

## LABORATOR NR. 1

### Reprezentarea algoritmilor prin scheme logice.

#### 1. Algoritm

Algoritmul este o „rețetă” pentru rezolvarea unei anumite probleme. Urmarea pas cu pas a indicațiilor din „rețetă” permite transformarea informațiilor inițiale (cele pe care le avem la dispoziție în momentul începerii rezolvării problemei – altfel spus a „ingredientelor”) în rezultate („produse finite”). Informațiile inițiale le mai numim și **date (informații) de intrare**. Rezultatele obținute în urma urmării pașilor dintr-o „rețetă” le numim **date (informații) de ieșire**. Indicațiile din „rețetă” sunt **pașii algoritmului**.

Un **program** poate fi format din unul sau mai mulți algoritmi. Și pentru program avem **date de intrare** (sunt informațiile de care dispunem la începutul programului) și **date de ieșire** (sunt informațiile obținute în urma rulării programului pe calculator).

Rezolvarea unei probleme presupune parcurgerea a două etape:

1. etapa de analiză a problemei (înțelegerea problemei, stabilirea datelor de intrare și a datelor de ieșire, elaborarea și descrierea algoritmului sau a algoritmilor necesari pentru rezolvarea problemei);

2. etapa de implementare a algoritmilor descriși în etapa de analiză.

Informațiile inițiale (datele de intrare) sunt introduse în program prin **citire**. Dacă, în urma analizei problemei, rezultă că, pentru rezolvarea problemei, avem nevoie de anumite informații, acestea trebuie **citite** (în această etapă) de la dispozitivul de intrare standard care este tastatura. Datele de intrare pot fi valori numerice (întregi sau reale), caractere sau șiruri de caractere.

Informațiile care rezultă dintr-un program (datele de ieșire) trebuie comunicate utilizatorului prin **scriere** pe dispozitivul standard de ieșire care este monitorul. Afișarea rezultatelor pe monitor se face, de obicei, prin intermediul unor mesaje care înglobează datele de ieșire.

Informațiile inițiale ale programului (cele pe care le citim de la tastatură) sunt stocate (sau memorate) în memoria sistemului de calcul în anumite zone. Aceste zone de memorie în care vor fi memorate, mai întâi informațiile inițiale ale programului și

apoi toate rezultatele intermediare au atașate „etichete” cu numele dat informației respective. De exemplu, dacă hotărâm ca o anumită dată de intrare să aibă numele **medie\_aritmetica**, atunci în memorie se va găsi o zonă în care se vor stoca valorile numerice ale acestei informații și care va avea această „etichetă”. Zonele de memorie care au atașat (prin program) un **nume sugestiv** și în care sunt stocate informațiile asociate numelui respectiv se numesc **variabile**. Valorile stocate în aceste zone de memorie se pot modifica pe parcursul programului.

Într-un program, pe lângă variabile, mai există și informații ale căror valori nu se pot modifica pe parcursul unui program. Aceste informații pot avea de asemenea asociat un nume. Și în alte domenii (matematică, fizică, chimie) avem astfel de informații pe care le numim constante. De exemplu, în matematică avem constanta notată cu  $\pi$  și care este egală cu 3.14159...

*Observație: în programare pentru exprimarea numerelor reale nu vom folosi ca separator între partea întreagă și cea fracționară virgula (nu vom mai scrie 3,14158), ci vom folosi ca separator punctul (vom scrie 3.14159).*

Pentru obținerea rezultatelor finale (datele de ieșire), precum și a unor rezultate intermediare, variabilele și constantele care apar într-un algoritm pot fi incluse în expresii aritmetice, logica sau relaționale.

**Expresiile aritmetice** sunt formate cu ajutorul operatorilor aritmetici: +, -, \*, / și %. Operatorul **%** îl mai numim și operatorul **modulo**, se aplică numai numerelor întregi pozitive, iar expresia  $a \% b$  are ca rezultat restul împărțirii lui **a** la **b**.

**Expresiile logice** sunt formate cu ajutorul operatorilor logici: non (notat  $\neg$ ), ȘI (notat  $\wedge$ ) și SAU (notat  $\vee$ ). Expresiile logice au ca operanzi propoziții (elementare sau compuse) și ca rezultat valoarea de adevăr a propoziției formate cu ajutorul lor (deci rezultatul poate avea două valori posibile: ADEVĂRAT (TRUE) sau FALS)).

**Expresiile relaționale** sunt formate cu ajutorul operatorilor relaționale: <, ≤, >, ≥, = și ≠. Expresiile relaționale sunt, de fapt, propoziții, deci au ca rezultat valoarea de adevăr a propozițiilor respective.

O altă operație importantă într-un program este **atribuirea**. Simbolul prin care vom nota această operație este  $\leftarrow$ . Forma generală pentru această operație este următoarea:

$$v \leftarrow e$$

în care **v** este o variabilă, iar **e** este o variabilă, o constantă sau o expresie.

Cum se execută această operație? Se evaluează **e** (adică se calculează rezultatul expresiei specificate), iar rezultatul va fi stocat (memorat) în zona de memorie asociată variabilei **v**.

De multe ori într-un program pot apare atribuiri de forma:

$$v \leftarrow v + 1$$

În acest caz modul de execuție este următorul: se aduce din zona de memorie corespunzătoare variabilei **v** valoarea găsită acolo; la această valoare se adună valoarea 1, iar rezultatul se trimite din nou în zona de memorie asociată variabilei **v**. După execuția acestei atribuiri, în zona de memorie asociată variabilei **v** se va afla o valoare mai mare cu o unitate față de valoarea inițială a variabilei **v**.

Inițializarea unei variabile înseamnă prima „umplere” a zonei de memorie asociate variabilei considerate cu o valoare. Această inițializare se poate face prin citire (când zona de memorie asociată variabilei „se umple” cu valoarea citită de la dispozitivul de intrare) sau prin atribuire (când zona de memorie asociată variabilei „se umple” cu rezultatul expresiei din dreapta operatorului de atribuire).

## 2. Reprezentarea algoritmilor prin intermediul schemelor logice

Un algoritm poate fi reprezentat sub forma unei diagrame formate din figuri geometrice. Fiecare figură geometrică din diagramă corespunde unui pas (unei etape) din algoritm. Figurile geometrice din diagramă sunt legate prin linii terminate cu săgeți care arată ordinea de execuție a pașilor algoritmului. Această ordine de execuție a pașilor algoritmului (sau a programului) se numește structura de control a programului.

Reprezentarea grafică bidimensională a unui algoritm cu ajutorul unor simboluri speciale se numește schemă logică program sau schemă logică (organigramă).

### 2.1. Simboluri folosite în schemele logice

Pentru reprezentarea schemelor logice au fost stabilite anumite simboluri geometrice care sugerează prin forma lor diferitele operații care trebuie executate și ordinea de execuție. Ea reprezintă în același timp și o imagine a fluxului de informații care există atunci când calculatorul execută programul bazat pe algoritmul descris. Schemele logice au fost inventate de Herman Goldstine (S.U.A.) în 1946.

Alcătuirea schemei logice permite ca programatorul să înțeleagă logica programului, să realizeze modularizarea programului (prin împărțirea programului în părți mai mici și mai detaliate), să verifice îndeplinirea tuturor condițiilor posibile din program, să traducă algoritmul într-un limbaj de programare (de exemplu în limbajul C).

Simbolurile folosite la construirea schemelor logice se pot împărți în mai multe categorii:

a. *Simbolul linie și vârf de săgeată*. Este prezentat în Figura 1.

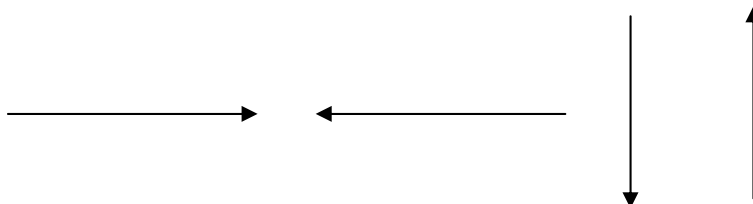


Figura nr. 1. Simboluri pentru direcția fluxului de informație,

Acesta simbolizează direcția fluxului de informație în calculator și succesiunea diferitelor operații. Liniile de flux se pot încrucișa. Liniile care se încrucișează nu înseamnă că sunt conectate. Dacă ele sunt conectate trebuie să apară vârfuri de săgeți lângă punctul de joncțiune pentru a marca conexiunea. Se reprezintă printr-un segment de dreaptă de orice lungime care trebuie să lege alte două simboluri. Direcția normală a fluxului este de la stânga la dreapta și de sus în jos. Când direcția fluxului este de sens opus direcției normale se folosesc vârfuri de săgeată. Pentru mai multă claritate mulți programatori folosesc vârfurile de săgeată în toate cazurile.

b. *Simbolul proces* (Figura nr. 2) reprezintă execuția unei (sau mai multor) operații de prelucrare a uneia sau mai multor informații în urma căreia se schimbă valoarea, forma sau localizarea informației. Operațiile de prelucrare includ operații aritmetice, operații logice sau mutarea datelor dintr-o zonă a memoriei în alta. Forma acestui simbol este un dreptunghi cu raportul de formă 2/3. Operația care este prezentă într-o astfel de figură geometrică este cea de atribuire (indicată prin simbolul  $\leftarrow$ ).

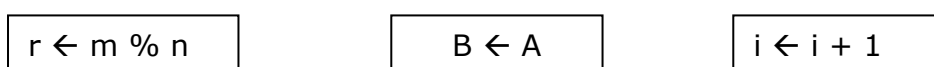


Figura nr. 2. Simbolul proces (sau operație)

c. *Simboluri de intrare – ieșire* reprezintă operațiile generale de intrare – ieșire, adică de introducere a informației în calculator pentru prelucrare (intrare) și de extragere a informației prelucrate (ieșire). Arată punctele de intrare a datelor și de ieșire a rezultatelor. Mediul suport de intrare sau de ieșire nu este specificat.

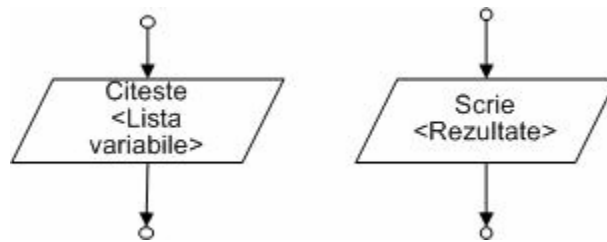
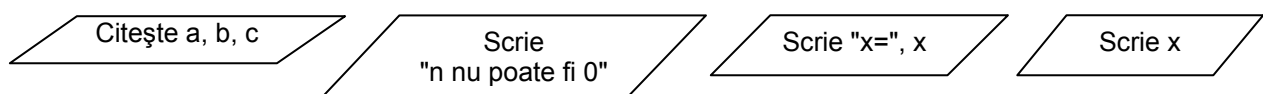


Figura nr. 3. Simboluri de intrare-ieșire

Exemple:



d. *Simboluri terminal* (Figura nr. 4) care marchează începutul (a) și sfârșitul (b) programului. Orice schemă logică trebuie să înceapă cu un simbol terminal START și să se termine cu un simbol terminal STOP.

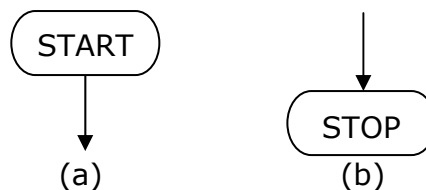


Figura 4. Simboluri pentru început și sfârșit program

e. *Simbolul conector* se folosește pentru a reprezenta o linie continuă când folosirea liniei continue este limitată de dimensiunea hârtiei sau din considerente de estetică ale schemei logice. Pentru a identifica conectorii de intrare și de ieșire se folosesc simboluri de identificare – litere sau cifre. Astfel conectorii reprezintă un mijloc elegant de a lega două puncte ale schemei logice fără a folosi liniile. O altă funcție a conectorilor este aceea că permit reprezentarea unei scheme logice pe mai multe pagini.

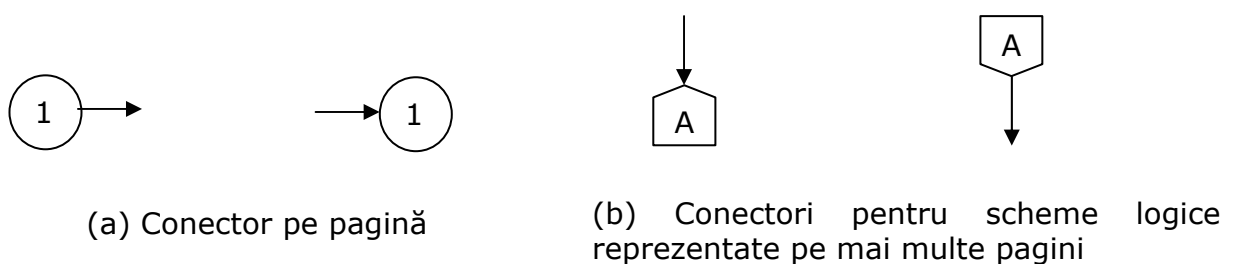


Figura nr. 5 Conectori

## 2.2. Structuri de program

### 2.2.1. Secvența

Este o structură formată din unul sau mai mulți pași (operații) care se execută secvențial (în ordinea în care apar în program).

Reprezentare prin schemă logică a secvenței este de forma dată în Figura nr. 6

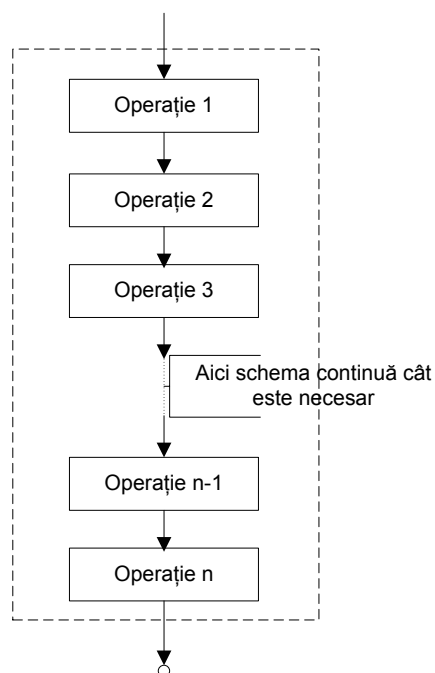


Figura nr. 6. O secvență cu **n** operații de prelucrare a informației inițiale

#### Exemplul nr. 1.

Să se facă schema logică pentru un program care citește două valori, calculează media lor aritmetică și afișează rezultatul.

Înainte de a desena schema logică se face o analiză a problemei enunțate stabilindu-se datele de intrare în program (care vor fi citite de la tastatură) și rezultatele (datele de ieșire) care trebuie calculate.

În acest caz ca date de intrare avem cele două valori pe care trebuie să le citim de la tastatură (și pe care le vom nota cu **a**, respectiv **b**). Data de ieșire (rezultatul) este în acest caz media aritmetică (pe care o vom nota **mediaAritmetica**).

Tot în acest punct stabilim cum vom calcula rezultatul (elaborăm **algoritmul de calcul**). Formula pentru calculul mediei aritmetice a două numere este:

$$mediaAritmetica = \frac{a + b}{2} \quad (1)$$

În acest caz algoritmul folosit este descris printr-o formulă matematică.

Schema logică este reprezentată în Figura nr. 7.

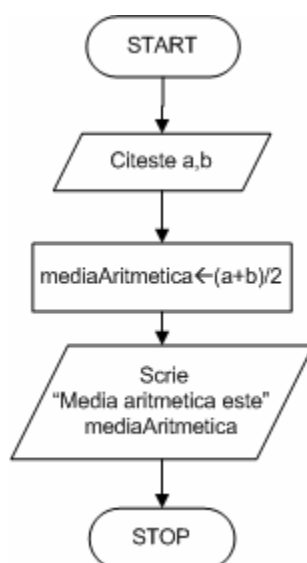


Figura nr. 7. Schema logică pentru problema din [Exemplul nr. 1](#)

După cum se observă, în schema logică nu există nici o ramificație, operațiile indicate executându-se exact în ordinea în care au fost specificate. Un astfel de program are o structură secvențială.

Tot structură secvențială va avea și programul din [Exemplul nr. 2](#).

### Exemplul nr. 2

Să se facă schema logică pentru un program care citește două valori, calculează media lor aritmetică și media lor geometrică și afișează rezultatele.

Procedând ca în [Exemplul nr. 1](#) se face mai întâi o analiză a problemei stabilindu-se datele de intrare, datele de ieșire și algoritmul (sau algoritmii) folosiți pentru rezolvarea problemei.

Datele de intrare sunt cele două numere care vor fi citite de la tastatură (notate și de această dată cu **a** și **b**). Ca date de ieșire avem în acest caz două rezultate: media aritmetică (pe care o vom numi **mediaAritmetica**) și media geometrică (pe care o vom numi **mediaGeometrica**).

Formulele de calcul (respectiv algoritmii folosiți pentru rezolvarea problemei) sunt:

$$\begin{aligned} \text{mediaAritmetica} &= \frac{a + b}{2} \\ \text{mediaGeometrica} &= \sqrt{a \cdot b} \end{aligned} \tag{2}$$

Schema logică este reprezentată în Figura nr. 8.

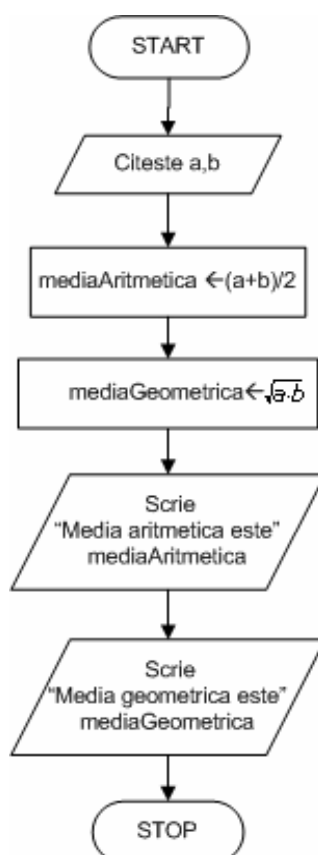


Figura nr. 8. Schema logică pentru problema din **Exemplul nr. 2**.

*Notă: schemele logice care trebuie să apară în documentarea programelor pot fi desenate cu diverse programe specializate. Microsoft pune la dispoziție pentru desenarea diagramelor necesare în diverse proiecte programul **Microsoft Office Visio** (cu varianta 2007). Acest program are o secțiune dedicată construirii schemelor logice (flowcharts) și este disponibil prin MSDN Academic Alliance.*

Cu toate că structura secvențială permite rezolvarea unor clase de probleme, există cazuri în care este necesară o ramificare a programului sau reluarea unor anumitor operații. De exemplu, în cazul în care citim două numere de la tastatură și trebuie să împărțim primul număr la cel de al doilea (primul fiind deîmpărțit, iar al doilea împărțitor) putem face acest lucru numai dacă ce de-al doilea număr este diferit de zero. În acest caz trebui să facem un test (testăm dacă al doilea număr este diferit de zero) și în funcție de rezultatul acestui test facem calculul cerut sau nu. Avem de luat în acest caz o **decizie** sau avem de făcut o **selecție**. Pentru astfel de probleme folosim în programare (regăsindu-se și în schemele logice) structura numită **decizie** sau **selecție**. Această structură este prezentată în paragraful următor.

### 2.2.2. Selecția (decizia)

Este o structură care permite ramificarea programului și continuarea prelucrării datelor pe una sau alta din ramuri în funcție de rezultatul unui test logic. Testul logic este o expresie a cărei evaluare conduce la un rezultat logic (din punctul de vedere al logicii matematice), adică rezultatul poate fi adevărat sau fals.

#### 2.2.2.1. Selecția cu două alternative

Reprezentare prin schemă logică este în Figura 9.

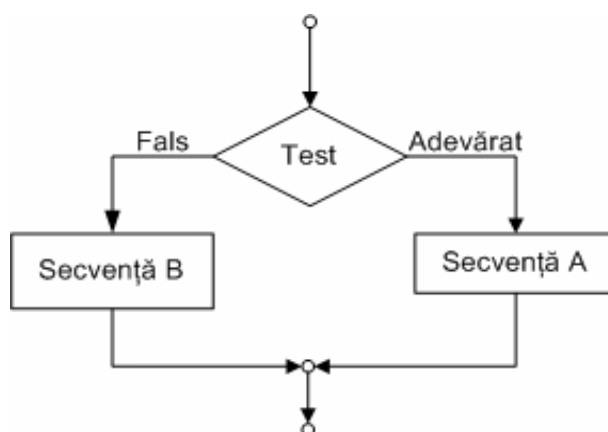


Figura 9. Selecția cu două alternative

Execuția se face astfel:

- se evaluează **Test** ;
- dacă rezultatul este **Adevărat** atunci se executa <Secvență A>;
- dacă rezultatul este **Fals** atunci se execută <Secvență B>.

După execuția uneia din cele două secvențe (A sau B), prelucrarea informațiilor continuă cu următoarea operație indicată în schema logică.

#### Exemplul nr. 3.

Să se deseneze schema logică pentru un program care citește două numere de la tastatură. Dacă primul număr este mai mare decât cel de-al doilea, atunci se va calcula suma celor două numere. În caz contrar se va calcula produsul celor două numere. În ambele cazuri se va afișa rezultatul calculului realizat.

Analiza problemei: datele de intrare sunt două numere (pe care le vom nota de data aceasta cu **x** și **y**). La ieșire vom avea un singur rezultat care depinde de valorile lui **x** și **y**. Pentru a ști ce operație trebuie să facem cu cele două numere trebuie să facem mai întâi un test și în funcție de rezultatul testului prelucrarea va continua pe o

ramură sau alta a programului. Pentru aceasta în program trebuie să existe atât operația de adunare a celor două numere cât și cea de înmulțire.

Rezultatul operației îl vom nota cu **rez**.

Schema logică este reprezentată în Figura nr. 10.

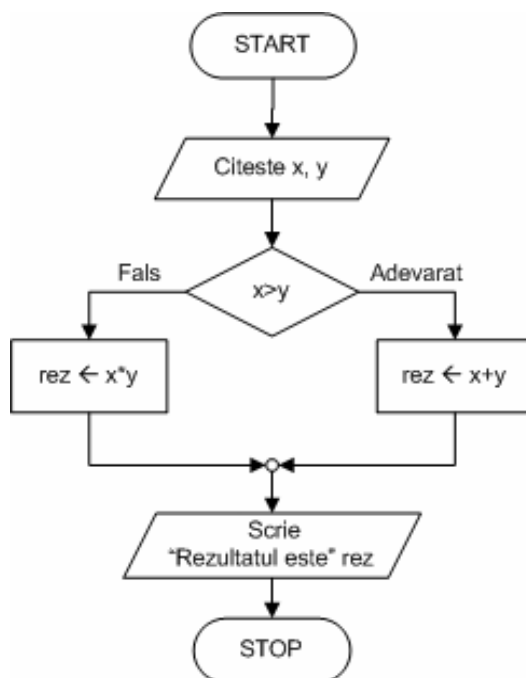


Figura nr. 10. Schema logică pentru Exemplul nr. 3

#### Exemplul nr. 4.

Fie funcția  $f: \mathbb{R} \rightarrow \mathbb{R}$  definită astfel:

$$f(x) = \begin{cases} x^2 + 1 & x \leq 1 \\ \ln(x^3 + 2) & x > 1 \end{cases} \quad (3)$$

Să se deseneze schema logică a unui program care citește de la tastatură valoarea lui **x** și afișează rezultatul expresiei:

$$f(x) + x + 1 \quad (4)$$

Analiza problemei: singura dată de intrare (informația care trebuie citită de la tastatură) este valoarea lui **x**. Data de ieșire este rezultatul expresiei din relația (4). Vom nota acest rezultat cu **r**.

Funcția **f** fiind dată pe intervale (expresia (3)) trebuie să testăm valoarea pe care am citit-o de la tastatură și dacă, această valoare este mai mică sau egală cu 1 vom folosi prima relație pentru calculul funcției **f**, altfel vom folosi a doua relație din definiția funcției. Valoarea calculată a funcției **f** am notat-o cu **y**.

În ambele cazuri rezultatul **r** se calculează folosind relația (4).

La final vom afișa rezultatul obținut.

Schema logică este reprezentată în Figura nr. 11.

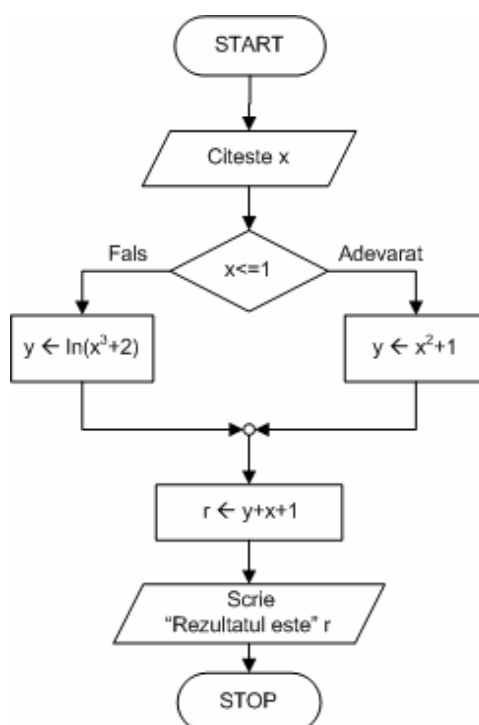


Figura nr. 11. Schema logică pentru Exemplul nr. 4

#### 2.2.2.2. Selecția cu o alternativă vidă

În cazul în care la evaluarea unui test nu avem două alternative de prelucrare, ci numai una singură, se folosește o structură de selecție derivată din cea prezentată în paragraful precedent numită selecția cu o alternativă vidă. Testul va fi construit astfel încât alternativa vidă (ramura pe care nu se execută nici o operație) să corespundă cazului în care rezultatul testului este Fals.

Reprezentare prin schema logică a acestui tip de selecție este dată în Figura nr. 12.

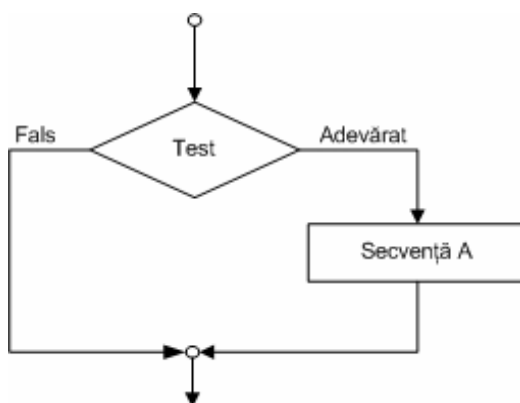


Figura nr. 12. Selecția cu o alternativă.

Este echivalentă cu structura anterioară numai că lipsește <secvența B>. Modul de execuție:

- se evaluează <**Test**>;
- dacă rezultatul este **Adevărat** atunci se executa <Secvență A>;
- dacă rezultatul este **Fals** nu se realizează nici o operație
- se trece apoi la următoarea operație de prelucrare din program.

#### Exemplul nr. 5.

Să se deseneze schema logică pentru un program care citește de la tastatură un număr și afișează modulul numărului respectiv.

Analiza problemei: programul are ca singură dată de intrare valoarea citită de la tastatură. Ca ieșire programul are de asemenea o singură informație și anume modulul numărului citit.

Algoritmul: dacă numărul citit este pozitiv, atunci modulul coincide cu numărul (deci în acest caz nu avem de făcut **nimic**). Dacă numărul citit este negativ atunci, pentru modulul este numărul cu semn schimbat. Modulul îl vom nota cu **modul**.

În ambele cazuri, vom afișa rezultatul.

Schema logică este reprezentată în Figura nr. 13.

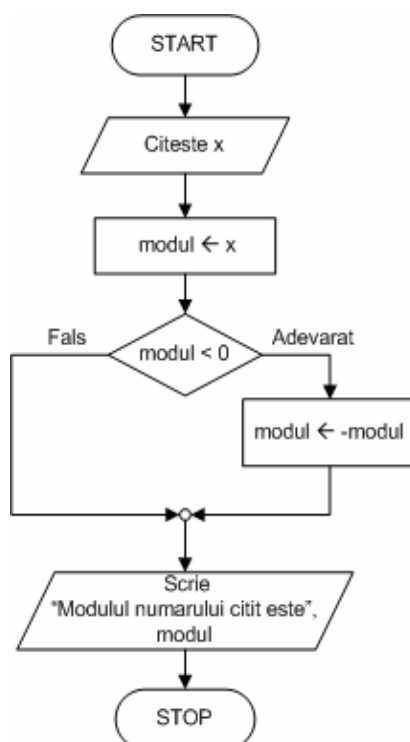


Figura nr. 13. Schema logică pentru Exemplul nr. 5.

### 2.2.2.3. Selecția multiplă

În cazul în care pe ramurile unei selecții avem nu operații simple, ci avem alte selecții avem de a face cu selecția multiplă.

#### Temă

Desenați schema logică pentru cazul în care pe fiecare din ramurile unei selecții cu două alternative avem câte o selecție cu o alternativă,

Un caz particular al selecției multiple este acela în care trebuie să evaluăm o expresie și în funcție de rezultatul expresiei trebuie să se execute o anumită secvență de instrucțiuni. Expresia care trebuie evaluată nu trebuie să fie o expresie logică sau de relație astfel încât rezultatul ei poate fi un număr oarecare (de obicei întreg).

Reprezentarea prin schemă logică

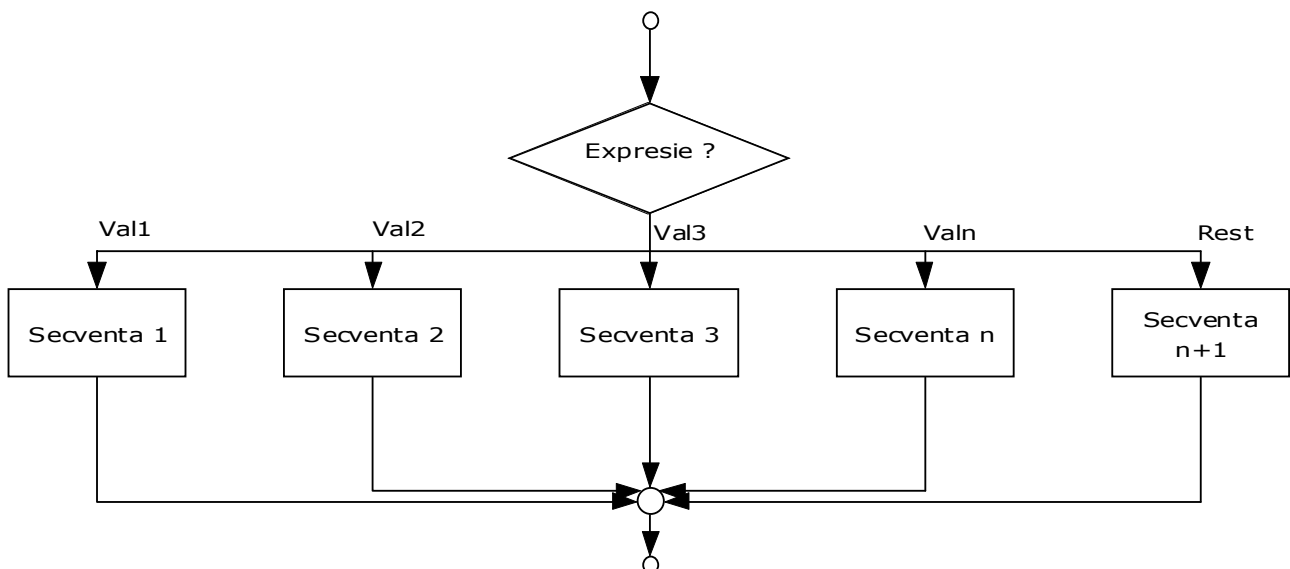


Figura nr. 14. Selecție multiplă (cazul II)

### 2.2.3. Repetiția (iterația)

Există situații în care pentru a rezolva problema dată trebuie să parcurgem aceeași pași de mai multe ori. De exemplu în cazul în care trebuie să calculăm suma unui șir de numere procedăm în felul următor: adunăm primele două numere, la suma obținută adunăm cel de-al treilea număr, la noua sumă adunăm cel de-al patrulea număr s.a.m.d.

Un alt exemplu: trebuie să citim de la tastatură un șir de numere. Citirea se face atât timp cât nu am citit un număr negativ. La primul număr negativ citit, operația de citire se termină și trecem mai departe să prelucrăm datele citite.

Pentru astfel de situații avem la dispoziție structuri repetitive sau iterative. Pașii care se repetă constituie partea iterată (repetată).

Întotdeauna, pe lângă partea iterată trebuie să existe și un test (ca în structurile de selecție) funcție de rezultatul căruia se reia execuția părții iterate sau nu.

Partea iterată împreună cu testul (care și el se repetă) formează bucla structurii repetitive.

**Repetiția (iterația)** este o structură compusă care conține o parte iterată care se execută de zero sau mai multe ori în funcție de rezultatul unui test logic. Partea iterată poate fi o instrucțiune simplă, o secvență, o selecție sau o altă iterație.

Acest tip de structură compusă poate fi întâlnită în una din formele

#### 2.2.3.1. Repetiție (iterație) cu test inițial

Schema logică este dată în Figura nr. 15:

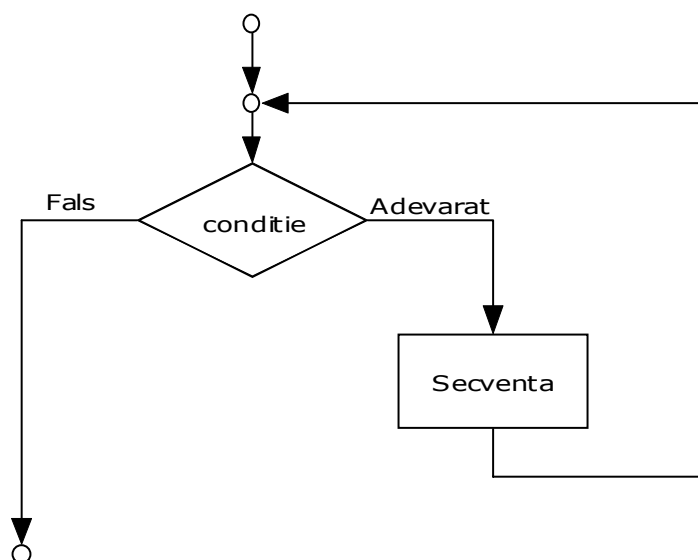


Figura nr. 15. Iterația cu test inițial

Modul de execuție al unei iterații este următorul:

- se evaluează <condiție>;
- dacă rezultatul este **Adevărat** se execută <secvența>;
- se calculează din nou valoarea <condiție>;
- dacă rezultatul este **Adevărat** se execută din nou <secvența>;
- secvența se execută în acest fel până când <condiție> capătă valoarea **fals**; atunci <secvență> nu se mai execută și iterația ia sfârșit.

Pentru ca iterația să se termine într-un număr finit de pași (și așa trebuie pentru că un program are un număr finit de pași, întotdeauna) este necesar ca partea iterată (<secvență>) să modifice valoarea unei variabile sau a mai multor variabile ce apar și în expresia logică reprezentând condiția de verificat, astfel încât după un număr de pași această expresie să capete valoare **fals**. Condiția se verifică întotdeauna înainte de execuția părții iterate, de aceea se numește iterație cu test inițial. Se poate întâmpla ca, în funcție de rezultatul evaluării condiției, partea iterată să nu se execute niciodată (se execută de zero ori).

#### 2.2.3.2. Repetiție (iterație) cu test final

Schema logică este dată în Figura nr. 16.

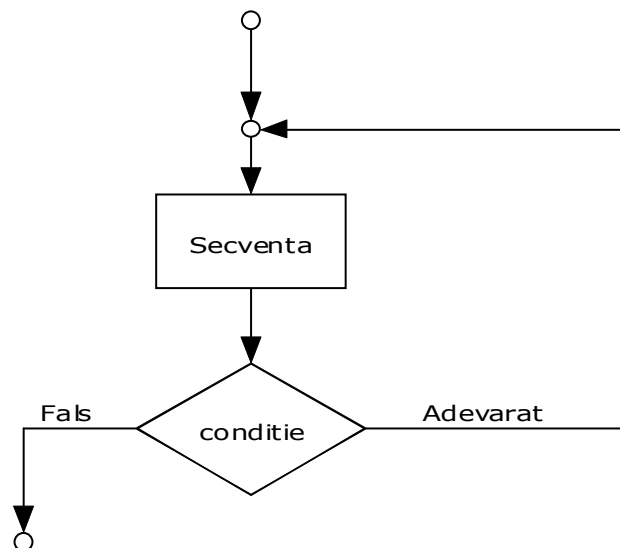


Figura nr. 16. Iterația cu test final

Modul de execuție este următorul:

- se execută partea iterată <secvență>;
- se evaluează <condiție>;
- dacă rezultatul este **adevarat** se execută din nou partea iterată <secvență>;
- se evaluează <condiție>;
- procedeul se repetă până când condiția devine **fals**; atunci partea iterată nu se mai execută.

Trebuie să observăm că iterația cu test inițial (1) este complet diferită de iterația cu test final (2). În cel de-al doilea caz, execuția părții iterate nu este precedată de evaluarea condiției ca în primul caz. Datorită acestui fapt în cazul

iterației cu test final, partea iterată se execută întotdeauna de cel puțin o dată, în timp ce în iterația cu test inițial partea iterată se poate executa de zero ori.

### 2.2.3.3. Repetiție (iterație) cu contor

Reprezentarea prin schema logică este dată în Figura nr. 17, forma generală fiind dată în Figura nr. 18.

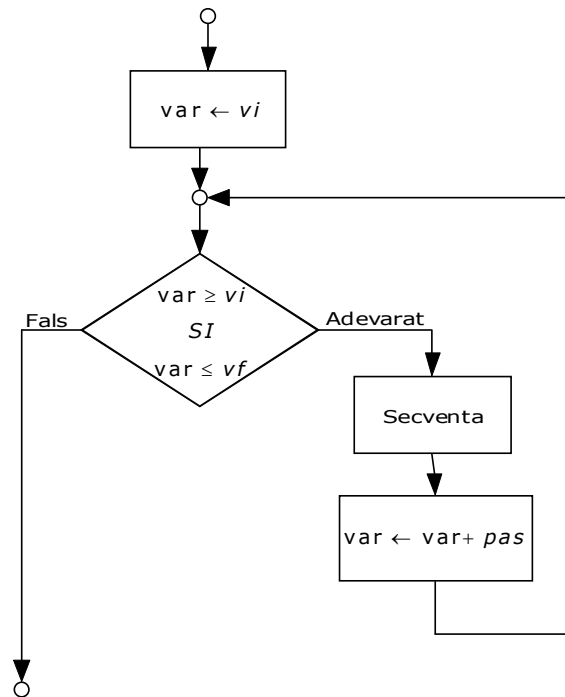


Figura nr. 17. Iterația cu contor

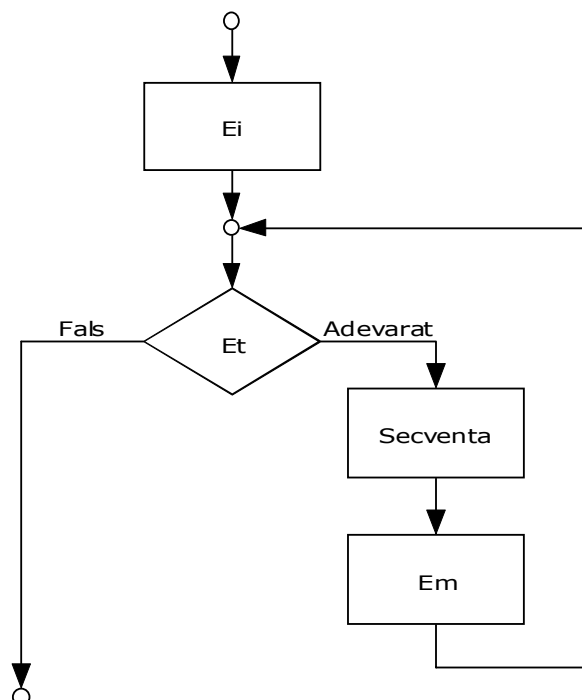


Figura nr. 18. Iterația cu contor (forma generală)

În acest caz iterația este controlată de variabila întreagă <var>, numită variabilă de contor a buclei. Valorile sale succesive constituie lista buclei. Acest tip de structură de control se compune din:

- o expresie de inițializare a variabilei de buclă **Ei**: <var> ← v<sub>i</sub>;
- testarea unei condiții pe care trebuie să o îndeplinească variabila de buclă prin expresia de test **Et**, de exemplu: valoarea variabilei întregi <var> trebuie să ∈ [v<sub>i</sub>, v<sub>f</sub>];
- o expresie de modificare (**Em**) a variabilei de buclă, care permite acestei variabile să ia toate valorile din lista buclei; de obicei este o instrucțiune de atribuire, de exemplu în acest caz <var> ← <var>+pas. pas reprezintă valoarea cu care se modifică variabila de buclă și poate lua atât valori pozitive, cât și valori negative, întregi sau reale.

Testarea unei condiții prin evaluarea expresiei **Et** se face înainte de execuția secvenței care reprezintă corpul iterației (a părții iterate) astfel încât se poate construi o echivalență între acest tip de iterație și iterația cu test inițial.

Acest tip de iterație se folosește în cazurile în care cunoaștem numărul de pași care trebuie executați pentru rezolvarea problemei, avem la dispoziție o relație de recurență sau putem evidenția cele trei tipuri de expresii (Ei, Et, Em) referitoare la o anumită variabilă folosită în descrierea algoritmului.

### 3. TEMA

Să se facă analiza și apoi să se deseneze schema logică pentru rezolvarea problemelor date mai jos. Pentru fiecare problemă se vor specifica (pe foaia de hârtie) datele de intrare, datele de ieșire și descrierea algoritmului (prin formule sau prin limbaj natural).

#### 3.1. Selecția

##### Problema nr. A1

Se citesc de pe dispozitivul de intrare 3 numere care reprezintă lungimile laturilor unui triunghi. Să se calculeze aria triunghiului folosind formula lui Heron.

$$A = \sqrt{p(p-a)(p-b)(p-c)}$$

unde  $p$  este semiperimetrul triunghiului.

**Problema nr. A2**

Se citesc două numere naturale de pe dispozitivul de intrare. Să se afișeze câtul împărțirii numărului mai mare la cel mai mic.

**Problema nr. A3**

Se citesc de pe dispozitivul de intrare 3 numere reale care reprezintă coeficienții unei ecuații de gradul 2. Să se determine rădăcinile acestei ecuații. În cazul în care ecuația nu poate fi rezolvată se vor afișa mesaje corespunzătoare.

**Problema nr. A4**

Un punct din plan este dat prin coordonatele sale  $(x, y)$ . Să se stabilească poziția punctului prin indicarea cadranelor (1, 2, 3, 4) în care este plasat. Pentru un punct situat pe una din semiaxe se vor preciza cadranele separate de semiaxa respectivă (de exemplu 2-3).

**Problema nr. A5**

Să se calculeze durata unei conexiuni Internet cunoscându-se momentul conectării (dat prin oră, minut și secundă) și momentul deconectării (de asemenea dat prin oră, minut și secundă). Trebuie să se ia în considerație cazul în care o conexiune începe într-o zi și se încheie în ziua următoare. **Atenție: rezultatul trebuie afișat sub forma ore, minute, secunde**, deci pentru calcul NU se va transforma totul în secunde.

**Problema nr. A6**

Să se calculeze momentul întreruperii unei conexiuni Internet dacă se cunoaște momentul de conectare (dat prin oră, minut și secundă) și durata totală a conexiunii (dată prin ore, minute și secunde). Se ia în considerație și cazul în care o conexiune începută într-o anumită zi se poate încheia în ziua următoare. **Atenție: rezultatul trebuie afișat sub forma ore, minute, secunde**, deci pentru calcul NU se va transforma totul în secunde.

**Problema nr. A7**

Se citesc trei numere naturale de pe dispozitivul de intrare (tastatura). Să se facă schema logică prin care se verifică dacă cele trei numere pot reprezenta laturile unui triunghi. În caz afirmativ se va afișa pe dispozitivul de ieșire (monitorul) un

mesaj care indică tipul triunghiului (echilateral, isoscel, dreptunghic) ale cărui laturi pot avea valorile citite de la tastatură. (Nu se pune problema ca triunghiul ar putea fi dreptunghic isoscel – de ce?).

### Problema nr. A8

Fie următoarea funcție:

$$f(x) = \begin{cases} x^2 + 3x + 5, & x \in (-\infty, 2] \\ 3x, & x \in (2, 8) \\ e^x + 2, & x \in [8, \infty) \end{cases}$$

Să se calculeze valoarea funcției  $f$  pentru un  $x$  citit de pe dispozitivul de intrare.

## 3.2. Iterația

### Problema nr. B1

a) Să se deseneze schema logică pentru un program care afișează un tabel care pe prima coloană are temperaturi Fahrenheit și pe cea de-a doua coloană echivalențele lor în grade Celsius, folosind formula

$$C = \frac{5}{9}(F - 32)$$

unde  $C$  reprezintă valoarea temperaturii dată în grade Celsius, iar  $F$  reprezintă valoarea temperaturii în grade Fahrenheit. Tabelul este generat pentru temperaturi Fahrenheit cuprinse între 0 și 300, cu un pas egal cu  $10^\circ\text{F}$ .

b) Să se deseneze schema logică pentru tabelul corespunzător de transformare din grade Celsius în grade Fahrenheit. De această dată pasul va fi citit de pe dispozitivul de intrare.

### Problema nr. B2

De pe dispozitivul de intrare se citește un număr real  $b$  și un șir de valori reale pozitive terminate printr-o valoare negativă (care nu face parte din șir). Să se stabilească elementul din șir cel mai apropiat de  $b$ . Se va preciza și poziția acestuia (al câtelea element din șir este). Termenii șirului vor fi păstrați în aceeași variabilă  $a$ .

**Problema nr. B3**

De pe dispozitivul de intrare se citește un șir de numere întregi pozitive, șir terminat cu valoarea 0 care nu face parte din șir. Pentru fiecare element citit să se indice cel mai mare pătrat perfect mai mic sau egal cu el.

**Problema nr. B4**

Șirul  $\{x_n\}$  generat cu relația de recurență  $x_n = \frac{1}{p} \left[ (p-1)x_{n-1} + \frac{a}{(x_{n-1})^{p-1}} \right]$ , cu  $a \in R$  și  $p \in N$ , și  $x_0 = \frac{a}{p}$  este convergent și are ca limită  $\sqrt[p]{a}$  (în cazul în care  $p$  este par trebuie ca  $a$  să fie un număr pozitiv). Pentru un  $a$  oarecare dat să construiască schema logică a unui program care calculează  $\sqrt[p]{a}$  ca limită a acestui șir cu precizia  $\varepsilon$  de asemenea dată.

Precizia este dată de relația  $\varepsilon = |x_n - x_{n-1}|$ .

Să se completeze schema logică astfel încât să se poată calcula rezultatul pentru mai multe valori ale lui  $p$ . Continuarea programului se va stabili în urma unui dialog cu utilizatorul (după exemplul dat la curs).

**Problema nr. B5**

Să se determine valoarea  $n$  pentru care  $S_n = \sum_{k=1}^n \frac{2}{\sqrt{4n^2 - k}}$  satisface condiția  $\left| S_n - \frac{\pi}{3} \right| < \varepsilon$ , în care  $\varepsilon$  este dat. Se știe că  $\lim_{n \rightarrow \infty} S_n = \frac{\pi}{3}$ .

**Problema nr. B6**

Să se calculeze funcția  $J_n(x)$  știind că există relația de recurență:

$$J_n(x) = (2n-2)/x \cdot J_{n-1}(x) - J_{n-2}(x)$$

$$\text{cu } J_0(x) = \sum_{k=0}^{\infty} (-1)^k \frac{\left(\frac{x}{2}\right)^{2k}}{(k!)^2} \text{ și } J_1(x) = \sum_{k=0}^{\infty} (-1)^k \frac{\left(\frac{x}{2}\right)^{2k+1}}{k!(k+1)!}.$$

Calcululele se fac cu precizia  $\varepsilon$ . De pe dispozitivul de intrare se citesc  $x, n$  și  $\varepsilon$ .

**Problema nr. B7**

Fie  $x$  un număr real și  $n$  un număr întreg. Să se calculeze  $x^n$  folosind un număr cât mai mic de înmulțiri de numere reale.

*Indicație.* Se observă că  $x^n$  poate fi descompus într-un produs de factori în care pot apare  $x, x^2, x^4, x^8 \dots$ . Un factor apare în produs dacă în reprezentarea binară a lui  $n$  cifra corespunzătoare este 1.

Vom forma deci pe rând cifrele din reprezentarea binară a lui  $n$ , începând cu cea mai puțin semnificativă și puterile lui  $x$ :  $x, x^2, x^4, x^8 \dots$ . O putere a lui  $x$  apare în produs dacă cifra corespunzătoare din reprezentarea binară este 1. De exemplu:  $x^{13} = x^8 x^4 x$  pentru că reprezentarea în baza 2 a lui 13 este 1101.

Cifrele reprezentării binare a lui  $n$  se obțin ca resturi ale împărțirii succesive a lui  $n$  la 2.

**Problema nr. B8**

Se consideră ecuația  $ax^3 + bx + c = 0$ . Notând cu  $x_1, x_2, x_3$  rădăcinile ecuației, să se calculeze  $x_1^n + x_2^n + x_3^n$ , fără a rezolva ecuația. Se dau  $a, b, c$  și  $n$ .

*Indicație.* Înmulțim ecuația cu  $x^{n-3}$ :  $ax^n + bx^{n-2} + cx^{n-3} = 0$  și ținând cont de faptul că fiecare rădăcină satisface ecuația dată, rezultă:

$$ax_1^n + bx_1^{n-2} + cx_1^{n-3} = 0$$

$$ax_2^n + bx_2^{n-2} + cx_2^{n-3} = 0$$

$$ax_3^n + bx_3^{n-2} + cx_3^{n-3} = 0$$

Adunăm cele trei relații și obținem:

$$a(x_1^n + x_2^n + x_3^n) + b(x_1^{n-2} + x_2^{n-2} + x_3^{n-2}) + c(x_1^{n-3} + x_2^{n-3} + x_3^{n-3}) = 0$$

Notând  $s_n = x_1^n + x_2^n + x_3^n$ , obținem relația de recurență:

$$a \cdot s_n + b \cdot s_{n-2} + c \cdot s_{n-3} = 0$$

cu următoarele valori inițiale:

$$s_0 = x_1^0 + x_2^0 + x_3^0 = 3$$

$$s_1 = x_1 + x_2 + x_3 = 0 \text{ (relațiile Viete)}$$

$$s_2 = x_1^2 + x_2^2 + x_3^2 = -2b/a \text{ (relațiile Viete)}$$