

LABORATOR NR. 3

Instrucțiuni pentru implementarea structurilor de control

Se reiau o parte din noțiunile prezentate în laboratorul nr. 1 indicând și instrucțiunile **C** corespunzătoare celor mai utilizate structuri de control.

1. Selecția (decizia) – instrucțiunea **if**

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.

Reprezentare prin schemă logică a selecției cu două alternative este dată în Figura 1.

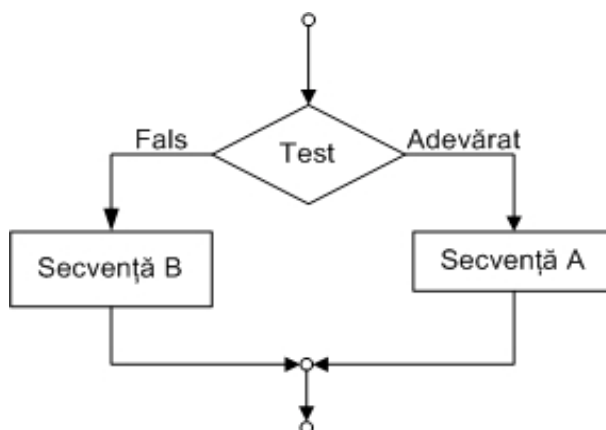


Figura 1. Selecția cu două alternative

Instrucțiunea corespunzătoare este instrucțiunea **if**.

Forma instrucțiunii **if** este:

```
if(test)
{
    Secventa A
}
else
{
    Secventa B
}
```

2. 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

2.1. Repetiție (iterație) cu test inițial – instrucțiunea **while**

Schema logică este dată în Figura nr. 2:

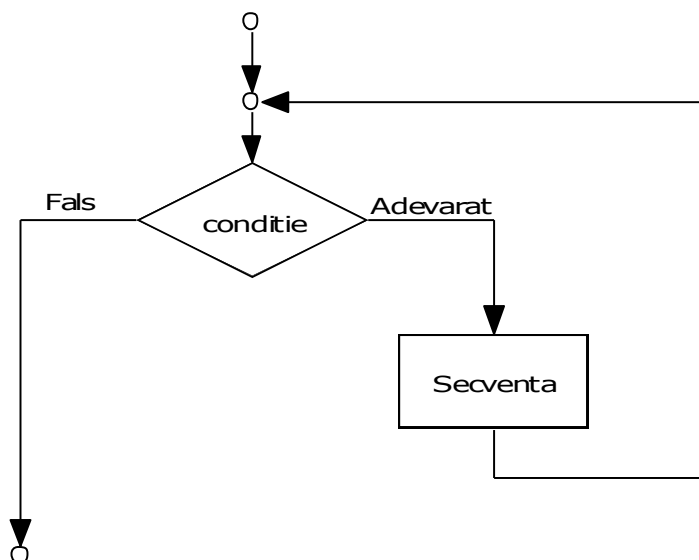


Figura nr. 2. Iterația cu test inițial

Instrucțiunea **C** corespunzătoare este instrucțiunea **while**. Forma generală a instrucțiunii este:

```

while (conditie)
{
    Secventa
}
  
```

2.2. Repetiție (iterație) cu test final – instrucțiunea **do ... while**

Schema logică este dată în Figura nr. 3.

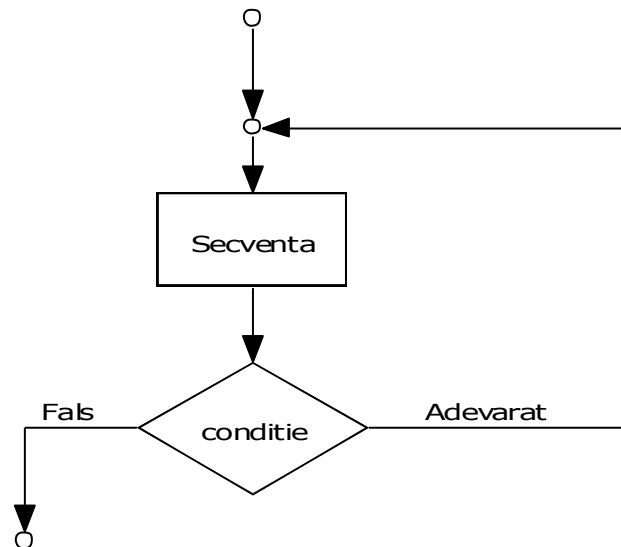


Figura nr. 3. Iterația cu test final

Instrucțiunea corespunzătoare este **do ... while** a cărei formă este:

```

do {
    Secventa
} while(conditie);
  
```

2.3. Repetiție (iterație) cu contor – instrucțiunea **for**

Reprezentarea prin schema logică este dată în Figura nr. 4, forma generală fiind dată în Figura nr. 5.

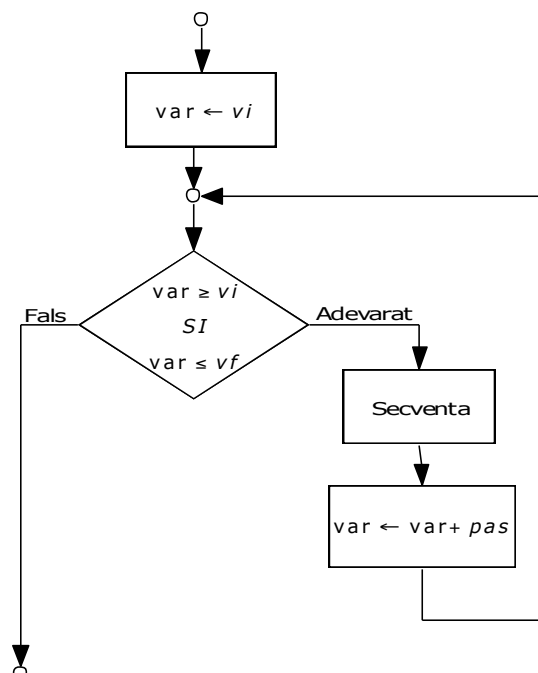


Figura nr. 4. Iterația cu contor

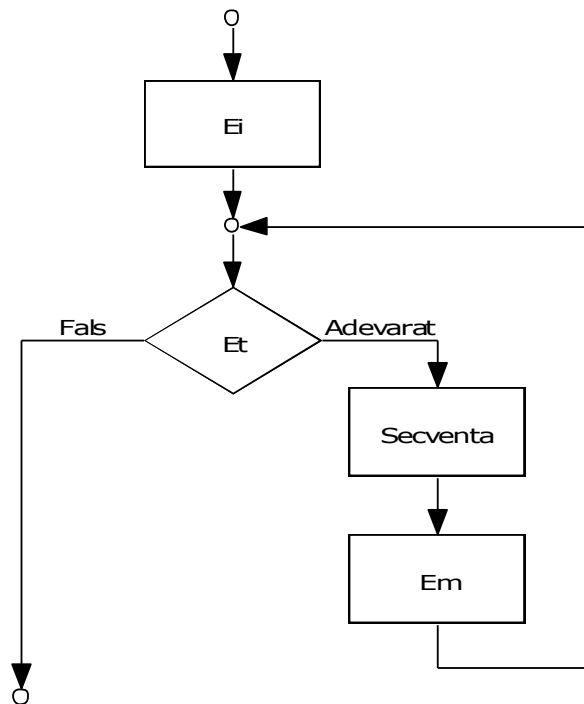


Figura nr. 5. Iterația cu contor (forma generală)

Instrucțiunea corespunzătoare acestei structuri de control este **for**.
Forma generală corespunzătoare schemei logice din Figura 5 este:

```
for(ei; et; em)
{
    Secventa
}
```

În cazul (cel mai întâlnit în aplicațiile care urmează) în care avem o variabilă **i** care are ca valoare inițială valoarea 0 (zero), variază prin creșterea valorii sale cu o unitate și cu această variabilă **i** trebuie să numărăm **n** (**n** număr natural) pași forma instrucțiunii **for** este următoarea:

```
for(i=0; i<n; i=i+1)
{
    secventa;
}
```

3. TEMA 1

În toate problemele din laboratorul 2 s-a folosit structura de control numită decizie.

În acest laborator ne vom concentra asupra repetiției (sau iterației).

Problema nr. 1.1

Să se scrie un program care citește n numere întregi, calculează suma lor și o afișează pe monitor – nu se vor folosi vectori.

Problema nr. 1.2

Să se scrie un program care citește un șir de numere întregi până când se introduce valoarea 0, calculează produsul lor și afișează valoarea produsului pe monitor – nu se vor folosi vectori.

Problema nr. 1.3

Să se scrie un program care citește un șir de numere reale până când se introduce valoarea 0, calculează produsul lor și afișează valoarea produsului pe monitor, cu două zecimale – nu se vor folosi vectori.

Problema nr. 1.4

a) Să se scrie 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°F – nu se vor folosi vectori.

b) Să se scrie un program care afișează tabelul corespunzător de transformare din grade Celsius în grade Fahrenheit. De această dată pasul va fi citit de pe dispozitivul de intrare, iar valorile gradelor Celsius variază între 0 și 400°C – nu se vor folosi vectori.

Problema nr. 1.5

Să se scrie un program care determină cel mai mare divizor comun pentru două numere întregi strict pozitive după algoritmul lui Euclid: se împarte numărul mai mare la cel mai mic atât timp cât împărțitorul este nenul; la fiecare pas, până când restul este zero, se înlocuiește deîmpărțitul cu împărțitorul și împărțitorul cu restul; ultimul împărțitor nenul este c.m.m.d.c. Pentru rest se folosește operatorul modulo: % (exemplu: $a\%b$ înseamnă restul împărțirii lui a la b) – se va scrie o funcție pentru calculul cmmdc.

4. TEMA 2

Problema nr. 2.1

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. 2.2

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. 2.3

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 ε este dat. Se știe că $\lim_{n \rightarrow \infty} S_n = \frac{\pi}{3}$ (asigurându-na astfel existența unei valori pentru n).

Problema nr. 2.4

O pereche de numere naturale a și b se numesc prietene dacă suma divizorilor unuia din numere este egală cu celălalt număr (și invers). De exemplu, 220 și 284 sunt numere prietene deoarece:

1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110 sunt divizori ai numărului 220 și suma lor este 284

1, 2, 4, 71, 142 sunt divizori ai numărului 284 și suma lor este 220.

Să se scrie un program care găsește toate numerele prietene dintr-un interval dat de numere naturale $[x, y]$. Cele două numere care reprezintă capetele intervalului sunt citite de la tastatură și programul face și validarea datelor (verifică corectitudinea datelor de intrare). Nu se vor folosi tablouri.

Indicații:

În afara funcției **main** se vor mai scrie două funcții ale căror prototipuri se vor indica într-un fișier header:

1. o funcție care primește ca parametru un număr natural și returnează suma divizorilor numărului.
2. o funcție având ca parametri două numere naturale și care returnează **1** sau **0** după cum cele numere sunt sau nu prietene.

Problema nr. 2.5

Numerele naturale pot fi clasificate în: deficiente, perfecte sau abundente după cum suma divizorilor este mai mică, egală sau mai mare decât valoarea numărului. Astfel:

$n = 12$ este abundent deoarece are suma divizorilor $sd = 1+2+3+4+6 = 16 > 12$,

$n = 6$ este perfect pentru că $sd = 1+2+3 = 6$,

$n=14$ este deficient deoarece $sd = 1 + 2 + 7 = 10 < 14$.

Scrieți un program care citește două valori naturale x și y și, în funcție de opțiunea utilizatorului, afișează toate numerele perfecte, abundente sau deficiente din intervalul $[x, y]$. Programul trebuie să conțină o secvență de validare a datelor. Nu se vor folosi tablouri.

Indicații:

În afara funcției **main** se vor mai scrie două funcții ale căror prototipuri se vor indica într-un fișier header:

1. o funcție care primește ca parametru un număr natural și returnează suma divizorilor numărului.
2. o funcție având ca parametru un număr natural, funcție care returnează -1, 1 sau 0 după cum numărul transmis ca parametru este deficient, perfect sau abundent.

5. TEMA 3

Problema nr. 3.1

Se citesc de la tastatură un număr n de numere naturale nenule. Să se stabilească dacă acestea:

- formează un șir strict crescător (tip șir = 6)
- formează un șir crescător (tip șir = 5)
- formează un șir strict descrescător (tip șir = 2)
- formează un șir descrescător (tip șir = 3)
- sunt identice (șirul este constant) (tip șir = 4)
- nu sunt ordonate (tip șir = 1)

Programul are posibilitatea de a analiza mai multe seturi de date (va exista posibilitatea de reluare a programului).

Observație:

Numerele citite **nu** se vor memora într-un vector (tablou).

Indicații:

Intre doua elemente ale șirului se pot stabili următoarele relații de ordine:

- mai mic
- egal
- mai mare

Se determina numărul de relații de ordine de fiecare tip și în funcție de acest număr se stabilește tipul șirului astfel:

- a. numai relații mai mic - sir strict crescător
- b. relații mai si relații egal - sir crescător
- c. numai relații egal - sir constant
- d. relații egal si relații mai mare - sir descrescător
- e. numai relații mai mare - sir strict descrescător
- f. toate tipurile de relații - sir oarecare

În afara funcției **main** se vor mai scrie două funcții ale căror prototipuri se vor indica într-un fișier header:

1. o funcție care nu are nici un parametru și returnează o valoare în funcție de tipul șirului analizat. Această funcție citește și analizează șirul.
2. o funcție care primește ca parametru un număr natural reprezentând tipul șirului și afișează mesajul corespunzător.

Problema nr. 3.2

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 $a \in \mathbb{R}$ și

$p \in \mathbb{N}$, 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 ε 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. 3.3

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 ε . De pe dispozitivul de intrare se citesc x, n și ε .

Problema nr. 3.4

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. 3.5

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:

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

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

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