înscriem simbolul pe care l-am definit, deoarece calculatorul se poate „descurca” și singur analizînd funcția **USR** „M”, unde M reprezintă caracterul căruia îi va fi afectat simbolul definit de noi. Pentru a înregistra simbolul, vom scrie:

POKE USR „M”+i, n

unde n reprezintă valoarea din linia <i> de puncte. Se mai poate utiliza și forma de scriere

POKE USR „M”+i, **BIN** (linia i, 8 biți).

De notat că în mod grafic, ori de cîte ori vom acționa tasta [M] (M reprezintă caracterul căruia i s-a atribuit simbolul definit de noi) se va afișa pe ecran simbolul definit.

Observație. În BASIC-AMSTRAD pentru definirea caracterelor utilizator se folosește instrucțiunea **SYMBOL** (v. vol. 2, pag. 57).

Aplicații

1. Folosind tehnica generării simbolurilor grafice definite de utilizator programați tasta [A] pentru reprezentarea unui pătrat și apoi a unui elefant.

```
a) 10 FOR J=0 TO 7
    20 READ B
    30 POKE USR "A"+J, B
    40 NEXT J
    50 PRINT A
    60 DATA BIN 00000000
    65 DATA BIN 00001000
    70 DATA BIN 00010100
    75 DATA BIN 00100010
    80 DATA BIN 01000001
    85 DATA BIN 0C10C010
    90 DATA BIN 00010100
    95 DATA BIN 00001000
    100 STOP
```

b) Se înlocuiesc liniile 60–95 cu

```
60 DATA BIN 0C110000
65 DATA BIN 01110000
70 DATA BIN 01111110
75 DATA BIN 01011111
80 DATA BIN 10011111
85 DATA BIN 00010010
90 DATA BIN 00010010
95 DATA BIN 00000000
```

2. Introduceți și executați următoarele programe:

```
a) 10 LET x=0
    20 PRINT AT 5, x; "■"
    30 PRINT AT 5, x; " "
    40 LET x=x+1
    50 GOTO 20

b) 10 LET x=0
    20 PRINT AT 5, x; "■"
    30 FOR i=1 TO 10
    40 NEXT i
    50 PRINT AT 5, x; " "
    60 LET x=x+1
    70 IF x>31 THEN LET x=0
    80 GO TO 20

c) 10 LET x=0
    20 LET p=x
    30 LET x=x+1
    40 IF x>31 THEN LET x=0
    50 PRINT AT 5, p; " "; AT 5, x; "■"
    60 GO TO 20

d) 10 FOR x=0 TO 30
    20 PRINT AT 5, x; "■"; AT 5,31;"
    "AND x=0
    30 NEXT x
    40 GO TO 10

e) 10 POKE USR "A", BIN 00011100
    20 POKE USR "A"+1, BIN 00011100
    30 POKE USR "A"+2, BIN 00001000
    40 POKE USR "A"+3, BIN 00001000
    50 POKE USR "A"+4, BIN 00001000
    60 POKE USR "A"+5, BIN 00011100
    70 POKE USR "A"+6, BIN 00100010
    80 POKE USR "A"+7, BIN 00100010
    90 FOR i=1 TO 20
    100 LET r=INT (RND*20)
    110 LET c=INT (RND*30)
    120 PRINT AT r, c; CHR$ (144);
    130 NEXT i

f) 10 LET y=0
    20 LET x=0
    30 PRINT AT y, x; "■"
    40 IF INKEY$="5" THEN LET x=x-1
    50 IF x<=0 THEN LET x=0
    60 IF INKEY$="6" THEN LET y=y+1
    70 IF y>=21 THEN LET y=21
    80 IF INKEY$="7" THEN LET y=y-1
    90 IF y<=0 THEN LET y=0
    100 IF INKEY$="8" THEN LET x=x+1
    110 IF x>=31 THEN LET x=31
    120 GO TO 30
```

Generarea sprite-urilor pe calculatorul COMMODORE

Ce este un sprite? Un sprite (spiriduș) este o formă de reprezentare grafică de înaltă rezoluție pe calculatorul personal COMMODORE ce presupune existența unei grile cu 24×21 puncte (504 puncte în total). Fiecare linie a sprite-ului este divizată în trei grupe a câte 8 puncte fiecare. Prima dumneavoastră sarcină în desenarea unui sprite este de a decide care din

aceste puncte doriți să se „aprindă” pentru a forma o figură sprite. Sprite-urile pot fi realizate și color (patru culori)! Îndată ce-ați umbrat sprite-ul, trebuie să aduceți figura la o formă pe care calculatorul să o poată înțelege. Fiecare din cele trei numere binare (1 corespunde pătratului înnegrit) urmează a fi convertit în echivalentul lui zecimal. Procedura se repetă pentru toate cele 21 de linii ale sprite-ului.

SPRITE-uri originale

Presupunind că nu aveți înclinații artistice, de unde vă puteți inspira pentru a desena figuri sprite? Literale alfabetului, cifrele construite din mai multe linii drepte, pot constitui o sursă. Priviți de asemenea la reclamele

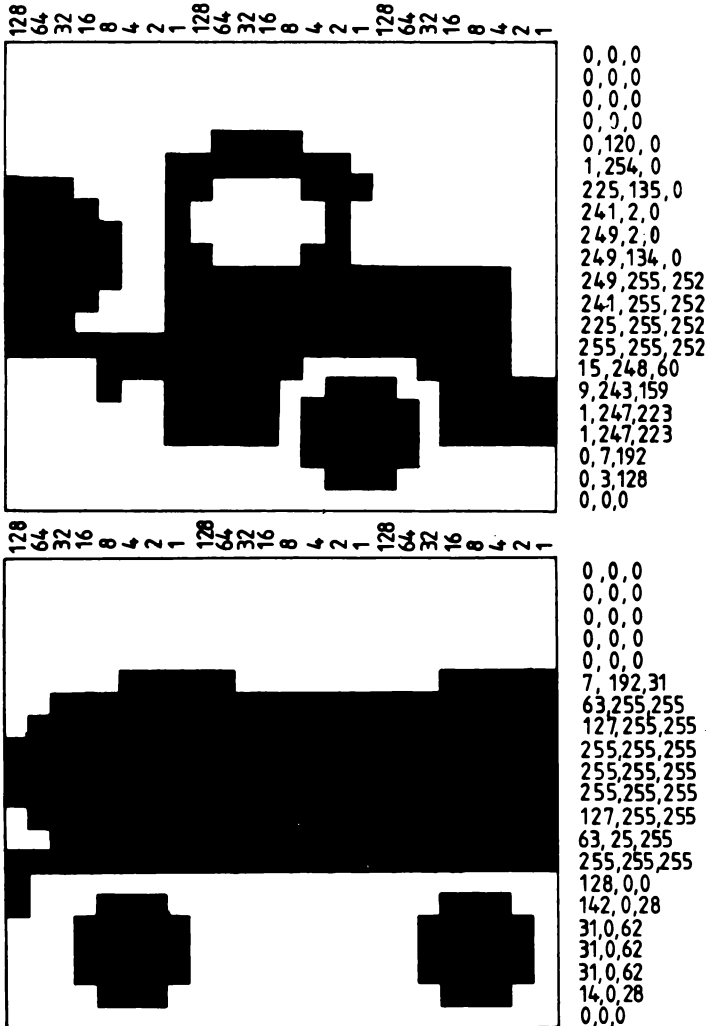


Fig. 13.1. Generarea unei cisterne cu petrol utilizând tehnica sprite-urilor.

de pe marile străzi luminate! Nu neglijați nici cusăturile artizanale, șabloanele ori vederile colorate pe care de regulă . . . le primiți!

Dacă preferați să așteptați (nefiind grăbiți să desenați propriile dvs. sprite-uri) ne permitem să vă prezentăm două sprite-uri originale reprezentând o cisternă cu petrol (v. fig. 13.1).

Stocarea sprite-urilor

Figurile sprite (putem lucra cu 8 sprite-uri, numerotate de la 0–7) se definesc cu instrucțiunile **DATA** care conțin 63 de valori. Stocarea unui sprite în memorie implică două faze distincte: a) precizarea adresei de memorare; b) introducerea propriu-zisă a valorilor sprite-ului la adresa indicată.

Indicarea adresei de memorare

Fiecare din cele 8 sprite-uri cu care putem lucra în BASIC-COMMODE are asociat un pointer. SPRITE POINTER-ul vi-l puteți imagina ca pe cuprinsul unei cărți care ne indică de fiecare dată unde începe fiecare capitol al lucrării. SPRITE POINTER-ul transmite programului adresa de încărcare în memorie a valorilor declarate în **DATA** pentru fiecare sprite. Astfel, sprite-ul zero are pointer-ul 2040, . . . , sprite-ul 7 are pointer-ul 2047.

Introducerea valorilor

După cum știți nu toată memoria calculatorului COMMODORE este la dispoziția dvs.! Amintiți-vă că BASIC-ul cit și anumite componente ale sistemului de operare sînt rezidente!

Prin stabilirea SPRITE POINTER-ului pentru sprite-ul #0 la adresa 192, de exemplu, valorile datelor pentru acest sprite trebuie citite în memorie la adresele $(192*64)+0$ pînă la $(192*64)+62$. Această operație poate fi ușor realizată utilizînd o buclă **FOR-NEXT**.

```
10 V 53248
20 FOR T 0 TO 62
30 READ A
40 POKE 192*64+T, A
50 NEXT T
```

Deoarece toate cele opt sprite-uri pot fi diferite, avînd propriile lor valori **DATA**, rezultă că fiecare sprite va trebui să fie localizat distinct.

○ simplă buclă **FOR-NEXT** poate realiza acest lucru.

```
10 FOR T=0 TO 7
20 POKE 2040+T,192+T
30 NEXT T
```

Acum, cînd avem precizată aria din memorie pentru valorile blocului de date **DATA** urmează doar ca aceste valori să fie citite.

```
10 FOR T=0 TO 62
20 READ A
30 POKE 192*64+T, A
40 NEXT T
```

Secvența de instrucțiuni BASIC-COMMODORE de mai sus va plasa cele 63 de valori (definite în **DATA**) ale sprite-ului #0 în locațiile de memorie $(192 * 64) + 0$ până la $(192 * 64) + 62$. Vom proceda analog pentru celelalte șapte sprite-uri modificând adresa zonei de memorie.

```

10 FOR T=0 TO 62 : READ A : POKE 193*64+T, A : NEXT T
20 FOR T=0 TO 62 : READ A : POKE 194*64+T, A : NEXT T
30 FOR T=0 TO 62 : READ A : POKE 195*64+T, A : NEXT T
40 FOR T=0 TO 62 : READ A : POKE 196*64+T, A : NEXT T
50 FOR T=0 TO 62 : READ A : POKE 197*64+T, A : NEXT T
60 FOR T=0 TO 62 : READ A : POKE 198*64+T, A : NEXT T
70 FOR T=0 TO 62 : READ A : POKE 199*64+T, A : NEXT T

```

Din acest moment puteți privi liniștit orice sprite. Dacă doriți să citiți fiecare din cele opt sprite-uri în zone de memorie succesive, utilizați secvența de instrucțiuni:

```

10 FOR B=0 TO 7
20   FOR T=0 TO 62
30     READ A
40     POKE (192+B)*64+T, A
50   NEXT T
60 NEXT B

```

Observație. Nu întotdeauna aveți nevoie de opt sprite-uri diferite în programele dvs. Să presupunem că doriți opt sprite-uri identice. Procedați astfel:

```

10 POKE 2040, 192
20 POKE 2041, 192
30 POKE 2042, 192
40 POKE 2043, 192
50 POKE 2044, 192
60 POKE 2045, 192
70 POKE 2046, 192
80 POKE 2047, 192

```

Și-acum, vă propunem, un program original pentru generarea unei cisterne pornind de la cele două sprite-uri pe care vi le-am prezentat la început.

```

10 Q=53248
20 FOR I=0 TO 62
30   READ A
40   POKE 255*64+I, A
50 NEXT I
60 POKE 2040, 255
65 FOR I=0 TO 62
70   READ A
75   POKE 254*64+I, A
80 NEXT I
85 POKE 2041, 254
90 POKE Q+21,3
95 POKE Q+39,1 : POKE Q+40,1
100 POKE Q+130 : POKE Q+1, 130.
110 POKE Q+2, 178 : POKE Q+3, 130
120 POKE Q+23,3 : POKE Q+29,3
130 GET AS : IF AS=" " THEN 130
140 POKE Q+21,0
160 DATA 0, 0, 0, 0, 0, 0, 0, 0
170 DATA 0, 0, 0, 0, 0, 0, 0, 7
180 DATA 192, 31, 63, 255, 255, 127, 255, 255
190 DATA 255, 255, 255, 255, 255, 255, 255, 255
200 DATA 255, 127, 255, 255, 63, 255, 255, 127
210 DATA 255, 255, 7, 1, 192, 3, 57, 128
220 DATA 1, 125, 0, 1, 109, 0, 0, 124

```

```

230 DATA 0, 0, 36, 0, 0, 0
240 DATA 0, 0, 0, 0, 0, 0, 0
250 DATA 0, 0, 0, 0, 0, 120, 0, 1
260 DATA 254, 0, 225, 135, 0, 241, 2, 0
270 DATA 249, 2, 0, 249, 134, 0, 249, 255
280 DATA 249, 241, 255, 248, 225, 225, 248, 255, 255,
    248, 255, 248, 56, 255, 243, 152
290 DATA 1, 247, 220, 1, 246, 220, 0, 7, 192, 0, 8, 128, 0, 0, 0

```

Aplicație

1. Includeți în programele dvs. următoarele sprite-uri reprezentind:

a) O ceașcă de cafea.

```

10000 REM CEAȘCA DE CAFEA
10010 DATA 8
10020 DATA 66, 0, 4, 33, 0
10030 DATA 2, 16, 128, 4, 33
10040 DATA 0, 8, 66, 0, 0
10050 DATA 0, 0, 255, 255, 192
10060 DATA 255, 255, 192, 245, 85
10070 DATA 252, 245, 85, 254, 245
10080 DATA 85, 199, 245, 85, 195
10090 DATA 245, 85, 195, 245, 85
10100 DATA 195, 245, 85, 199, 245
10110 DATA 85, 206, 245, 85, 252
10120 DATA 245, 85, 248, 245, 85
10130 DATA 240, 245, 85, 192, 255
10140 DATA 255, 192

```

b) Un lup.

```

10280 DATA 0, 0, 0, 0, 0
10290 DATA 0, 0, 0, 48, 1
10300 DATA 192, 112, 0, 240, 240
10310 DATA 0, 121, 224, 0, 61
10320 DATA 192, 0, 31, 192, 0
10330 DATA 15, 224, 0, 63, 224
10340 DATA 0, 255, 240, 127, 199
10350 DATA 248, 255, 255, 252, 255
10360 DATA 255, 254, 255, 255, 254
10370 DATA 128, 255, 254, 42, 255
10380 DATA 254, 127, 255, 252, 1
10390 DATA 255, 248, 1, 255, 240
10400 DATA 1, 255, 240

```

2. Introduceți și executați următorul program care produce mișcarea unui fluture.

```

15 V=53248
30 PRINT CHR$(147) : POKE V+21,0
40 POKE 53281,7 : POKE 53280,5
90 POKE 2040,192
110 FOR M=0 TO 162
120 READ A
130 POKE 192*64+M, A
140 NEXT M
150 POKE V+39,6
165 POKE V+0,100
170 FOR Y=1 TO 255
180 POKE V+1, Y
185 POKE V+21,1
190 NEXT Y

```

```

195 GO TO 170
10000 DATA 3, 0, 192, 57, 129
10010 DATA 156, 124, 195, 62, 254
10020 DATA 102, 127, 255, 36, 255
10030 DATA 255, 155, 255, 255, 219
10040 DATA 255, 255, 255, 255, 255
10050 DATA 255, 255, 255, 255, 255
10060 DATA 255, 255, 255, 255, 255
10070 DATA 255, 255, 255, 255, 255
10080 DATA 255, 255, 255, 255, 255
10090 DATA 255, 255, 255, 255, 255
10100 DATA 255, 127, 219, 254, 63
10110 DATA 153, 252, 31, 24, 248
10120 DATA 14, 24, 112

```

3. Introduceți și executați următorul program:

```

5 REM SIMULAREA LUI DRAW TO
10 INPUT "INTRODUCEȚI X1": X1
20 INPUT "INTRODUCEȚI Y1": Y1
30 INPUT "INTRODUCEȚI X2": X2
40 INPUT "INTRODUCEȚI Y2": Y2
42 POKE 53265, 59
44 POKE 53272,25
45 FOR V=1024 TO 2023
46 POKE V, 16
47 NEXT V
48 FOR T=8192 TO 16383
49 POKE T, 0
50 NEXT T
55 DX=X2-X1 : DY=Y2-Y1
57 IF ABS(DY)<ABS(DX) THEN 64
58 FOR YL=Y1 TO Y2 STEP SGN(DY)
59 X=DX/DY*YL+X1

```

```

60 Y=YL
61 GOSUB 200
62 NEXT YL
63 GOTO 300
64 FOR XL=X1 TO X2 STEP SGN(DX)
65 X=XL
66 Y=DY/DX*XL+Y1
67 GOSUB 200
68 NEXT XL
69 GOTO 300
200 YP=INT(Y/8)
210 XP=INT(X/8)
220 A1=(YP*40+XP)*8
230 AY=Y-8*YP+A1
240 AA=8192
250 R=X-8*XP
260 M=2+(7-R)

```

```

270 I=PEEK(AY+AA)
280 POKE AY+AA, I OR M
290 RETURN

```

```

300 FOR T=1 TO 5000 : NEXT T
310 POKE 53265,27 : POKE 53272,21
320 END

```

JOC PE COMMODORE

```

100 REM PARABOLA
110 PRINT CHR$(147)
120 X0=0 : Y0=0
130 PRINT CHR$(147) : INPUT "INTRODUCETI
    UNGHIUL"; W
140 INPUT "INTRODUCETI VO" : VO
150 W=W*2*3.1416/360
160 VX=VO*COS(W)
170 VY=VO*SIN(W)
180 G=9.81
190 T=0 : H=0 : L=0
200 Y=-G/2*T*T+VY*T

```

```

210 IF Y/0 THEN FOR D=1 TO 500 :
    NEXT D : GOTO 130
220 X=VX*T
230 LOC=1024+40*INT (24-Y/50)+INT
    (X/50)
240 COL=55296+40*INT (24-Y/50)+INT
    (X/50)
250 IF LOC<1024 THEN T=T+1 : GOTO
    200
260 POKE LOC, 81 : POKE COL, 4
270 L=X : H=Y
280 T=T+1
290 GOTO 200

```

□ Programarea imaginilor animate pe calculatorul aMIC

Deplasarea unui caracter semigrafic

Teoria privind mișcarea unei imagini grafice precizează următoarele posibilități: a) se trasează graficul; b) se produce o întârziere; c) se șterge graficul; d) se trasează un nou grafic; d) se reia ciclul descris.

Efectul de mișcare a unui caracter grafic poate fi obținut prin suprimarea iluminării acestuia nu cu mult timp înainte de apariția unui caracter identic în poziția imediat următoare sau prin plasarea în aceeași poziție a unui caracter grafic urmat sau nu de un blank. Pentru definirea poziției caracterelor se utilizează variabilele (de stare).

Să privim exemplele care urmează ca pe niște aplicații „brutale” ale teoriei animației. Treziți aMIC-ul dvs!

```

10 INIT
15 FOR J=16 TO 25
20 PRINT AT (16, J); "■"
25 PRINT AT (16, J); " "
30 NEXT J
40 END

```

Ați avut o impresie de deplasare paralelă cu axa Ox, a caracterului "■"? Încercați următoarea secvență, mult mai rapidă.

```

10 INIT
20 FOR J=15 TO 24
25 PRINT AT (16, J); " ■"
30 NEXT J
40 END

```

După cum ați putut constata cele două secvențe sînt identice.

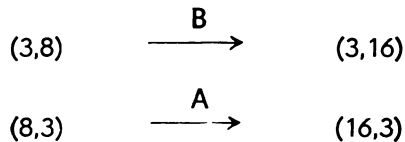
Obțineți acum același efect de deplasare a caracterului "■" pe verticală!

```
10 INIT
20 FOR I=16 TO 25
25   PRINT AT (I, 15); "■"
30   PRINT AT (I, 15); " "
40 NEXT I
```

Aplicație. Să se simuleze deplasarea caracterului semigrafic "■" pe drumul cel mai scurt dintre punctele A (1, 2) și B (16, 17).

Deplasarea simultană a două caractere

Să considerăm cazul simulării deplasării a două caractere A și B legate între ele printr-o funcție.



După cum constatați, funcția de legătură este evidentă (punctele sînt simetrice față de diagonală).

```
5 INIT
10 FOR I=8 TO 16
15   PRINT AT (3, I); "B"
20   PRINT AT (3, I); " "
25   PRINT AT (I, 3); "A"
30   PRINT AT (I, 3); " "
40 NEXT I
50 END
```

Observație. Impresia de deplasare se datorează rapidității mișcării (instrucțiunile se execută una după alta).

Deplasarea unei figuri mai complicate

Să abordăm o problemă puțin mai complicată – simularea deplasării unui vehicul. Vom considera vehiculul alcătuit din blocuri de caractere "□" de lungime J1 și înălțime I1. Roțile le vom reprezenta prin "○".

Subrutina de generare a vehiculului „parcat” în punctul de coordonate (L1, K1) este:

```
5 INIT
10 FOR J=0 TO J1
20   FOR I=0 TO I1
30     PRINT AT (L1+I, K1+J); "□"
40   NEXT I
50   PRINT AT (L1+I1+1, K1+J); "○"
60 NEXT J
```

Ideea de animație privind deplasarea vehiculului din poziția K1 în poziția K2 constă în ștergerea coloanei de caractere bloc din stînga vehiculului și

adăugarea acestuia în dreapta vehiculului (v. liniile 25 și 45 ale subrutinei de mai jos)

```

10 FOR J=K1 TO K2
20   FOR I=0 TO I1
25     PRINT AT (L1+I, J+J1+1); "□"
30     PRINT AT (L1+I, J); " "
35   NEXT I
40   PRINT AT (L1+I1+1, J+J1+1); " "
45   PRINT AT (L1+I1+1, J); " "
50 NEXT J
60 END

```

□ Animație la nivel de caracter pe calculatorul AMSTRAD

vol. 2, pag. 372

Programul pe care vi-l prezentăm în continuare realizează o simulare pe calculatorul AMSTRAD privind aprovizionarea cu benzină (de trei calități) a trei beneficiari. El constituie un pretext pentru a putea urmări o animație la nivel de caracter privind: umplerea/golirea rezervoarelor și deplasarea vehiculelor.

Lista variabilelor programului este următoarea: d (distanța dintre centrele de simetrie ale rezervoarelor); n (numărul de rezervoare); l1, k1 (coordonatele punctului unde se generează vehiculul); t (vectorul cantității de benzină încărcate pe fiecare vehicul); x (vectorul absciselor centrelor de simetrie ale vehiculelor); v (matricea cu nivelul minim, nivelul maxim și nivelul curent al benzinei pentru un rezervor k); c) matricea cantităților de benzină solicitate de către beneficiarul i din rezervorul j); z (vectorul ale cărui elemente indică locul de oprire al vehiculului).

Vă invităm în continuare să analizați singuri, fără ajutorul nostru următoarele secvențe de instrucțiuni ale programului:

- 60– 160 generare rezervoare din caractere mozaic
- 161– 163 generare drum autovehicule
- 205– 235 afișare cerere benzină
- 240– 275 alimentare cu benzină
- 280– 290 „distrugere” vehicule
- 340– 380 subrutină de generare vehicul
- 390– 440 subrutină de deplasare vehicul
- 750– 795 simulare curgere benzină (o tonă)
- 950– 980 subrutină umplere rezervor
- 1000–1025 subrutină ștergere vehicul
- 1100–1135 subrutina afișare calitate (cifra octanică) benzină

A fost și . . . penultima provocare informatică!

TEMA 13

 Răspundeți prin DA sau NU la următoarele întrebări:

- Cu instrucțiunile **PLOT** și **DRAW** putem reprezenta orice simbol de pe cărțile de joc.
- În BASIC-PRAE putem programa orice tastă pentru reprezentarea unui lup.
- În BASIC HC-85 un caracter definit se introduce în memorie cu instrucțiunea **POKE**.
- Un sprite poate fi desenat pe o grilă de 24×21 puncte.

 Înlocuiți cuvintele care lipsesc din următoarele propoziții:

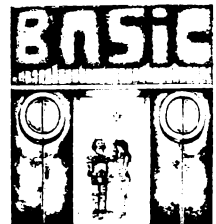
- a) Un sprite este o formă de reprezentare grafică de rezoluție pe calculatorul .
- b) Cu instrucțiunea **PRINT AT** nu se poate șterge decât .
- c) Cu instrucțiunea **PRINT AT** se poate/nu se poate șterge o linie creată cu instrucțiunea **DRAW**.
- d) Pentru programarea simbolurilor grafice definite de utilizator în BASIC HC-85, TIM S, SPECTRUM procedăm astfel: a) b) c) .
- e) Instrucțiunea
20 **POKE** **USR** "A"+1, **BIN** 00011100
are următoarea semnificație .
- f) În BASIC-COMMODORE pot fi programate sprite-uri.
- g) Prin stabilirea **SPRITE POINTER**-ului pentru sprite-ul #0 la adresa 255, valorile datelor pentru acest sprite trebuie citite în memorie de la adresele pînă la .

 Scrieți cite un program BASIC pentru fiecare din problemele de mai jos:

- a) Să se simuleze aprovizionarea cu benzină a trei beneficiari în limbajul BASIC-aMIC.
- b) Să se simuleze aprovizionarea cu benzină a trei beneficiari în limbajul BASIC-PRAE.
- c) Să se simuleze aprovizionarea cu benzină a trei beneficiari în limbajul BASIC HC-85, TIM S, SPECTRUM.
- d) Să se simuleze aprovizionarea cu benzină a trei beneficiari în limbajul BASIC-80.
- e) Să se simuleze mișcarea unei mașini pe plan înclinat.

- f) Să se simuleze mișcările unui înotător.
- g) Să se simuleze mișcările calului și nebunului pe tabla de șah.
- h) Să se simuleze mișcările unei sfere în aruncarea pe oblică și pe orizontală.
- i) Să se simuleze ciclul Carnot.
- Să se reprezinte cu ajutorul imaginilor animate, dinamica producției de jucării la Intreprinderea URSULEȚUL.
- Să se reprezinte consumurile trimestriale ale unui articol cu ajutorul imaginilor animate.

Proiectarea și realizarea unui simulator pentru o stație de livrare produse lichide. Animație, tehnica WINDOW, semnalizare optică și acustică, reprezentarea în 3D



TEMA 14

În această ultimă conversație ne propunem să abordăm o problemă de grafică interactivă privind simularea pe calculatoarele personale HC-85, TIM S, SPECTRUM a livrărilor de benzină, motorină și petrol efectuate de către o stație de livrare produse petroliere.

Vom începe conversația cu o temă care permite aplicarea metodelor și tehnicilor de grafică pe calculator.

Să se proiecteze și realizeze un program BASIC HC-85, TIM S, SPECTRUM care să simuleze livrările de benzină, motorină și petrol dintr-o stație de livrare produse petroliere.

□ Cerințe de prelucrare

Se solicită realizarea următoarelor prelucrări:

- reprezentarea grafică (2D) a celor trei rezervoare (R_1 , R_2 , R_3), a pompei centrifuge și a conductelor de alimentare;
- încărcarea rezervoarelor cu cele trei produse petroliere: benzină (R_1); motorină (R_2); petrol (R_3);
- apariția cisternelor având inscripționate numerele auto corespunzătoare: (30-MS-1045; 31-EZ-3014 etc.);
- umplerea conductelor de alimentare cu produsele petroliere corespunzătoare;
- încărcarea cisternelor (în același timp cu golirea rezervoarelor corespunzătoare);
- plecarea cisternei care a alimentat și aducerea unei alte cisterne pentru încărcat;
- afișarea orei curente (în colțul din dreapta sus al ecranului). Programul de lucru este 7–14. În momentul depășirii orei 14 se va semnaliza (acustic și optic) „Sfârșitul programului de lucru”. Ora curentă de începere a programului poate fi schimbată;
- umplerea conductei de alimentare va marca pornirea pompei centrifuge în vederea începerii umplerii unei cisterne sau a continuării umplerii cisternei în cazul umplerii rezervorului;
- golirea conductei va marca oprirea pompei centrifuge. Pot apare două cazuri: a) cisterna de la poziția respectivă s-a umplut; b) rezervorul respectiv s-a golit;
- în cazul golirii unui rezervor se va întrerupe activitatea de livrare și se va trece automat la umplerea rezervorului respectiv;
- în cazul în care o cisternă s-a umplut se va trece automat la expedierea ei și se va aduce o nouă cisternă pentru încărcare;
- pentru fiecare din cele trei produse petroliere se vor afișa:
 - capacitatea maximă a cisternei respective (în tone);
 - cantitatea de combustibil încărcată în cisternă;
 - timpul care a mai rămas pentru încărcarea completă a cisternei;

- în funcție de produsul solicitat, se vor genera următoarele rapoarte:
 - **L1.** Lista numerelor auto și a capacităților cisternelor;
 - **L2.** Situația livrărilor față de planificat, în procente, pentru lunile anului în curs;
 - **L3.** Situația valorică a livrărilor pe produs petrolier;
 - **L4.** Situația valorică a livrărilor lunare pentru fiecare produs în parte;
- modul de lucru privind încărcarea cisternelor poate fi secvențial (prin acționarea unei taste) sau automat;
- livrările planificate și realizate se vor reprezenta într-o histogramă tridimensională.

□ Cerințe de rezolvare

- Pentru deplasarea cisternei (intrare/ieșire) se va realiza o animație la nivel de caracter.
- Pentru descărcarea rezervorului, încărcarea cisternei, umplerea/golirea conductei de alimentare se va realiza o animație la nivel de pixel.
- Afișarea datelor privind starea sistemului, livrărilor etc. se va realiza prin aplicarea tehnicii *WINDOW*.
- Pentru introducerea orei și a minutului se va realiza un *minieditor* de cifre.
- Tratarea succesivă în timp a mai multor teme (descărcare rezervor, încărcare cisternă) de același tip se va realiza prin folosirea aceluiași bloc de program.
- Se vor emite *semnale acustice* (avertizare acustică cu înălțimi sonore aleatoare) și *semnale optice* (semnalizare optică intermitentă) în momentul când o cisternă este deja plină.
- Numerele auto și capacitățile cisternelor se vor genera în mod aleator.
- Programul va fi ghidat de un meniu principal și de submeniuri pentru realizarea următoarelor funcțiuni:

A – AUTO

- B – Benzină
- P – Petrol
- M – Motorină

O – OPȚIUNI

- H – Opțiune oră
- A – Livrări azi
- % – Livrări procentuale
- L – Livrări lunare
- S – Secvențare
 - V – bucla se închide secvențial
 - A – bucla se închide automat
- H – HIST

SOLUȚIA TEMEI 14

□ Proiectarea programului

Se pot imagina numeroase variante pentru realizarea acestui simulator. În ceea ce ne privește, ne-am străduit să vă prezentăm o soluție simplă în care să regăsiți aplicate majoritatea tehnicilor utilizate curent în infografică: reprezentare în 2D și 3D, animație la nivel de pixel și caracter, semnalizare optică și acustică, meniuri și submeniuri, tehnicile WINDOW etc.

În fig. 14.1 este reprezentată diagrama de structură a simulatorului.

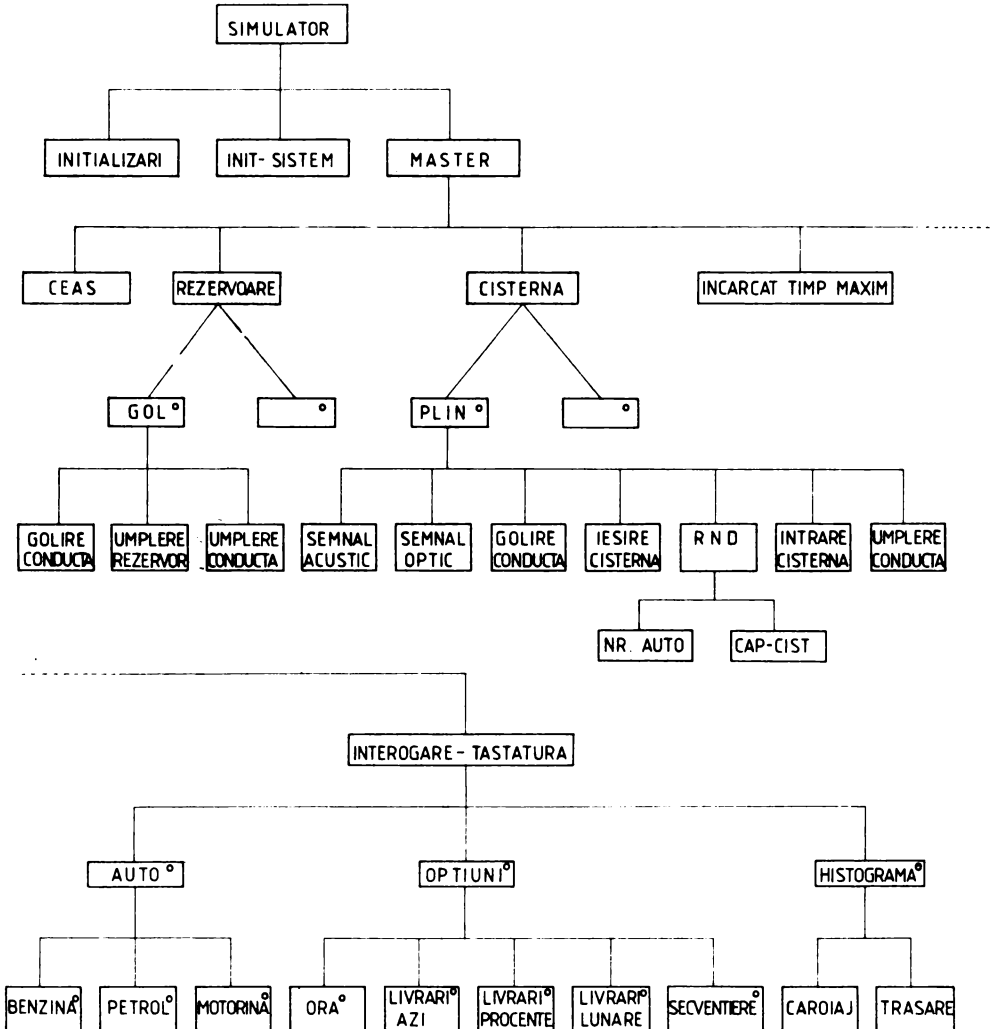


Fig. 14.1.

□ Codificarea în limbajul BASIC HC-85, TIM S, SPECTRUM*

vol. 2, pag. 276

Inițializări

Instrucțiunile 1000–1160 dimensionează tablourile de date și le inițializează.

Inițializări sistem

Secvența de instrucțiuni 1500–1595 inițializează variabilele simple (sum, o, m, pre, d, secv) și indexate $x(1)$, $x(2)$, $x(3)$. Se apelează subrutinele (8600, 8517, 8435, 8210) pentru umplerea rezervoarelor, alegerea capacității cisternelor (în mod aleator), aducerea cisternei, umplerea conductei de alimentare.

Ceas

Instrucțiunile 2105–2150 afișează ora și minutul (incrementat cu 2) în timp real. Ceasul dă semnalul privind sfârșitul programului de lucru.

Rezervoare

Secvența de instrucțiuni 2299–2324 realizează golirea rezervoarelor la nivel de pixel. Se realizează animația pentru umplerea/golirea conductei, umplerea rezervorului.

```
2299 REM                      - rezervoare -
2300 FOR n=1 TO 3
2310 PLOT OVER 1; 23, w (n)+3+x(n)
2315 DRAW OVER 1; 70,0
2317 LET g(n)=g(n)+1
2320 IF g(n)>=f(n) THEN LET g(n)=0 : LET x(n)=x(n)-1
2324 IF x(n)<=0 THEN GO SUB 8200 : GO SUB 8600 : GO SUB 8210
```

Cisterna

Instrucțiunile 2327–2347 realizează încărcarea cisternei pixel cu pixel. Se apelează subrutinele pentru: generarea semnalului acustic și optic, golirea conductei, ieșirea cisternei (animație la nivel caracter), generarea aleatoare a numerelor auto și a capacităților cisternelor, intrarea cisternelor (animație la nivel de caracter).

```
2327 REM                      - cisterne -
2330 LET i(n)=i(n)+.25
```

* Programul a fost realizat cu ing. Dan Drăgan, după o idee a autorului.

```

2335 IF i(n)<y(n) THEN GO TO 234
7
2337 LET y(n)=y(n)+r(n)
2340 IF INT (y(n)/(r(n)+.0001))>=3 THEN LET inc=136 : LET lung=80 : GO TO 2345
2343 LET inc=168 : LET lung=16
2345 PLOT OVER 1 ; inc, w(n)+8+INT (y(n)/(r(n)+.0001)) : DRAW OVER 1; lung, 0
2347 IF y(n)/(r(n)+.0001)>16 (THEN GO SUB 8300 : PRINT FLASH 1; AT n*7-4,19;
" ■ " : GO SUB 8200 : GO SUB 8400 : GO SUB 8517 : BEEP .04,32 : PAUSE 1 :
BEEP .06,32 : GO SUB 8435 : GO SUB 8210 : PRINT FLASH 0; AT n*7-4,19;
" "

```

Incărcat-timp maxim

Secvența de instrucțiuni 2353–2370 afișează la interval de aproximativ două secunde: cantitatea (în tone) încărcată în cisternă pentru fiecare produs (benzină, motorină, petrol); timpul (în minute) rămas pînă la umplerea completă a cisternei; capacitatea cisternei.

Interogare tastatură

Instrucțiunile 2380–2410 permit, conform meniului, alegerea uneia din următoarele opțiuni: A (auto), O (opțiuni), H (histogramă).

Auto interoghează produsul petrolier (B – benzină; M – motorină; P – petrol). În funcție de produsul petrolier selectat se generează raportul L1.

Opțiuni cuprinde cinci alternative după cum urmează: **H** setează ora de început program (7,00 sau o altă oră din intervalul 7–14); **A** interoghează produsul petrolier (B – benzină; M – motorină; P – petrol) și generează raportul L3; **O** generează raportul L2; **L** generează raportul L4. **S** realizează închiderea buclei în mod secvențial (prin acționarea tastei V) automat (A).

Histograma 3D

Instrucțiunile 7000–7245 generează histograma tridimensională (în spațiul 3D) a livrărilor (planificat, realizat) pentru cele trei produse petroliere: motorină, petrol, benzină. În planul orizontal se reprezintă produsele petroliere și lunile anului: IF (ianuarie, februarie), MA (martie, aprilie) etc. Pe verticală (Z1 și Z2) se reprezintă planificatul și realizatul privind livrările celor trei produse petroliere. Instrucțiunile 7000–7130 realizează textul și inițializările pentru histogramă; instrucțiunile 7135–7180 trasează caroiajul, iar instrucțiunile 7185–7245 trasează elementele histogramei, în conformitate cu valorile lui Z1 și Z2.

Alte subrutine

În cadrul programului mai sînt apelate subrutinele: Livrări azi (7900–8010); Opțiune ora (8100–8170); Umplerea/golirea conductei (8200–8270); Semnal acustic (8300–8340); Leșire cisternă (8400–8432); Intrare cisternă (8435–8460); Generare aleatoare capacității cisterne/Generare aleatoare

numere auto (8500–8600); Umplerea rezervorului (8600–8680); Livrări procente (8700–8740); Secvențiere (8800–8840); Livrări lunare (8900–8945); 50021 (transferă imaginea binară din memoria ecran într-o zonă tampon de memorie); 50000 (aduce imaginea binară salvată cu subrutina 50021 înapoi în memoria ecran).

Imagini generate de program

Simularea livrării celor trei produse petroliere este ilustrată în fig. 14.2.

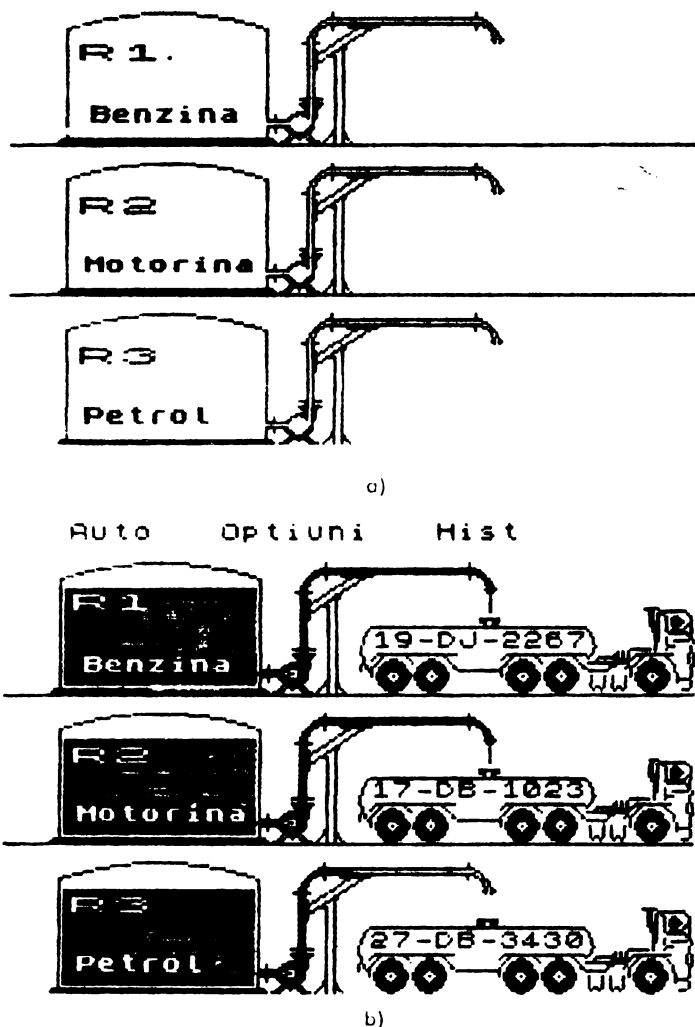
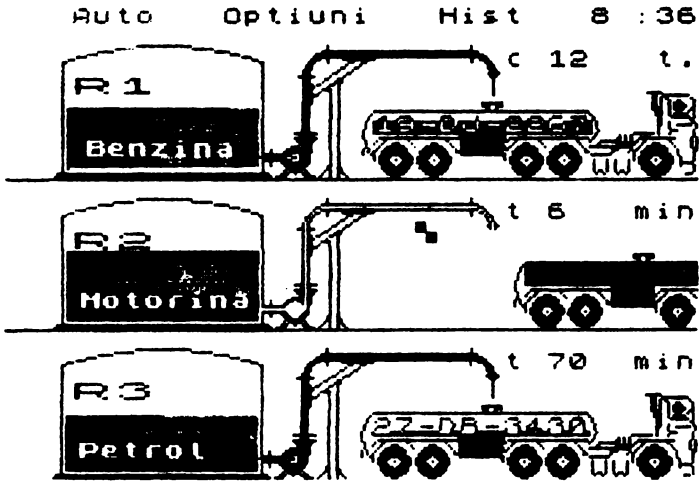
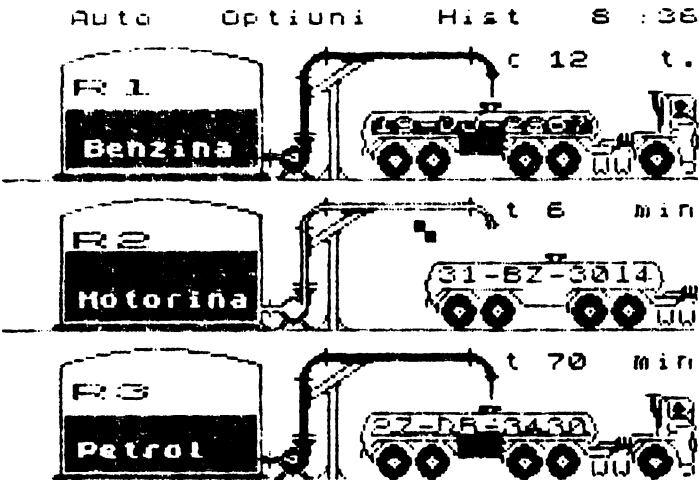


Fig. 14.2. Simulator: a) imaginea rezervorului înainte de inițializările sistemului; b) încărcarea celor trei cisterne;

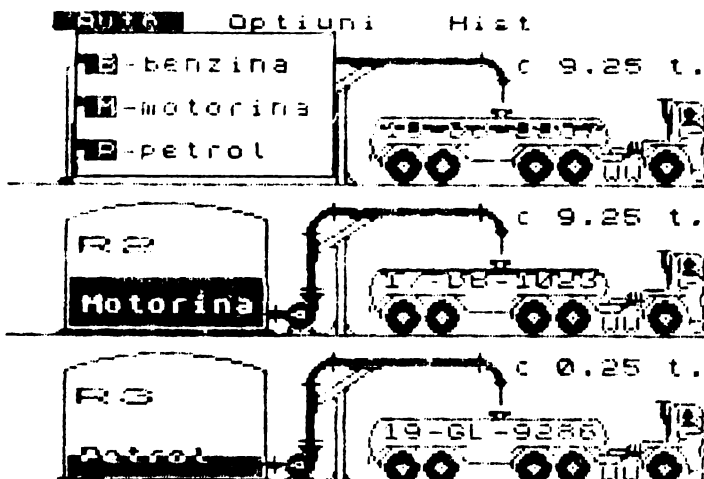


c)

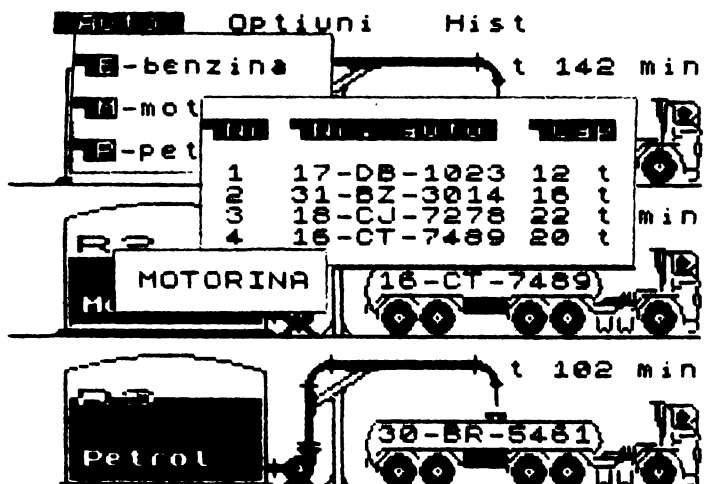


d)

Fig. 14.2. c) ieșirea (animată) a cisternei pline cu motorină; d) intrarea animată a cisternei pentru încărcarea motorinei;

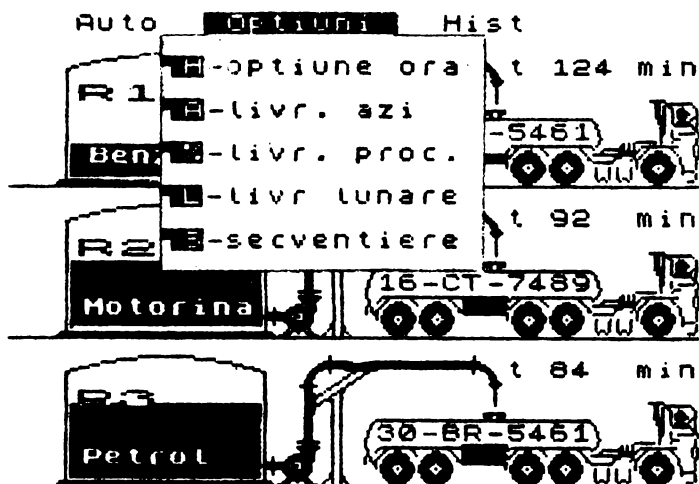


e)

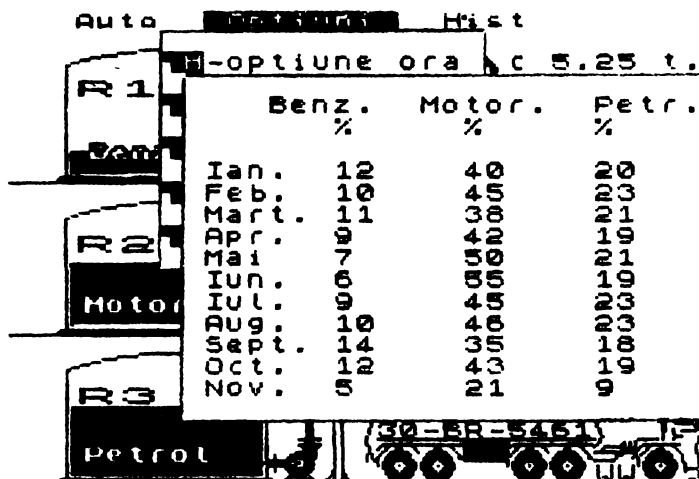


f)

Fig. 14.2. e) apelarea modului „Auto” din meniul principal; f) apelarea opțiunii „MOTORINA” din submeniul corespunzător;

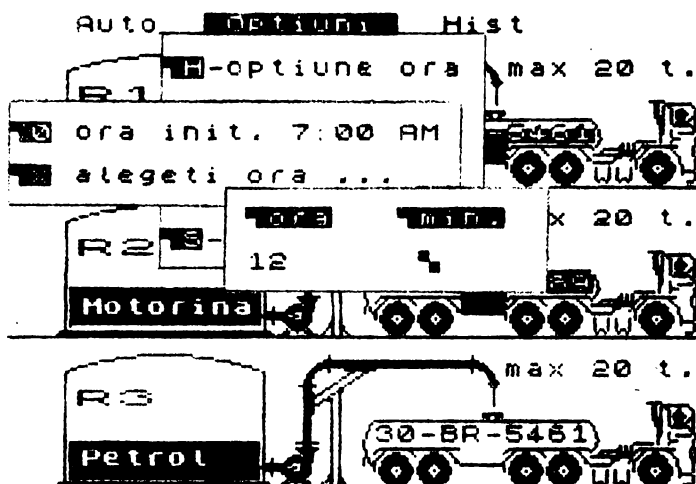


g)

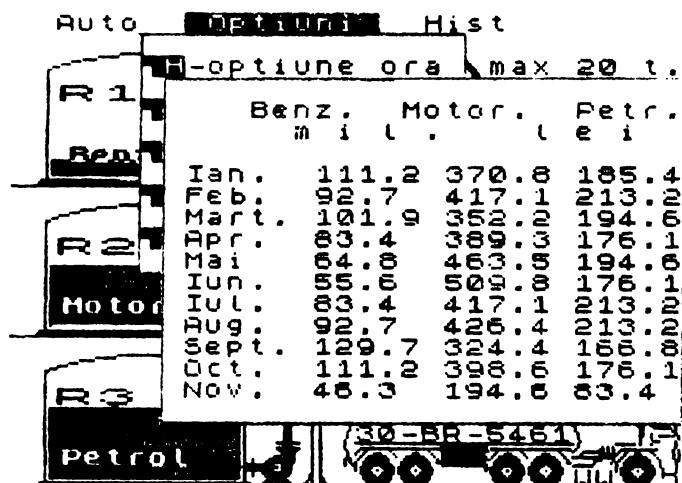


h)

Fig. 14.2. g) apelarea modului „optiuni” din meniul principal; h) apelarea optiunii „%” din submeniul corespunzător;

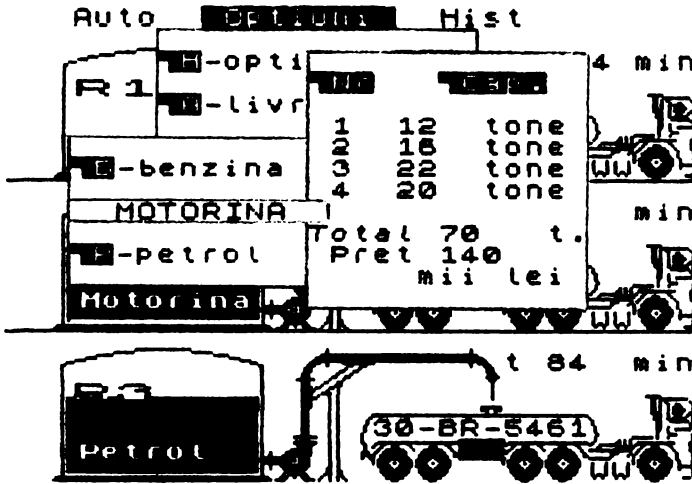


i)

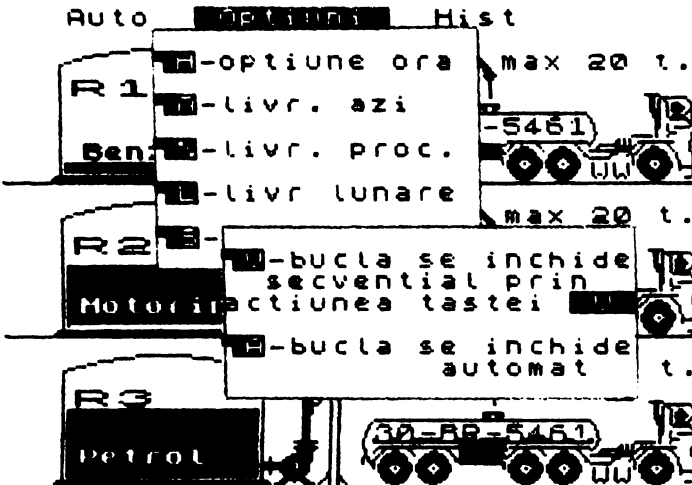


ii)

Fig. 14.2. i) apelarea modului „Opțiune oră”; ii) apelarea modului „L”; se afișează situația livrărilor lunare din anul în curs;



k)

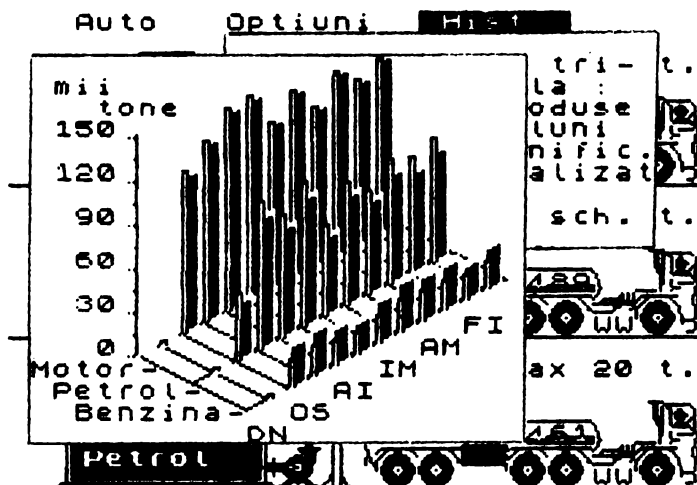


l)

Fig. 14.2. k) apelarea modului „A”; se afișează cantitățile livrate pină la ora curentă, cantitatea totală livrată și valoarea livrărilor in mii lei; l) apelarea modului „S”; se selectează modul de lucru secvențial (V) sau automat (A);



m)



n)

Fig. 14.2. m) apelarea modului „H” din meniul principal; n) histograma 3D.

Pași printre... conversații și sinteze.

Un dialog autor – editor – cititor (II)

În **conversații** s-a urmărit – în principal – să deprindem cititorul cu:

(1) – o metodă structurată de analiză, proiectare și programare a unor probleme simple pentru rezolvarea asistată de calculator, folosind limbajul interactiv **BASIC**, implementat pe mai multe clase de calculatoare: calculatoare medii-mari, minicalculatoare, microcalculatoare, calculatoare personal-profesionale și personale; s-au folosit felurite căi pentru "implementarea" cunoștințelor (v. pașii de la pag. 8, 9) dar, deși s-a prezentat pe parcurs codificarea în fiecare din cele 8 familii reprezentative de calculatoare (din care 6 familii reprezentate prin calculatoare fabricate în România), **cunoașterea cuvintelor rezervate din variantele de BASIC nu a constituit** – cum am afirmat – **un obiectiv esențial al conversațiilor.**

Sintezele vin să atace (cu excepția sintezei 20, în care sînt incluse toate programele din exemplele conversațiilor 1–14) și alte fațete ale cunoașterii limbajului **BASIC**, ce deschid **noi porți pentru stăpînirea practicii programării** pe familiile utilizate în conversații, prin:

(2) – prezentarea cuvintelor rezervate și a mesajelor de eroare pentru diversele variante de limbaj – cu comentarea exemplificată a unui **BASIC pentru calculatoare medii (ABASIC)** și, mai ales, a cîte unui **BASIC pe calculatoare personale (BASIC-AMSTRAD)** și **pe calculatoare personal-profesionale (GW BASIC pe Felix PC și IBM PC).**

De asemenea s-au dat:

(3) – în afara unui **succint istoric al limbajului, variante performante de BASIC**, ca **MBASIC, ZBASIC, CBASIC, BETABASIC**, sau **particularizări** pe calculatoare personale neutilizate în conversații, ca **APPLE, ATARI, TI, KAPIRO, ZENITH**, prin listarea sistematică a cuvintelor rezervate respective. Dar, **sintezele** își extind aripile și în alte 7 direcții diferite, unele doar "dilatînd", altele depășind cu mult, sfera limbajului:

(4) – cea a **aplicațiilor limbajului în domeniul proiectării asistate și al reprezentărilor geometrice în BASIC**, dar mai ales în cel al **jocurilor pe calculator**, ilustrat de 25 programe sursă pentru calculatoare personale;

(5) – cea a **sistemelor de operare** (date prin caracterizări și comenzi) **tipice** pentru toate clasele de calculatoare, ca UNIX, MS-DOS, CP/M, OS/2, ca și a altora ca DOS-PC, Z-DOS, COMMODORE și APPLESOFT DOS, cu o **detaliere pentru familia CP/M**, disponibilă pentru calculatoare personale pe 8 biți și pentru calculatoare personale-profesionale pe 16 biți, fabricate în România;

(6) – cea a **microprocesoarelor** grupate pe: 8 biți (INTEL 8080, Zilog 80, MOTOROLA 6800, MOS 6502), 16 biți (INTEL 8086, Z 8000), 16/32 biți (MOTOROLA 68000), 32 biți (80386 INTEL) reprezentată prin caracterizări, utilizări la diverse tipuri de calculatoare și prin lista instrucțiunilor;

(7) – cea a altor **20 limbaje de programare de nivel înalt**, reprezentată prin cuvintele rezervate pentru Ada, Algol, C, COBOL, Forth, Fortran, LISP, Logo, PASCAL, Prolog ș.a.;

(8) – cea a **produselor program generalizabile** (Visicalc, Multiplan, Lotus 1–2–3, Framework ș.a.), reprezentată prin caracterizări și cuvinte rezervate, cu o extensie și o legătură cu BASIC pentru dBASE II, III, III Plus;

(9) – cum și, ca un corolar pentru limbaje, **cea integratoare a tuturor cuvintelor rezervate în 20 limbaje de programare de nivel înalt** (inclusiv în 18 variante de BASIC), ca și a celor **10 produse program generalizabile**, într-o prezentare alfabetică a cuvintelor cheie, cu indicarea pentru fiecare a limbajelor în care se utilizează;

(10) – în sfârșit, o **bibliografie** – cercetată/recomandată.

Chiar dacă **pașii noștri** (am vorbit de autor și de redactor pentru că împreună ne-am sucit mințile să selectăm și să parcurgem... cărările optime pentru cititor) **printre conversații și sinteze** s-ar limita la această explicitare a celor **zece direcții**, din care rezultă conținutul complex al sintezelor și se reflectă succint obiectivul – cunoscut de acum – al conversațiilor, **ne-am putea declara mulțumiți** de îndrumările date cititorilor.

Dar... între etapele elaborării iterative și cele ale apariției cărții s-au mai produs și alte evoluții ale tehnicii de calcul, ale informaticii, ale limbajului ș.a., **unele semnificative pe plan mondial**, altele interesante pentru aplicațiile **caracteristice țării noastre** și pentru **opțiunile din prezentele volume**. Așa că **vom continua pașii**, care sînt de fapt „**pentru și către cititor**”, cu reflectarea sintetică a unor variante performante de BASIC pe calculatoare personale de 16 biți apărute în ultimii ani (în continuarea familiilor ATARI, APPLE) ca și a unora pe calculatoare personal-profesionale recente de 16 și 32 biți (ca MACINTOSH, PS/2), cu indicarea noilor microprocesoare utilizate și a sistemelor de operare specifice, ca și a unora din extensiile hardware-software ale sistemelor anterioare, inclusiv perife-

ricele și mediul de lucru. Nu vom uita nici **clasa de superminicalculatoare** reprezentată prin elementul său de bază VAX 11 (continuatorul familiei PDP 11), care are un reprezentant fabricat de curînd în România (CORAL 8730). Vom arunca și "ochiade" asupra tehnicii meniurilor, cum și asupra domeniului **aplicațiilor grafice, legat de proiectarea asistată de calculator** și nu vom termina înainte de a mai reveni la **importanța înțelegerii metodelor de analiză, proiectare și programare structurată pentru asistarea cu calculatorul**, în general, și pentru lucrul în BASIC, în particular.

Cu acești alți pași (v. începutul volumului 2), care continuă **dialogul* autor—editor—cititor**, vom considera îmbrățișate și actualizate toate aspectele importante ale cărții, în... interiorul căreia se află (împovărat?!) cititorul.

* Mai am avea de spus ce... n-am spus. Pînă aici nu ne-am preocupat — și nici nu o vom face — de calculatoarele mari, de supercalculatoare și de eventualele limbaje BASIC ale acestora (aici este, însă, zona unor FORTRAN și LISP performante, sau a unor limbaje specializate), limitîndu-ne plafonul la tradiționalul Felix C, calculator mediu din generația a 3-a, deosebit de răspîndit în țară, cum și, mai ales, la mai noile mini și microcalculatoare din generația 3,5→4. Celor care vor să afle ce s-a mai întîmplat în lume în drumul către generația a 5-a, să știe ce e un calculator vectorial, unul „pipeline” sau unul MIMD, un masiv de multiprocesoare, în sfîrșit un **transputer** (rezultat al tehnologiilor de integrare pe scară foarte largă — VLSI — ce au culminat cu comercializarea primului microprocesor pe un singur cip conceput ca bloc de construcție pentru calculatoarele paralele, transputerul INMOS) dar, mai ales, ce sînt limbajele paralele structurale (DAP FORTRAN, FORTRAN 8X, CMLISP) sau procesuale (OCCAM) și ce ne... așteaptă în viitor, nu ne rămîne decît să le recomandăm cartea **Parallel Computers 2**, de R. W. Hockney și C. R. Jesshope, IOP Publishing Ltd, Anglia—S.U.A., 1988 (aflată în curs de traducere în limba română, în aceeași redacție, la Editura Tehnică).

PARSIA

... (text obscured by noise) ...

... (text obscured by noise) ...

... (text obscured by noise) ...

... (text obscured by noise) ...

... (text obscured by noise) ...

M 118 FELIX C
CORLE LA MIC JUNIOR
FELIX PC SPECTRUM AMSTRAD
COMMODORE INDEPENDENT TPD