LABORATOR NR. 1

Reprezentarea algoritmilor prin scheme logice.

1. Algoritm

Algoritmul este o „reţetă" pentru rezolvarea unei anumite probleme. Urmarea pas cu pas a indicaţiilor din „reţetă" permite transformarea informaţiilor iniţiale (cele pe care le avem la dispoziţie în momentul începerii rezolvării problemei – altfel spus a „ingredientelor") în rezultate („produse finite"). Informaţiile iniţiale le mai numim şi **date (informaţii) de intrare**. Rezultatele obţinute în urma urmării paşilor dintr-o „reţetă" le numim **date (informaţii) de ieşire.** Indicaţiile din „reţetă" sunt **paşii algoritmului**.

Un **program** poate fi format din unul sau mai mulţi algoritmi. Şi pentru program avem **date de intrare** (sunt informaţiile de care dispunem la începutul programului) şi **date de ieşire** (sunt informaţiile obţinute în urma rulării programului pe calculator).

Rezolvarea unei probleme presupune parcurgerea a două etape:

1. etapa de analiză a problemei (înţelegerea problemei, stabilirea datelor de intrare şi a datelor de ieşire, elaborarea şi descrierea algoritmului sau a algoritmilor necesari pentru rezolvarea problemei);

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

Informaţiile iniţiale (datele de intrare) sunt introduse în program prin **citire**. Dacă, în urma analizei problemei, rezultă că, pentru rezolvarea problemei, avem nevoie de anumite informaţii, 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 şiruri de caractere.

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

Informaţiile iniţiale 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 informaţiile iniţiale ale programului şi

apoi toate rezultatele intermediare au ataşate „etichete” cu numele dat informaţiei 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 informaţii şi care va avea această „etichetă”. Zonele de memorie care au ataşat (prin program) un **nume sugestiv** şi în care sunt stocate informaţiile 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 şi informaţii ale căror valori nu se pot modifica pe parcursul unui program. Aceste informaţii pot avea de asemenea asociat un nume. Şi în alte domenii (matematică, fizică, chimie) avem astfel de informaţii pe care le numim constante. De exemplu, în matematică avem constanta notată cu π şi care este egală cu 3.14159...

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

Pentru obţinerea rezultatelor finale (datele de ieşire), precum şi a unor rezultate intermediare, variabilele şi constantele care apar într-un algoritm pot fi incluse în expresii aritmetice, logica sau relaţionale.

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

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

Expresiile relaţionale sunt formate cu ajutorul operatorilor relaţionale: $<$, $\leq$, $>$, $\geq$, $=$ şi $\neq$. Expresiile relaţionale sunt, de fapt, propoziţii, deci au ca rezultat valoarea de adevăr a propoziţiilor respective.

O altă operaţie importantă într-un program este **atribuirea**. Simbolul prin care vom nota această operaţie este $\leftarrow$. Forma generală pentru această operaţie 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 operaţie? 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 execuţie 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 variabile ***v***. După execuţia acestei atribuiri, în zona de memorie asociată variabilei ***v*** se va afla o valoare mai mare cu o unitate faţă de valoarea iniţială a variabilei ***v***.

Iniţializarea unei variabile înseamnă prima „umplere” a zonei de memorie asociate variabilei considerate cu o valoare. Această iniţializare 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 săgeţi care arată ordinea de execuţie a paşilor algoritmului. Această ordine de execuţie a paşilor algoritmului (sau a programului) se numeşte structura de control a programului.

Reprezentarea grafică bidimensională a unui algoritm cu ajutorul unor simboluri speciale se numeşte 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 operaţii care trebuie executate şi ordinea de execuţie. Ea reprezintă în acelaşi timp şi o imagine a fluxului de informaţii 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 înţeleagă logica programului, să realizeze modularizarea programului (prin împărţirea programului în părţi mai mici şi mai detaliate), să verifice îndeplinirea tuturor condiţiilor posibile din program, să traducă algoritmul într-un limbaj de programare (de exemplu în limbajul **C**).

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

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

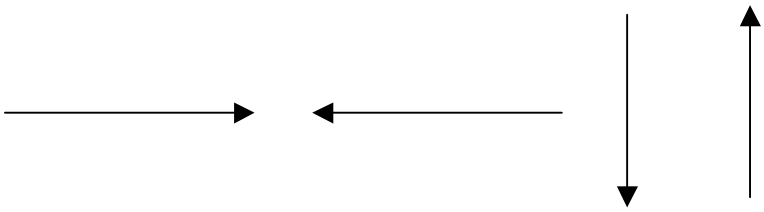


Figura nr. 1. Simboluri pentru direcţia fluxului de informaţie,

Acesta simbolizează direcţia fluxului de informaţie în calculator şi succesiunea diferitelor operaţii. Liniile de flux se pot încrucişa. Liniile care se încrucişează nu înseamnă că sunt conectate. Dacă ele sunt conectate trebuie să apară vârfuri de săgeţi lângă punctul de joncţiune pentru a marca conexiunea. Se reprezintă printr-un segment de dreaptă de orice lungime care trebuie să lege alte două simboluri. Direcţia normală a fluxului este de la stânga la dreapta şi de sus în jos. Când direcţia fluxului este de sens opus direcţiei normale se folosesc vârfuri de săgeată. Pentru mai multă claritate mulţi programatori folosesc vârfurile de săgeată în toate cazurile.

b. *Simbolul proces* (Figura nr. 2) reprezintă execuţia unei (sau mai multor) operaţii de prelucrare a uneia sau mai multor informaţii în urma căreia se schimbă valoarea, forma sau localizarea informaţiei. Operaţiile de prelucrare includ operaţii aritmetice, operaţii logice sau mutarea datelor dintr-o zonă a memoriei în alta. Forma acestui simbol este un dreptunghi cu raportul de formă 2/3. Operaţia care este prezentă într-o astfel de figură geometrică este cea de atribuire (indicată prin simbolul ←).

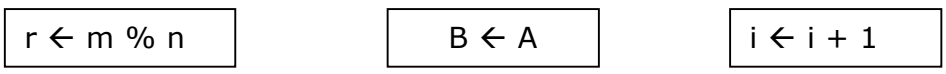


Figura nr. 2. Simbolul proces (sau operaţie)

c. *Simboluri de intrare – ieşire* reprezintă operaţiile generale de intrare – ieşire, adică de introducere a informaţiei în calculator pentru prelucrare (intrare) şi de extragere a informaţiei prelucrate (ieşire). Arată punctele de intrare a datelor şi de ieşire a rezultatelor. Mediul suport de intrare sau de ieşire nu este specificat.

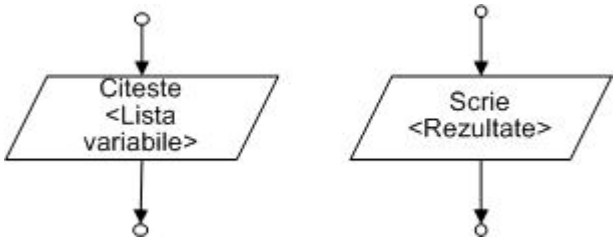
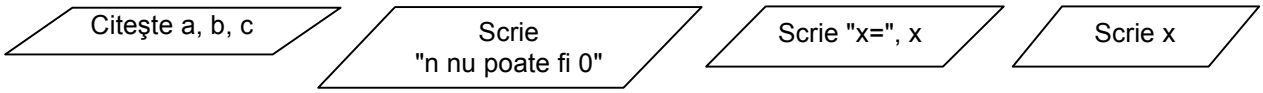


Figura nr. 3. Simboluri de intrare-ieşire

Exemple:



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

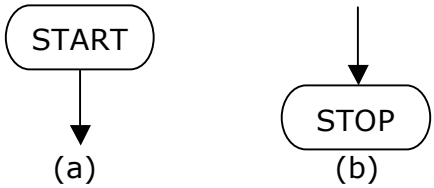


Figura 4. Simboluri pentru început şi sfârşit program

e. *Simbolul conector* se foloseşte 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 şi de ieşire 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 funcţie a conectorilor este aceea că permit reprezentarea unei scheme logice pe mai multe pagini.

(a) Conector pe pagină

(b) Conectori pentru scheme logice reprezentate pe mai multe pagini

Figura nr. 5 Conectori

2.2. Structuri de program

2.2.1. Secvenţa

Este o structură formată din unul sau mai mulţi paşi (operaţii) care se execută secvenţial (în ordinea în care apar în program).

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

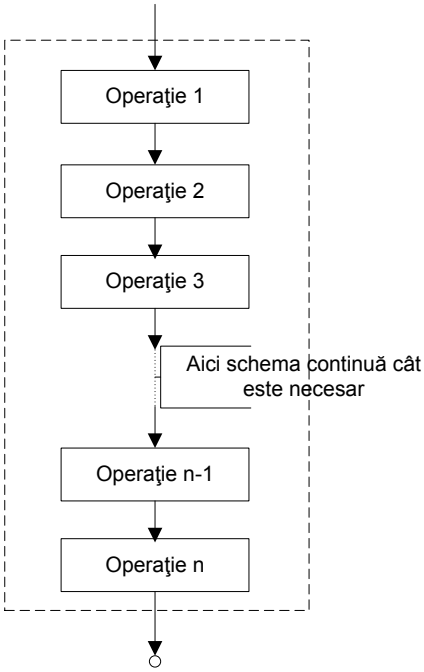


Figura nr. 6. O secvenţă cu **n** operaţii de prelucrare a informaţiei iniţiale

Exemplul nr. 1.

Să se facă schema logică pentru un program care citeşte două valori, calculează media lor aritmetică şi afişează rezultatul.

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

În acest caz ca date de intrare avem cele două valori pe care trebuie să le citim de la tastatură (şi pe care le vom nota cu **a**, respectiv **b**). Data de ieşire (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:

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

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

Schema logică este reprezentată în Figura nr. 7.

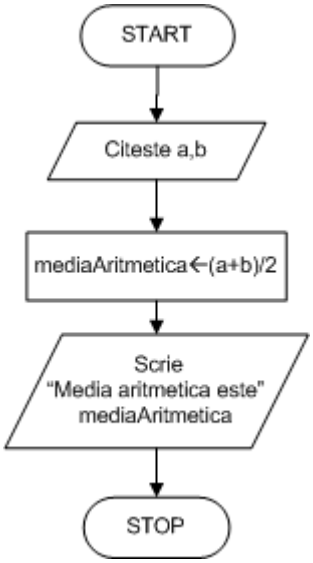


Figura nr. 7. Schema logică pentru problema din Exemplul nr. 1

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

Tot structură secvenţială va avea şi programul din Exemplul nr. 2.

Exemplul nr. 2

Să se facă schema logică pentru un program care citeşte două valori, calculează media lor aritmetică şi media lor geometrică şi afişează rezultatele.

Procedând ca în Exemplul nr. 1 se face mai întâi o analiză a problemei stabilindu-se datele de intrare, datele de ieşire şi algoritmul (sau algoritmii) folosiţi pentru rezolvarea problemei.

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

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

$$\begin{aligned} mediaAritmetica &= \frac{a+b}{2} \\ mediaGeometrica &= \sqrt{a \cdot b} \end{aligned} \qquad (2)$$

Schema logică este reprezentată în Figura nr. 8.

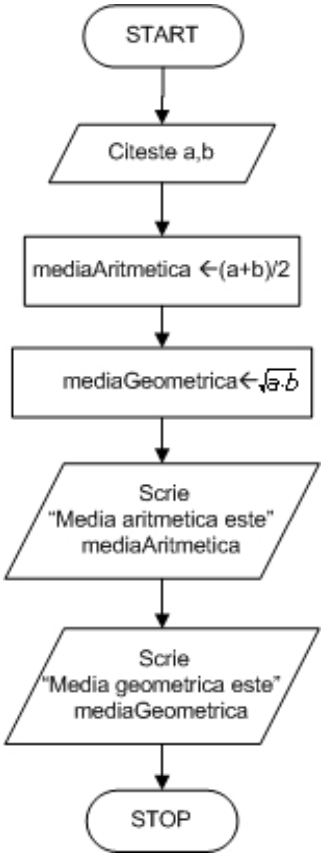


Figura nr. 8. Schema logică pentru problema din Exemplul nr. 2.

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

Cu toate că structura secvenţială permite rezolvarea unor clase de probleme, există cazuri în care este necesară o ramificare a programului sau reluarea unor anumitor operaţii. De exemplu, în cazul în care citim două numere de la tastatură şi trebuie să împărţim primul număr la cel de al doilea (primul fiind deîmpărţit, iar al doilea împărţitor) 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) şi în funcţie de rezultatul acestui test facem calculul cerut sau nu. Avem de luat în acest caz o **decizie** sau avem de făcut o **selecţie**. Pentru astfel de probleme folosim în programare (regăsindu-se şi în schemele logice) structura numită **decizie** sau **selecţie**. Această structură este prezentată în paragraful următor.

2.2.2. Selecţia (decizia)

Este o structură care permite ramificarea programului şi continuarea prelucrării datelor pe una sau alta din ramuri în funcţie 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. Selecţia cu două alternative

Reprezentare prin schemă logică este în Figura 9.

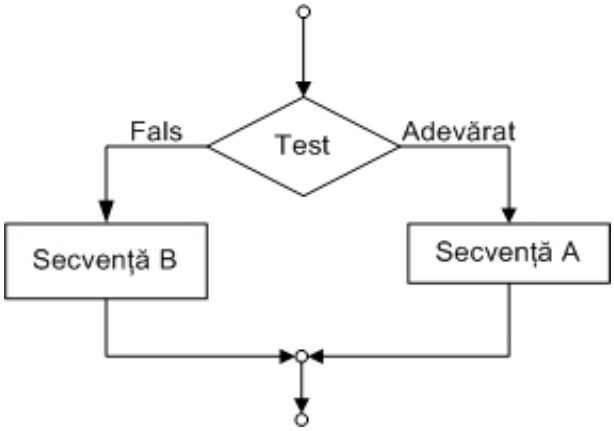


Figura 9. Selecţia cu două alternative

Execuţia se face astfel:

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

După execuţia uneia din cele două secvenţe (A sau B), prelucrarea informaţiilor continuă cu următoarea operaţie indicată în schema logică.

Exemplul nr. 3.

Să se deseneze schema logică pentru un program care citeşte 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 afişa rezultatul calculul realizat.

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

ramură sau alta a programului. Pentru aceasta în program trebuie să existe atât operaţia de adunare a celor două numere cât şi cea de înmulţire.

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

Schema logică este reprezentată în Figura nr. 10.

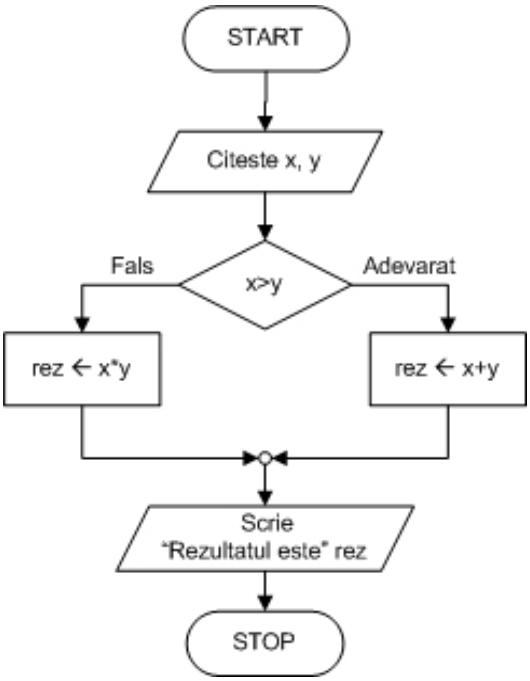


Figura nr. 10. Schema logică pentru Exemplul nr. 3

Exemplul nr. 4.

Fie funcţia f:R→R definită astfel:

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

Să se deseneze schema logică a unui program care citeşte de la tastatură valoarea lui **x** şi afişează rezultatul expresiei:

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

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

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

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

La final vom afişa rezultatul obţinut.

Schema logică este reprezentată în Figura nr. 11.

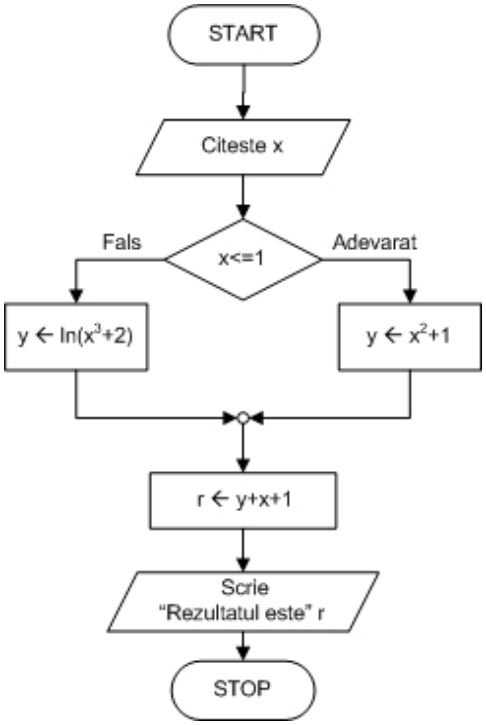


Figura nr. 11. Schema logică pentru Exemplul nr. 4

2.2.2.2. Selecţia cu o alternativă vidă

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

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

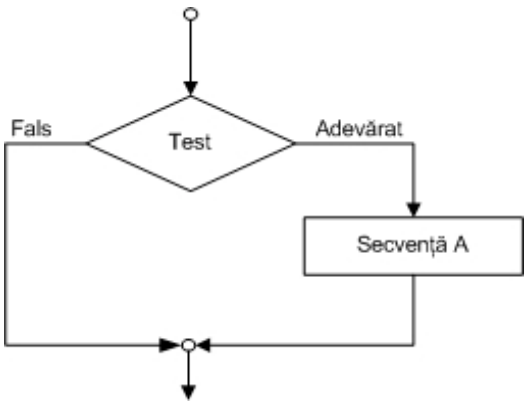


Figura nr. 12. Selecţia cu o alternativă.

Este echivalentă cu structura anterioară numai că lipseşte <secvenţa B>. Modul de execuţie:

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

Exemplul nr. 5.

Să se deseneze schema logică pentru un program care citeşte de la tastatură un număr şi afişează modulul numărului respectiv.

Analiza problemei: programul are ca singură dată de intrare valoarea citită de la tastatură. Ca ieşire programul are de asemenea o singură informaţie şi 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 afişa rezultatul.

Schema logică este reprezentată în Figura nr. 13.

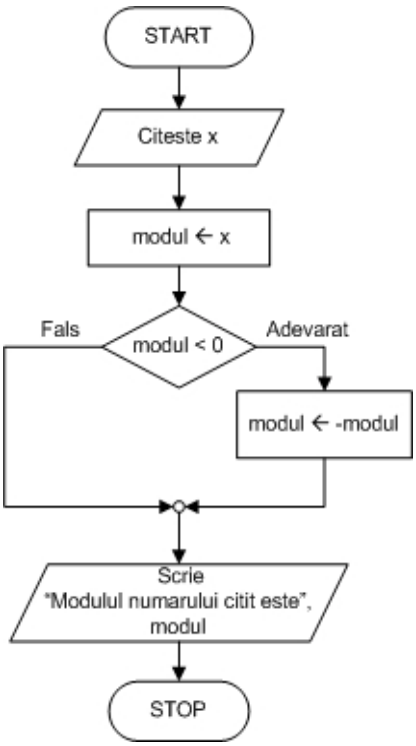


Figura nr. 13. Schema logică pentru Exemplul nr. 5.

2.2.2.3. Selecţia multiplă

În cazul în care pe ramurile unei selecţii avem nu operaţii simple, ci avem alte selecţii avem de a face cu selecţia multiplă.

Temă

Desenaţi schema logică pentru cazul în care pe fiecare din ramurile unei selecţii cu două alternative avem câte o selecţie cu o alternativă,

Un caz particular al selecţiei multiple este acela în care trebuie să evaluăm o expresie şi în funcţie de rezultatul expresiei trebuie să se execute o anumită secvenţă de instrucţiuni. Expresia care trebuie evaluată nu trebuie să fie o expresie logică sau de relaţie astfel încât rezultatul ei poate fi un număr oarecare (de obicei întreg).

Reprezentarea prin schemă logică

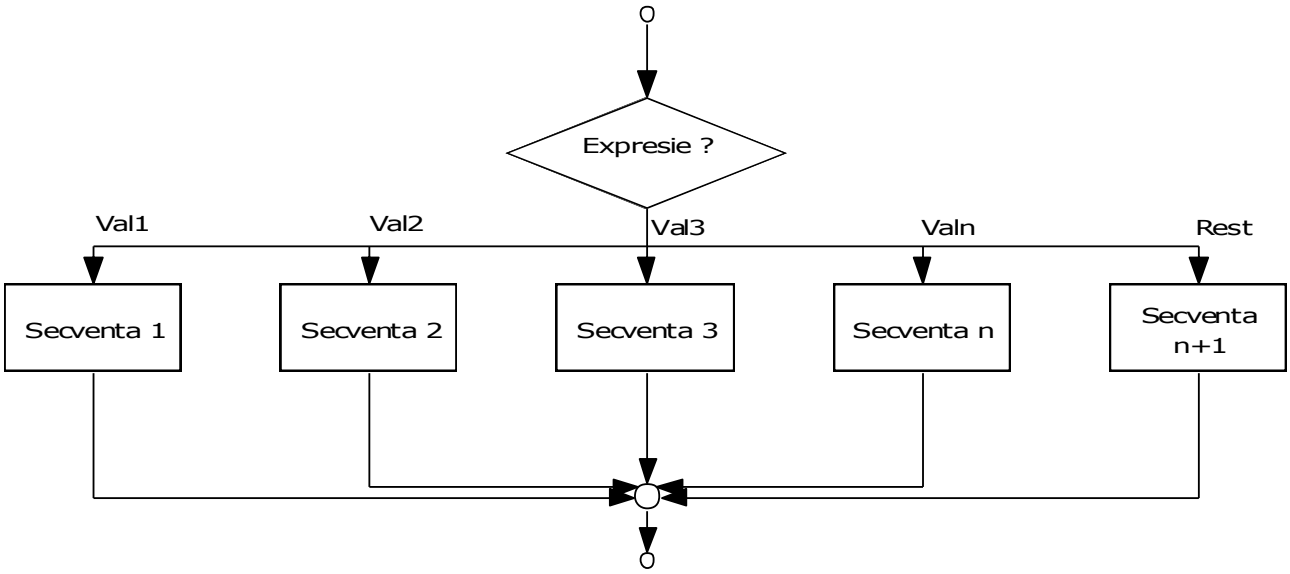


Figura nr. 14. Selecţie multiplă (cazul II)

2.2.3. Repetiţia (iteraţia)

Există situaţii în care pentru a rezolva problema dată trebuie să parcurgem aceeaşi paşi de mai multe ori. De exemplu în cazul în care trebuie să calculăm suma unui şir de numere procedăm în felul următor: adunăm primele două numere, la suma obţinută 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 şir de numere. Citirea se face atât timp cât nu am citit un număr negativ. La primul număr negativ citit, operaţia de citire se termină şi trecem mai departe să prelucrăm datele citite.

Pentru astfel de situaţii avem la dispoziţie structuri repetitive sau iterative. Paşii care se repetă constituie partea iterată (repetată).

Întotdeauna, pe lângă partea iterată trebuie să existe şi un test (ca în structurile de selecţie) funcţie de rezultatul căruia se reia execuţia părţii iterate sau nu.

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

Repetiţia (iteraţia) este o structură compusă care conţine o parte iterată care se execută de zero sau mai multe ori în funcţie de rezultatul unui test logic. Partea iterată poate fi o instrucţiune simplă, o secvenţă, o selecţie sau o altă iteraţie.

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

2.2.3.1. Repetiţie (iteraţie) cu test iniţial

Schema logică este dată în Figura nr. 15:

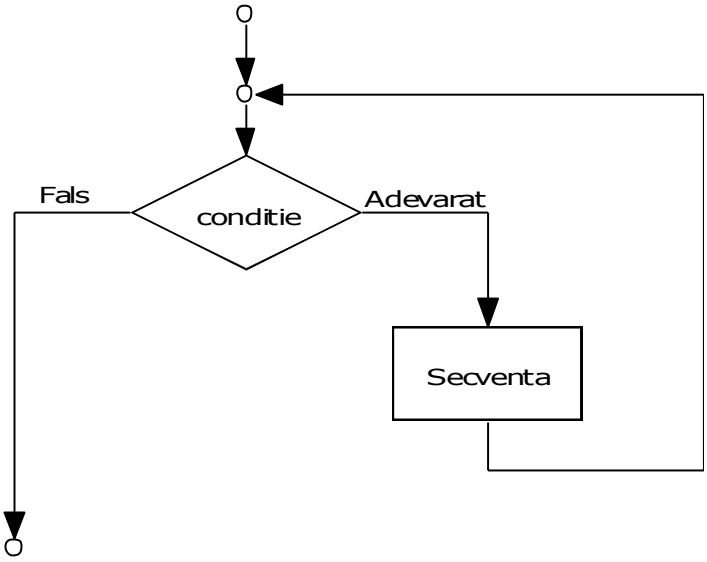


Figura nr. 15. Iteraţia cu test iniţial

Modul de execuţie al unei iteraţii este următorul:

– se evaluează <condiţie>;

– dacă rezultatul este **Adevărat** se execută <secvenţa>;

– se calculează din nou valoarea <condiţie>;

– dacă rezultatul este **Adevărat** se execută din nou <secvenţa>;

– secvenţa se execută în acest fel până când <condiţie> capătă valoarea **fals**; atunci <secvenţă> nu se mai execută şi iteraţia ia sfârşit.

Pentru ca iteraţia să se termine într-un număr finit de paşi (şi aşa trebuie pentru că un program are un număr finit de paşi, întotdeauna) este necesar ca partea iterată (<secvenţa>) să modifice valoarea unei variabile sau a mai multor variabile ce apar şi în expresia logică reprezentând condiţia de verificat, astfel încât după un număr de paşi această expresie să capete valoare **fals**. Condiţia se verifică întotdeauna înainte de execuţia părţii iterate, de aceea se numeşte iteraţie cu test iniţial. Se poate întâmpla ca, în funcţie de rezultatul evaluării condiţiei, partea iterată să nu se execute niciodată (se execută de zero ori).

2.2.3.2. Repetiţie (iteraţie) cu test final

Schema logică este dată în Figura nr. 16.

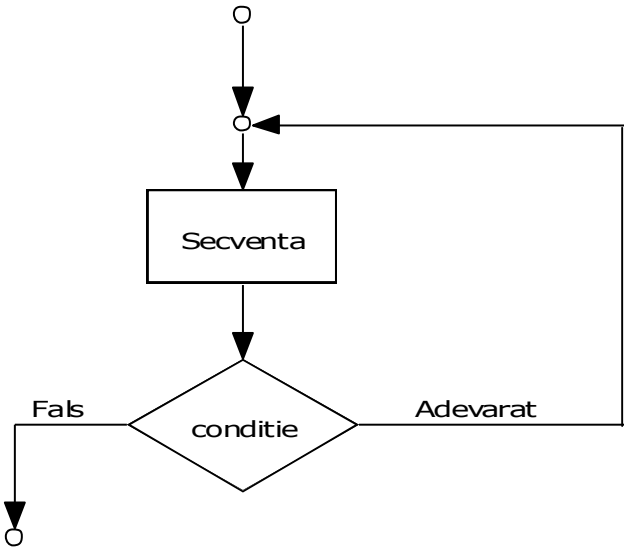


Figura nr. 16. Iteraţia cu test final

Modul de execuţie este următorul:

– se execută partea iterată <secvenţă>;

– se evaluează <condiţie>;

– dacă rezultatul este **adevarat** se execută din nou partea iterată <secvenţă>;

– se evaluează <condiţie>;

– procedeul se repetă până când condiţia devine **fals**; atunci partea iterată nu se mai execută.

Trebuie să observăm că iteraţia cu test iniţial (1) este complet diferită de iteraţia cu test final (2). În cel de-al doilea caz, execuţia părţii iterate nu este precedată de evaluarea condiţiei ca în primul caz. Datorită acestui fapt în cazul

iteraţiei cu test final, partea iterată se execută întotdeauna de cel puţin o dată, în timp ce în iteraţia cu test iniţial partea iterată se poate executa de zero ori.

2.2.3.3. Repetiţie (iteraţie) cu contor

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

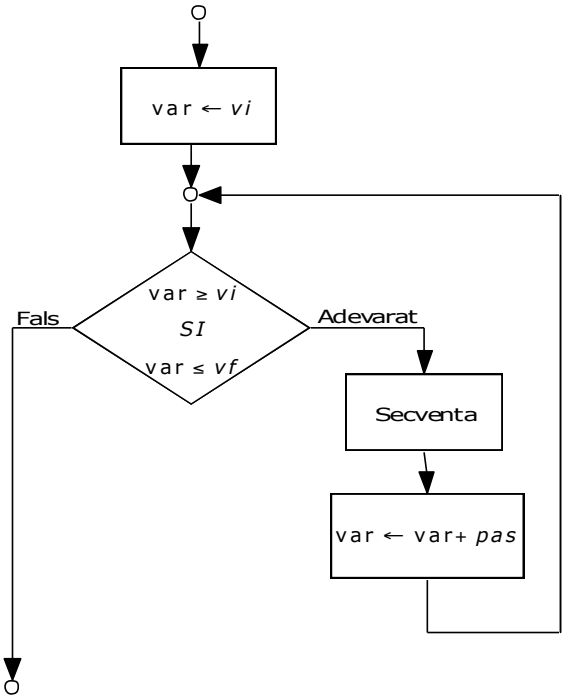


Figura nr. 17. Iteraţia cu contor

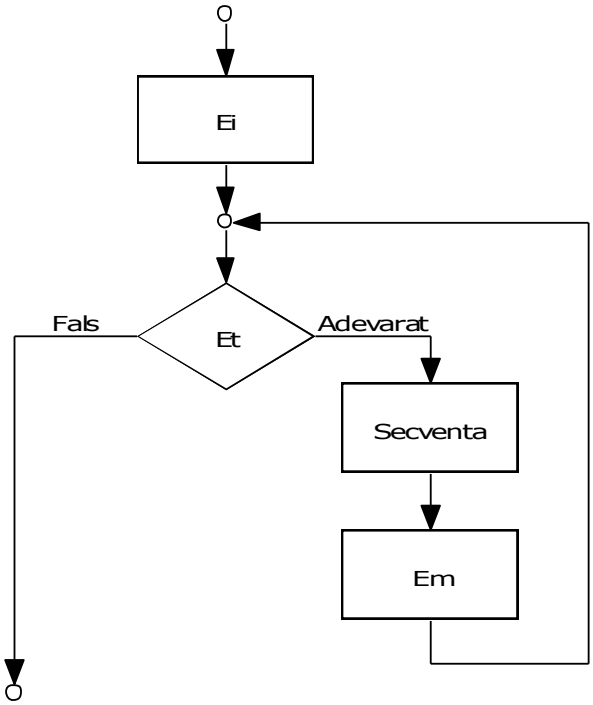


Figura nr. 18. Iteraţia cu contor (forma generală)

În acest caz iteraţia 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 iniţializare a variabilei de buclă **Ei**: <var> ← vi;

– testarea unei condiţii pe care trebuie să o îndeplinească variabila de buclă prin expresia de test **Et**, de exemplu: valoarea variabilei întregi <var> trebuie să ∈[vi, vf];

– o expresie de modificare (**Em**) a variabilei de buclă, care permite acestei variabile să ia toate valorile din lista buclei; de obicei este o instrucţiune de atribuire, de exemplu în acest caz <var> ← <var>+pas. pas reprezintă valoarea cu care se modifică variabila de buclă şi poate lua atât valori pozitive, cât şi valori negative, întregi sau reale.

Testarea unei condiţii prin evaluarea expresiei **Et** se face înainte de execuţia secvenţei care reprezintă corpul iteraţiei (a părţii iterate) astfel încât se poate construi o echivalenţă între acest tip de iteraţie şi iteraţia cu test iniţial.

Acest tip de iteraţie se foloseşte în cazurile în care cunoaştem numărul de paşi care trebuie executaţi pentru rezolvarea problemei, avem la dispoziţie o relaţie de recurenţă sau putem evidenţia cele trei tipuri de expresii (Ei, Et, Em) referitoare la o anumită variabilă folosită în descrierea algoritmului.

3. TEMA

Să se facă analiza şi 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 ieşire şi descrierea algoritmului (prin formule sau prin limbaj natural).

3.1. Selecţia

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 afişeze câtul împărţirii numărului mai mare la cel mai mic.

Problema nr. A3

Se citesc de pe dispozitivul de intrare 3 numere reale care reprezintă coeficienţii unei ecuaţii de gradul 2. Să se determine rădăcinile acestei ecuaţii. În cazul în care ecuaţia nu poate fi rezolvată se vor afişa mesaje corespunzătoare.

Problema nr. A4

Un punct din plan este dat prin coordonatele sale (x, y). Să se stabilească poziţia punctului prin indicarea cadranului (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 şi secundă) şi momentul deconectării (de asemenea dat prin oră, minut şi secundă). Trebuie să se ia în consideraţie cazul în care o conexiune începe într-o zi şi se încheie în ziua următoare. Atenţie: rezultatul trebuie afişat 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 cunoaşte momentul de conectare (dat prin oră, minut şi secundă) şi durata totală a conexiunii (dată prin ore, minute şi secunde). Se ia în consideraţie şi cazul în care o conexiune începută într-o anumită i se poate încheia în ziua următoare. Atenţie: rezultatul trebuie afişat 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 afişa pe dispozitivul de ieşire (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 funcţie:

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

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

3.2. Iteraţia

Problema nr. B1

a) Să se deseneze schema logică pentru un program care afişează un tabel care pe prima coloană are temperaturi Fahrenheit şi pe cea de-a doua coloană echivalenţele 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 şi 300, cu un pas egal cu 10°F.

b) Să se deseneze schema logică pentru tabelul corespunzător de tranformare 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 citeşte un număr real b şi un şir de valori reale pozitive terminate printr-o valoare negativă (care nu face parte din şir). Să se stabilească elementul din şir cel mai apropiat de b. Se va preciza şi poziţia acestuia (al câtelea element din şir este). Termenii şirului vor fi păstraţi în aceeaşi variabilă a.

Problema nr. B3

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

Problema nr. B4

Şirul $\{x_n\}$ generat cu relaţia de recurenţă $x_n = \frac{1}{p}\left[(p-1)x_{n-1} + \frac{a}{(x_{n-1})^{p-1}}\right]$, cu $a \in R$ şi $p \in N$, şi $x_0 = \frac{a}{p}$ este convergent şi 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 şir cu precizia ε de asemenea dată.

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

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 condiţia $\left|S_n - \frac{\pi}{3}\right| < \varepsilon$, în care ε este dat. Se ştie că $\lim_{n\to\infty} S_n = \frac{\pi}{3}$.

Problema nr. B6

Să se calculeze funcţia $J_n(x)$ ştiind că există relaţia de recurenţă:

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

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

Calculele se fac cu precizia ε. De pe dispozitivul de intrare se citesc x, n şi ε.

Problema nr. B7

Fie x un număr real şi n un număr întreg. Să se calculeze x^n folosind un număr cât mai mic de înmulţiri de numere reale.

Indicaţie. Se observă că x^n poate fi descompus într-un produs de factori în care pot apare $x, x^2, x^4, x^8 \ldots$ 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 puţin semnificativă şi puterile lui x: $x, x^2, x^4, x^8 \ldots$ 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 obţin ca resturi ale împărţirii succesive a lui n la 2.

Problema nr. B8

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

Indicaţie. Înmulţim ecuaţia cu x^{n-3}: $ax^n + bx^{n-2} + cx^{n-3} = 0$ şi ţinând cont de faptul că fiecare rădăcină satisface ecuaţia 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 relaţii şi obţinem:

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

Notând $s_n = x_1^n + x_2^n + x_3^n$, obţinem relaţia de recurenţă:

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

cu următoarele valori iniţiale:

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

$s_1 = x_1 + x_2 + x_3 = 0$ (relaţiile Viete)

$s_2 = x_1^2 + x_2^2 + x_3^2 = -2b/a$ (relaţiile Viete)