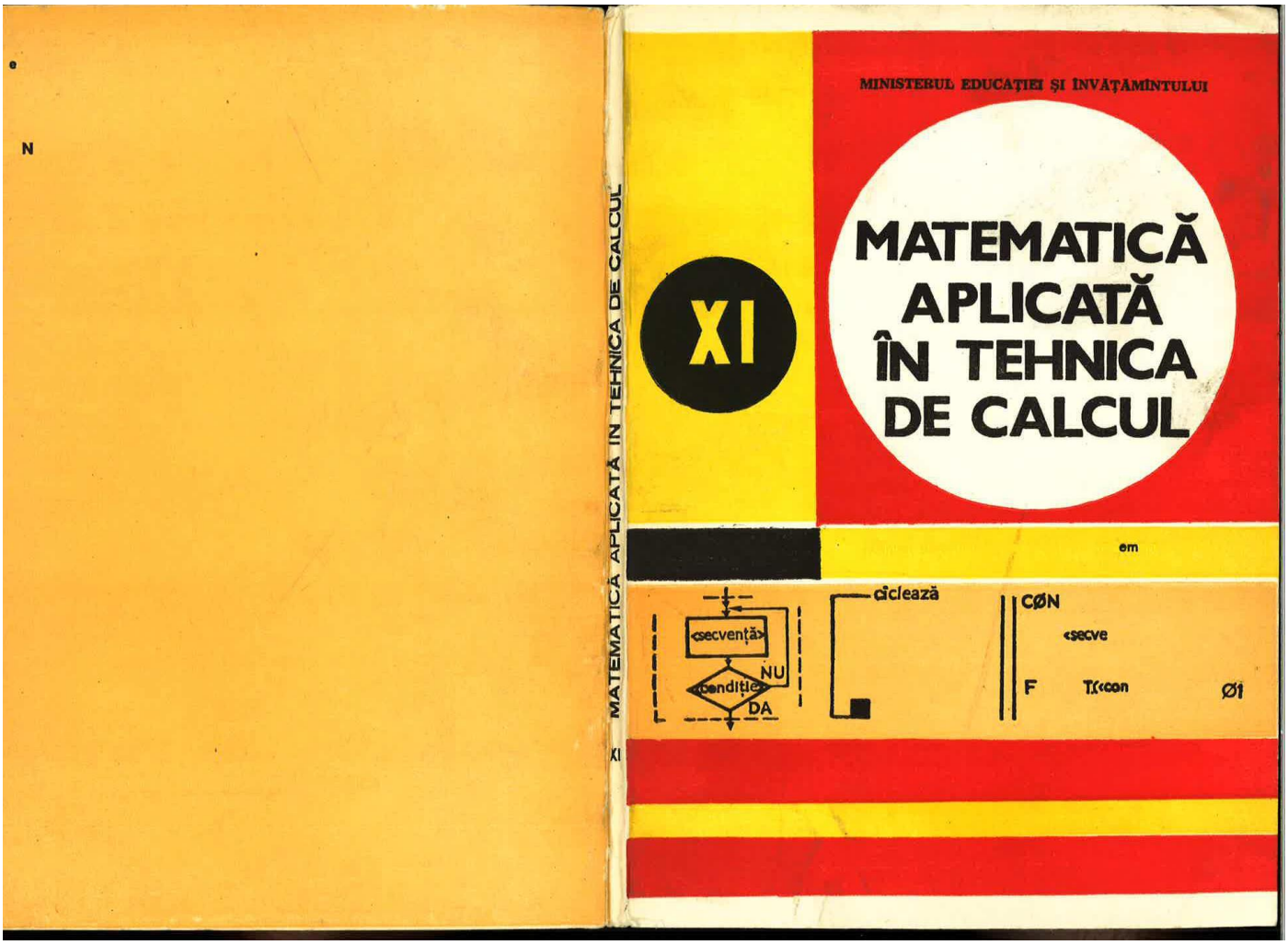
Capitolul ..

# Manualul de „Matematică aplicată la tehnica de calcul” pentru clasa XI –




## Contextul istoric

Ultimii ani ’70 au marcat pentru învățământul românesc un regres, datorat legii din 1978, mai ales dacă o comparăm cu câștigurile legii precedente, din 1968. „Politehnizarea” învățământului și introducerea Consiliului Superior al Educației și Învățământului, care subordona Ministerul Învățământului direct Consiliului Central al partidului, au anunțat „deceniul negru” ce a urmat după 1980.

**An & context**: 1980; introducerea programării în liceele mate-fizică.

**Public-țintă**: clasa a XI-a, 2 ore/săptămână.

**Conținut**: SPDS (model de calculator) → LPS (pseudocod în română) → programare structurată → punte spre FORTRAN.

**Obiectiv pedagogic**: gândire algoritmică înaintea instrumentului; algoritmi în limbaj accesibil + pseudocod + semantica de execuție.

Desigur, politehnizarea ne bucura pe noi, cei de la Politehnică, dar daunele produse științelor umaniste și culturii în general au fost grave. Totuși, calculatoarele au avut atunci de câștigat: una dintre măsurile preconizate a fost difuzarea în masă a cunoștințelor informatice către elevi și studenți.

**Inițiativa lui Octavian Stănășilă**

Introducerea programării în liceele de matematica-fizică din țară a fost rezultatul unei conjuncturi favorabile. Schimbarea de generație din minister adusese ca ministru secretar de stat pe unul dintre colegii mei de la Politehnică, Octavian („Tavi”) Stănășilă. Profesor de matematică, specialist în teoria spațiilor analitice, analiză matematică și teoria semnalelor, cu cinci ani mai mare decât mine, Stănășilă era un profesor carismatic, respectat de studenți și colegi, cu energie și o viziune inovatoare. Cursurile sale combinau rigoarea științifică cu intuiția pedagogică, iar chiar și cele mai complicate rezultate matematice ajungeau să pară firești în expunerea lui.

Stănășilă a reușit, prin relația sa de încredere cu prorectoarea noastră, Suzana Gâdea (și responsabilă de cultură și educație în CC al PCR), să impună introducerea unei ore de tehnică de calcul în toate liceele cu profil matematică-fizică. Într-un regim centralizat, unde manualele școlare erau unice și controlate strict de minister, o asemenea decizie nu putea fi luată decât „de sus”, printr-un amestec de viziune personală și conjunctură politică.

**Cum am ajuns autor de manual de liceu**

Scrierea manualului *Matematică aplicată în tehnica de calcul* nu a fost o alegere personală, ci o însărcinare directă de la minister. Stănășilă, cunoscându-mă ca specialist în informatică, dar și ca matematician, mi-a încredințat redactarea manualului pentru clasa a XI-a.

Manualul se înscria într-o serie de patru manuale care acopereau cele patru clase ale liceului:

- clasa a IX-a – Introducere în calcul și informatică;
- clasa a X-a – Combinatorică aplicată și grafuri;
- clasa a XI-a – Algoritmi (și, în versiunea extinsă, elemente de calculabilitate) și FORTRAN;
- clasa a XII-a – Elemente avansate de programare și aplicații.

Scrierea lor a fost încredințată unor profesori și cercetători din domeniul informaticii. Pentru mine, scrierea manualului venea la capătul unui deceniu în care mă afirmasem în informatică și îmi susținusem teza de doctorat. În viziunea mea, FORTRAN-ul era un limbaj de programare deja depășit: îl predasem peste cinci ani la Politehnică, dar în acel moment predam Pascal și C, știind că ele reprezintă viitorul imediat al programării.

Mi-am dat seama imediat că trebuia să împart responsabilitatea cu cineva din zona institutelor de cercetare. L-am convins pe Petre Preoteasa, matematician și informatician pasionat, cercetător la ICI, bun cunoscător al limbajului FORTRAN de pe vremea profesorului Grigore Moisil. Împărțirea a fost naturală: eu m-am ocupat de prima parte, dedicată algoritmilor (și fundamentelor de gândire formală), iar el de partea a doua, orientată spre FORTRAN. FORTRAN fusese ales deoarece era limbajul standard al vremii în mediul industrial (alături de COBOL, în multe întreprinderi).

## Conținutul manualului

**Cap. I – ELEMENTE DE BAZĂ ALE PROGRAMĂRII CALCULATOARELOR**

Structura primei părți a manualului era următoarea:

1. Structura și funcționarea sistemului SPDS
2. „Calculatorul C”: funcționare, stare, programarea calculatorului C
3. Reprezentarea algoritmilor în LPS:
    - definiția limbajelor de programare
    - instrucțiuni de bază
    - secvență de instrucțiuni
    - reprezentarea algoritmilor
    - **probleme**

4. Instrucțiunile de control LPS:
    - instrucțiune de decizie
    - cicluri cu condiție / cu contor
    - tablouri
    - **probleme**

Ceea ce mi se părea esențial era să le formez elevilor **o minte de programator**, nu doar deprinderi într-un limbaj trecător.

**Cap. II – PROGRAMAREA ÎN LIMBAJUL FORTRAN**
Petrică Preoteasa a redactat partea a doua cu o orientare practică. A prezentat tipurile de date și instrucțiunile FORTRAN, expresiile și structurile de control, completate cu exerciții și aplicații. Structura părții a doua a manualului era următoarea:

1. Date, instrucțiunea de atribuire, expresii
    a. Etapele rezolvării unei probleme în FORTRAN
    b. Declarații, instrucțiuni de atribuire
    c. Exerciții
2. Principalele instrucțiuni:
    a. IF logic / IF aritmetic
    b. GOTO, CONTINUE, STOP, PAUSE, END
    c. intrare/ieșire
    d. Instrucțiunea de citire
    e. Instrucțiunea de scriere
3. Instructiunea FORMAT
    a. Reguli de alcătuire a listei FORTRAN
    b. Exemple de programe FORTRAN
4. Facilități de programare in limbajul FORTRAN
    a. Instrucțiunea DO
    b. Subprograme
5. Exerciții
   Rezolvări

Dacă-l privim „de sus”, manualul îi spune unui elev:

1. *Înțelege* întâi ideea de sistem de prelucrare a datelor și componentele unui **calculator abstract**
2. *Învață* să gândești o problemă în termeni de **algoritmi**, si descrie-o într-un limbaj didactic (LPS).
3. *Vezi* apoi cum se scriu aceiași algoritmi într-un **limbaj real** (FORTRAN).

Din punct de vedere pedagogic, conținutul manualului astfel comceput era aproape ideal, valabil chiar și după multe standarde ulterioare.

Pentru un liceu teoretic / profil mate-fizică / tehnic, cu 2 ore pe săptămână tot anul și elevi in clasa XI deja obișnuiți cu gândirea matematică, volumul părea ambițios, dar realist: două capitole mari pentru un an întreg trebuiau bine etapizate. Principiu de bază: nu se „bagă” FORTRAN-ul până când elevul nu are „reflexul” cognitiv al LPS compatibil cu stilul programării structurate: secvență–selecție–ciclu.

## Prezentarea părții mele: algoritmi și calculabilitate

Voi prezenta în acest text partea I a manualului „Matematică aplicată la tehnica de calcul” (clasa a XI-a, 1980) dorit ca un parcurs de formare a gândirii algoritmice bazat pe o mașină abstractă (SPDS) și pe pseudocodul LPS, înainte de trecerea la FORTRAN. Miza este să formeze reflexe de proiectare (secvență–selecție–ciclu) și o înțelegere operațională a calculului, nu doar deprinderi într-un limbaj trecător. Contextul, structura și lecțiile pedagogice sunt distilate pentru utilizarea în 1980.

In manual am propus o abordare originală:

1. Am imaginat un **model abstract, intuitiv, de mașină de calcul** numit SPDS (Sistem de Prelucrare a Datelor Simplu),
2. Am introdus un model simplificat al unui calculator, calculatorul abstract C, aflat in centrul SPDS, cu componentele principale și cu funcționarea lui.
3. Am inventat un **pseudocod în limba română**, LPS (Limbaj de Programare Simplificat), un limbaj pentru a scrie algoritmi, independent de un calculator concret, tip pseudocod, dar în românește, cu care elevii să

învețe să gândească algoritmic și să descrie ce gândesc, să descompună problemele și să structureze soluțiile. Era gândit ca o punte între matematică și informatică, între logică și cod. Pentru ca elevii să fie introduși în programarea structurată am introdus principalele operații: secvența, instrucțiunea condițională și ciclul – elementele fundamentale ale gândirii algoritmice. Algoritmii ar fi putut fi reprezentați grafic, prin diagrame de flux, dar am preferat ca descrierea lor sa fie textuala, mai apropiată de programele reale. La introducerea instrucțiunilor LPA am schițat însă semantica lor cu diagrame de flux.

**Conținutul primei părți**

1. **SPDS – sistemul de prelucrare a datelor simplu**

Sistemul de prelucrare a datelor simplu (SPDS) din Fig. 1.1 este un model (o imagine simplificată) al sistemelor de calcul reale.

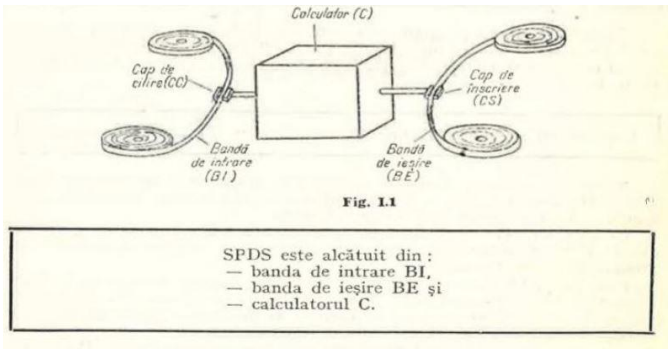



Fig. 1.1

SPDS este alcătuit din :
— banda de intrare BI,
— banda de ieșire BE și
— calculatorul C.

Fig. 1.1 Sistemul de calcul SPDS

Benzile de intrare și ieșire (BI, respectiv BE) servesc la citirea datelor inițiale în vederea prelucrării lor de către calculatorul C și respectiv la înscrierea rezultatelor calculelor realizate de C.

BI și BE sunt modele ale fișierelor de intrare (pachetul de cartele perforate conținând datele inițiale) respectiv de ieșire (succesiunea de linii scrise la imprimantă) ale sistemelor de prelucrare automată a datelor reale.

Am restrâns interpretarea datelor înscrise pe benzi la următoarele tipuri :

— numere întregi,

— numere reale,

— valori logice,

— șiruri de caractere, numite și texte (le vom prezenta cuprinse între ghilimele sau apostrofuri),

Funcționarea SPDS într-o prelucrare poate fi văzută și ca rol o transformarea a fișierului de intrare în fișierul de ieșire.

2. **Calculatorul C – modelul unui calculator**

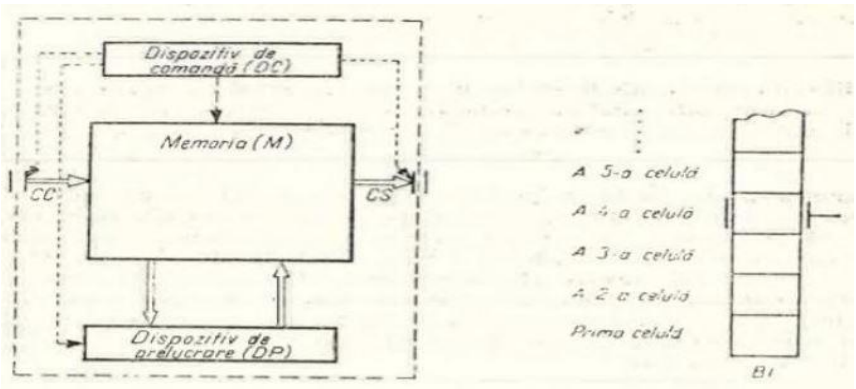



Fig. 1-3 Structura calculatorului C

Fig. 1.4 Banda de intrare cu celula curenta 4

Calculatorul C are următoarele părți componente, după cum se vede în Fig. 1.3::

— capul de citire CC,

— capul de înscriere CS,

— memoria M,

— dispozitivul de prelucrare DP și

— dispozitivul de comandă DC.

Fig. 1.4 prezinta o banda de intrare segmentata in celule. Avansul benzii prin fața capului de citire se face sacadat, celulă cu celulă, într-un singur sens. Starea capetelor este dată de numărul cc care reprezintă poziția lor pe benzile respective.

Memoria M este alcătuită dintr-o mulțime de celule (Fig. 1.5) ce pot găzdui date din cele patru tipuri menționate. În celulele memoriei pot fi „înscrise“ date și din celule pot fi „citite“ date. Într-o celulă a memoriei nu poate exista decât cel mult o dată (de tip numeric, logic sau șir de caractere!),

Starea memoriei în cursul unei prelucrări este dată de mulțimea perechilor (adresă ; valoare) ale tuturor celulelor zonei de date.

Dispozitivul de prelucrare DP realizează calculul valorii unei funcții pentru care se cunosc valorile argumentelor. DP este reprezentat schematic mai jos (Fig. 1.6, 1.7). Valorile argumentelor reprezintă „intrarea“ în DP, iar valoarea funcției „ieşirea“ din DP, Operațiile de bază ale calculatorului C sunt: citirea, atribuirea, scrierea şi oprirea.

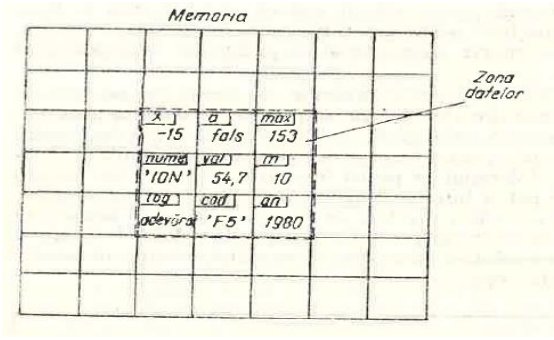



Fig. 1.5. Structura memoriei

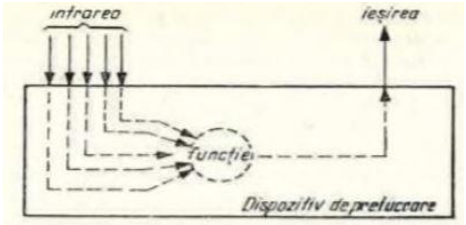



Fig. 1.6 Calculul unei funcții de către DP

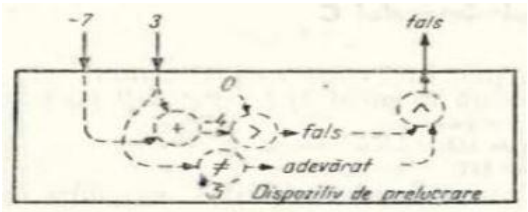



Fig. 1.7 Realizarea calculului unei expresii de către DP

**Starea calculatorului C**

La un anumit moment al funcționării calculatorului lui în cadrul unei aplicații este suficient să cunoaștem:

a) poziția CC pe BI (deci starea CC),

b) poziţia CS pe BE (deci starea CS),

c) starea memoriei, adică zona datelor aplicației ca mulțime a perechilor (adresă, valoare).

In Fig. 1.9 se prezintă doua stări distincte, consecutive ale calculatorului despărțite de o operație de citire.

Operațiile de bază ale DP sunt: citire, atribuire şi scriere modifică starea calculatorului astfel: ! A, ! at

a) citirea modifică poziţia capului de citire și valoarea celulei din A afectată de citire ; ET RP

b) atribuirea modifică valoarea celulei din M unde se depune rezultatul;

c) scrierea modifică poziția capului de înscriere.

Fig. 1.9 prezinta doua stări consecutive al calculatorului determinate de o operație de citire.

a) Efectuarea unei operații de citire a valorii celulei curente de pe BI şi depunerea ci într-o celulă dată din M necesită din partea DC comenzi pentru executarea următoarelor acțiuni :

— extragerea de către CC din celula curentă a lui BI a informației con: ținute de aceasta,

-— plasarea valorii în celula din M precizată în operație (deci: modificarea valorii celulei respective),

— avansul cu o celulă a lui BI (deci: modificarea poziției CC).

b) efectuarea unei operații de atribuire a valorii unei funcții date unei celule din M, de asemenea dată, necesită următoarele acțiuni comandate de DC:

-— copierea valorilor din celulele lui M cc reprezintă argumentele funcției,

— evaluarea funcției de către DP folosind valorile argumentelor extrase din M,

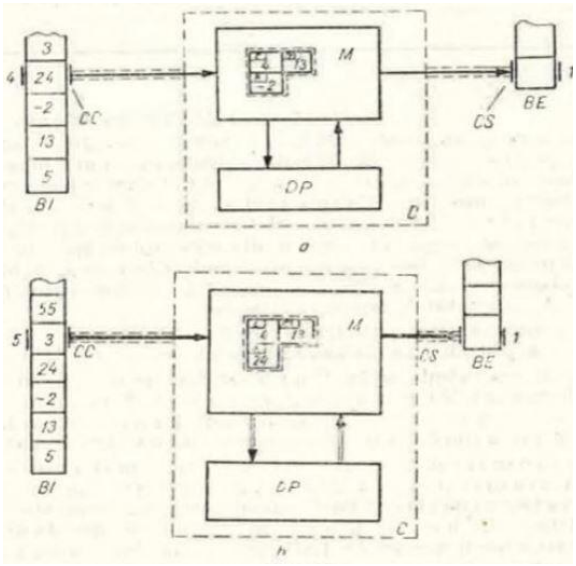
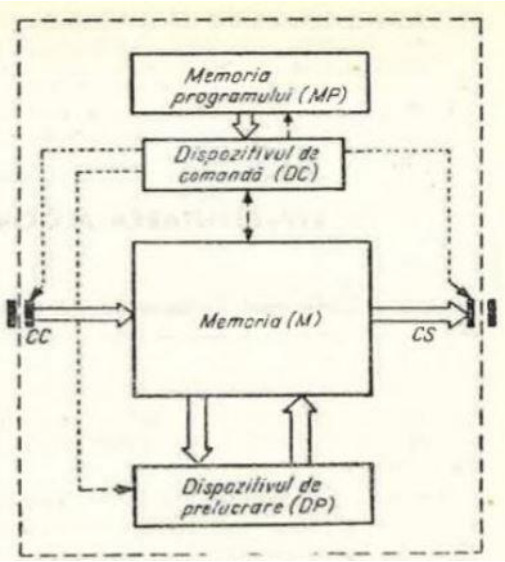



Fig. 1.9. Secvență a două stări ale calculatorului




— înscrierea valorii rezultatului în celula din M precizată în operație (deci: modificarea valorii celulei din dl).

c) Efectuarea unei operații de scriere a unei valori dintr-o celulă dată din M în celula curentă de pe banda BE necesită următoarele acțiuni comandate de DC:

-— copierea din celula dată a lui M a valorii conținute,

— înscrierea valorii în celula curentă a benzii BE (deci : modificarea conținutului celulei curente a lui BE),

— avansul cu o celulă a lui BE (deci: modificarea poziției CS).

d) Operația de oprire are ca efect blocarea funcționării calculatorului C de către DC,

**Programarea calculatorului C**

Am denumit *date de intrare* șirul datelor aflate în celulele citite ale BI şi *rezultate* șirul datelor aflate în celulele scrise ale BE la sfârșitul prelucrării (după execuția operației de oprire). Transformarea datelor de intrare în rezultate este consecința execuției unei secvențe de operații de bază.

De obicei, mulțimea datelor de intrare pentru care putem executa o anumită secvență de operații este dată prin specificarea problemei. Să notăm cu D această mulțime numită şi domeniul secvenței de operații,

Să considerăm și mulțimea tuturor perechilor (date de intrare, rezultate) pentru datele de intrare din D. Să notăm cu K această mulțime. Ea se numește *comportare externă* a secvenței de operații pentru domeniul D, Comportarea externă este a doua componentă a specificării oricărei probleme,

Dându-se o pereche (D, K), a *programa* calculatorul C înseamnă a determina o secvență de operații de bază care, executate în ordinea secvenței pentru toate datele din D, să aibă mulțimea K drept comportare externă.

Programul este memorat într-o zonă a programului MP aflată în memoria calculatorului, zonă distinctă în cazurile reale doar din punct de vedere al modului de interpretare informațiilor din ea. Dispozitivul de comandă își extrage din zona respectivă instrucțiunile una după cealaltă, conform unor reguli bine stabilite.

În cazul calculatorului C putem completa schema din figura 1.3 cu o memorie a programului MP unde utilizatorul depune instrucțiunile pro gramului, un sir de instrucțiuni pe care calculatorul va trebui sa le execute (fig. 1.10). Dispozitivul de comandă DC extrage de aici următoarea instrucțiune de executat.. Rolul operațiilor de control va fi prezentat în paragraful următor când se vor descrie instrucțiunile condiționate și de ciclare.

## REPREZENTAREA ALGORITMILOR ÎN LIMBAJUL LPS

### 3. Limbajul LPS

Aceasta secțiune are următoarea structură:

i. definirea limbajelor de programare,
ii. introducere LPS ca limbaj pentru reprezentarea algoritmilor;
iii. introducerea instrucțiunilor de bază,
iv. instrucțiuni de control (condițional (Fig. 1.18), cicluri), diagrame pentru semantica instrucțiunilor, algoritmi;
v. utilizarea tablourilor, exemple cu tabele și diagrame de memorie.

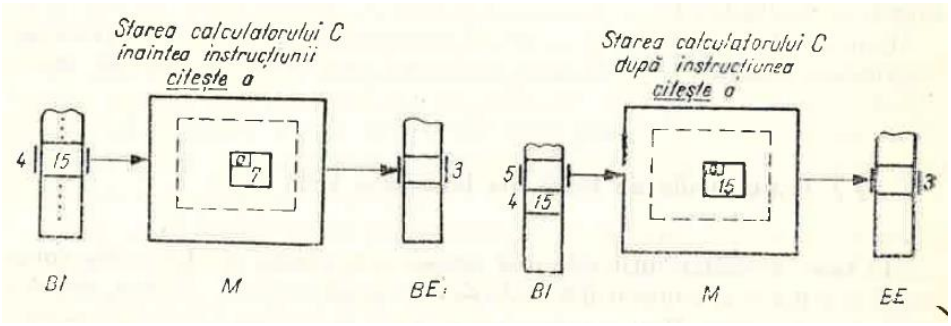
i. În cazul limbajelor de programare, definirea lor trebuie să fie precisă. Prin analogie cu limbajul natural un program este o „propoziție” într-un limbaj de programare, El descrie o anume funcționare a calculatorului. Care este această funcționare rezultă din interpretarea definiției rezultă din interpretarea programului respectiv conform limbajului. Definiția trebuie deci să fie clară pentru ca orice program astfel ca acesta sa

fie la fel înțeles de orice utilizator și anume exact în sensul pe care calculatorul îl atribuie programului respectiv.

ii. Orice element component al unui limbaj de programare trebuie definit pe două niveluri: ,
   a) cum se formează (scrie) corect (nivelul sintactic),
   b) ce semnificație are el pentru utilizator, semnificație înțeleasă ca efect produs asupra funcționării calculatorului (nivelul semantic).

Regulile de formare a programelor corecte alcătuiesc *sintaxa limbajului de programare*.
Regulile ce determină semnificația programelor corecte alcătuiesc *semantica limbajului de programare*. !
În cele ce urmează vom prezenta un limbaj folosit pentru descrierea algoritmilor: limbajul LPS.
Limbajul LPS va fi folosit ca o fază intermediară în scrierea programelor, alternativă posibilă, de: cele mai multe ori preferabilă, schemelor logice.

iii. Manualul prezinta apoi instrucțiunile de bază ale limbajului LPS: **citește**, **scrie**, **atribuie**. Fig. 1.13 prezinta efectul operației de citire. Apoi se prezinta *secvența de instrucțiuni* , declarațiile de tip și cele de tablouri și înregistrări.




1.15 Efectul operației de citire asupra stării calculatorului

iv. Manualul prezinta apoi instructiunile de control in LPS: condițională **dacă** .... **atunci**.... ■ , **dacă** .... **atunci** ... **altfel** ... ■ și ciclurile **cicleaza** ... **până**, **cat timp** ... ■ , **pentru** ... **execută**.....■

În f(Fig. 1.18 este prezentat un exemplu al modului în care se simulează pe un calculator real execuția instrucțiunii condiționale.

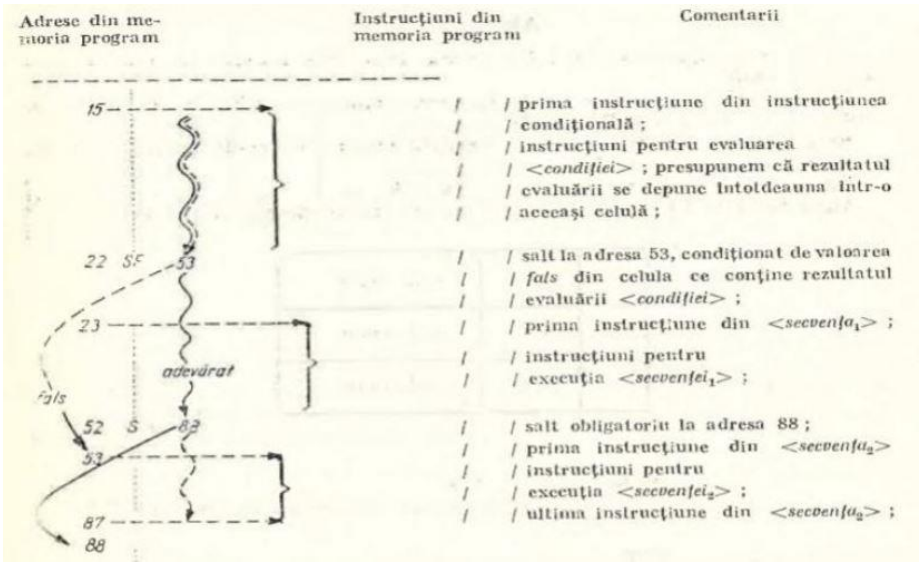



Fig. 1.18 Exemplu al modului În care se simulează instrucțiunea condițională pe un calculator real.

Fara a insista asupra introducerii limbajului LPS, iată un exemplu de algoritm descris in LSP:

**Algoritmul 3 (Algoritmul lui Euciid) - Exemplu de pseudocod**

Să se descrie în LPS algoritmul pentru determinarea celui mai divizor comun a două numere întregi strict pozitive.
Date inițiale: două numere întregi strict pozitive pe banda BI.
Rezultate: un număr întreg aflat pe BE.

| | | |
|---|---|---|
| $n$ | var. simplă | valori întregi [1, 100] |
| $a$ | tablou 1 dimens. (100) | valori reale |
| $i$ | var. simplă (contor) | valori întregi |
| $t$ | var. simplă | valori reale |
| $sem$ | var. simplă | valori logice |
| $m$ | var. simplă | valori întregi |

```
citește n
citește (a(i), i = 1, n)
┌ ciclează
    atribuie sem ← adevărat
    atribuie m ← n − 1
  ┌ pentru i = 1, m execută
    ┌ dacă a(i) < a(i + 1) atunci
        atribuie t ← a(i) & a(i) ← a(i + 1) & a(i + 1) ← t
        atribuie sem ← fals
    └ ■
  └ ■
  până sem
└ ■
scrie (a(i), i = 1, n)
stop
```

Fig. 1.22 Algoritmul lui Euclid în LPS

**Soarta manualului**

Am participat în acei ani la întâlniri cu profesorii de liceu care urmau să predea disciplina. Am încercat să le explic rostul părții de algoritmi. Nu îmi dau seama cât am reușit să îi conving. Cei mai mulți erau tineri ingineri, nu matematicieni, și simțeau dorința de a preda cât mai repede FORTRAN-ul, singurul considerat „util”. Pentru ei, partea mea apărea ca „prea multă teorie”, greoaie și, în aparență, inutilă. În plus, se pare că materia era multă pentru o singură oră pe săptămână. In anii ’80 s-au înființat și licee de informatica si, evident, manualul a fost extins si pentru aceste licee. Cred ca acolo scuza predării cât mai repede a FORTRAN-ului s-a accentuat.

După Revoluție, învățământului preuniversitar i-a lipsit stabilitatea și strategia. A suferit multe aventuri provocate de Ministerul Educației, ceea ce unui sistem educațional nu-i face bine. Nenumărați miniștri s-au perindat pe la șefia Ministerului (în primii 10 ani după Revoluție au fost 15 miniștri ai Educației), fiecare încercând să își impună propria viziune, ceea ce a adus daune grave sistemului de învățământ. In această lungă "expediție" pe spinarea elevilor, legile educației s-au modificat de nenumărate ori. S-au schimbat profiluri și specializări ale liceelor, s-au introdus noi materii și, desigur, manuale. La un moment dat, au apărut manualele „alternative” care, curând, în informatică au început să abunde. Evident, manualele de Matematica aplicată in tehnica de calcul din vremea lui Ceaușescu au dispărut pentru că a dispărut materia, fiind înlocuite cu manuale de Informatică, materie introdusa în liceele de profil cu un număr impresionant de ore (4 – 5 ore/săptămâna în unele clase si profiluri). Mai mult, elevii au devenit între timp mai precoci decât pe vremea mea, sau, oricum, mai obișnuiți cu calculatoarele. De aceea, cunoștințele pe care le-am inclus în manualul de clasa XI au devenit subiectul manualelor de clasa IX, cu mai multe ore la dispoziție.

**Învățămintele acestei experiențe**

Cu acest manual, îmi propusesem să ofer elevilor un instrument de gândire, nu doar o listă de comenzi de memorat, dar nu cred că am reușit să impun această idee. Mi-a rămas, din acest episod al manualului de informatică, două reflecții simple:

1. Adevărata bătălie, în educația informaticii, nu este numai între manuale, ci mai ales între două feluri de a înțelege formarea elevilor sau studenților. Ambiguitatea rezultatelor obținute în promovarea manualului meu reflectă tensiunea dintre două orientări pedagogice în predarea programării calculatoarelor:
    - Una, independentă de limbaj, privilegiază formarea gândirii algoritmice, logice, și abstractizarea;
    - Cealaltă, pragmatică, privilegiază utilitarismul imediat și predarea unui instrument comod; în cazul meu concret, FORTRAN.

    Privind în urmă, sunt convins că nu am greșit alegând ambele căi, dar puse față în față. După mine, exact acea „teorie” din prima parte, simplificată la maxim, dar nu redusă în esența ei, era cheia „bunului programator”. Limbajele de programare se schimbă (de altfel, FORTRAN-ul a dispărut din școli) dar algoritmica și gândirea logică rămân fundamentul pregătirii oricărui programator și ele pregătesc elevii să navigheze printre paradigmele tehnologice trecătoare.
2. Încă o dată, am realizat cât de important este profesorul: în afara carismei personale, indiscutabile, dacă el nu crede în ceea ce predă, transmiterea cunoștințelor eșuează.

**Concluzie mai generală**

Manualul *Matematică aplicată în tehnica de calcul* devine și un caz reprezentativ pentru modul în care educația informatică a pătruns în România comunistă: prin fereastra de oportunitate creată de câțiva oameni luminați, dar într-un mediu nepregătit în prealabil, inert sau chiar ostil inovației pe termen lung. Scenariul cărții mele ilustrează mecanismul tipic al aplicării unei inovații în acea perioadă:

- un nucleu vizionar (Stănășilă) aduce ideea;
- un cadru politic (Suzana Gâdea) o sprijină;
- sistemul birocratic și inerția practicienilor o erodează sau o sabotează, până când inițiativa se stinge.

Cazul arată că, chiar și în regimul comunist, inovația era posibilă atunci când câțiva oameni vizionari reușeau să convingă factorii politici. Totuși, fără o cultură instituțională care să susțină schimbarea, aceste inițiative se stingeau repede, înlocuite de soluții facile și conforme cu inerția sistemului. Lecția rămâne actuală: în educație, adevărata reformă nu stă doar în programe sau în manuale, ci în capacitatea de a forma gândirea învățăceilor pe termen lung, dincolo de utilitarismul imediat.