LABORATOR NR. 3
Instrucţiuni pentru repetiție (iterație)

1. 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

1.1. Repetiţie (iteraţie) cu test iniţial – instrucțiunea while

Schema logică este dată în Figura nr. 2:

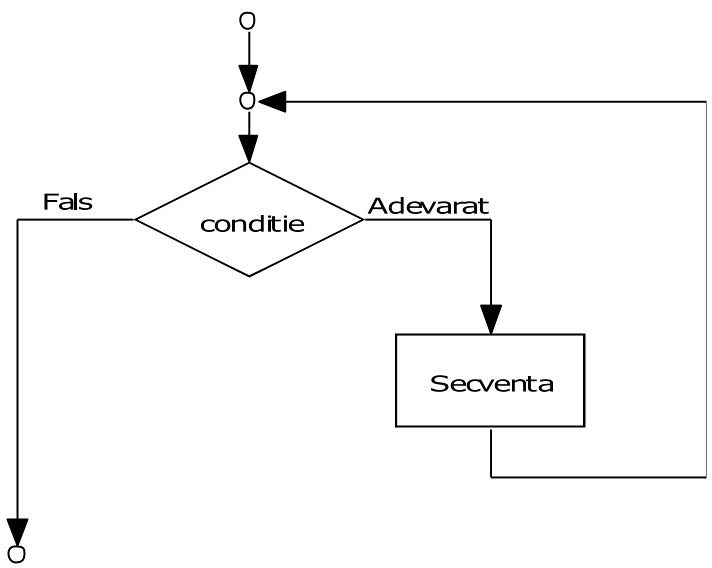

Figura nr. 2. Iteraţia cu test iniţial

Instrucţiunea **C** corespunzătoare este instrucţiunea **while**. Forma generală a instrucţiunii este:

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

1.2. Repetiţie (iteraţie) cu test final – instrucțiunea do ... while

Schema logică este dată în Figura nr. 3.

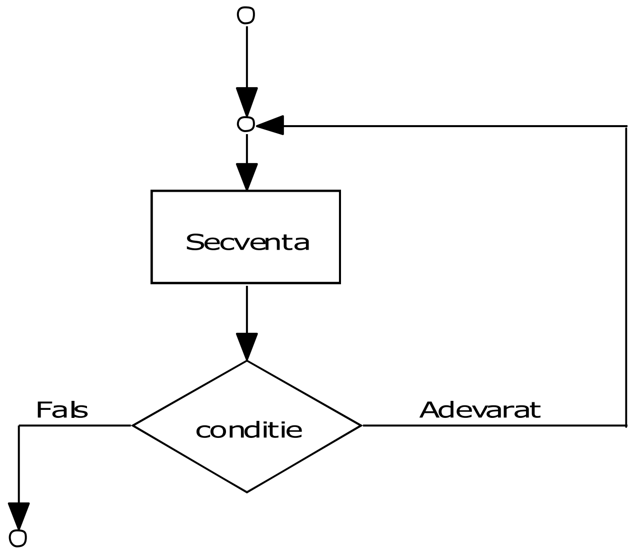

Figura nr. 3. Iteraţia cu test final

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

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

1.3. Repetiţie (iteraţie) cu contor – instrucțiunea for

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

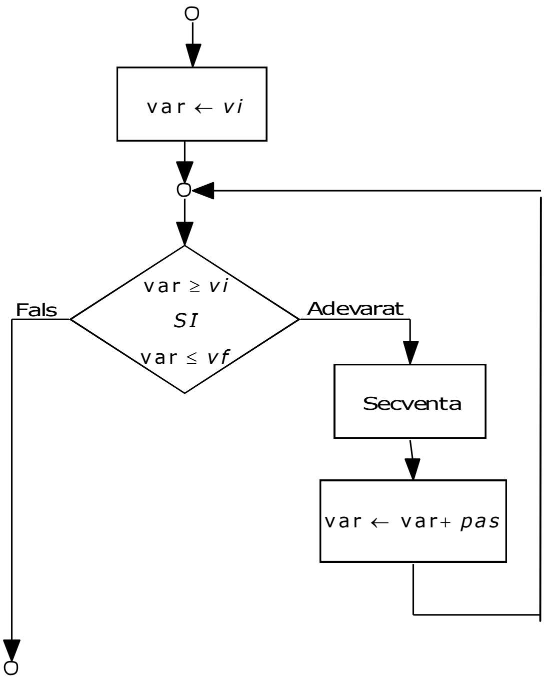

Figura nr. 4. Iteraţia cu contor

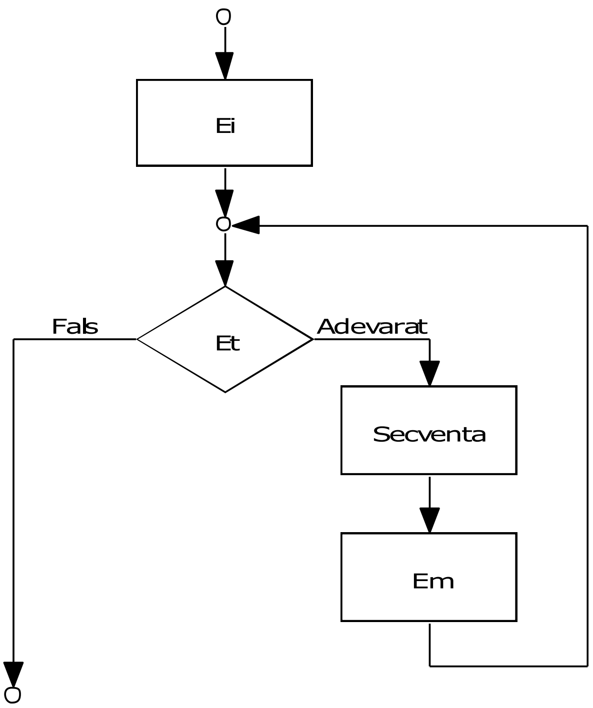

Figura nr. 5. Iteraţia cu contor (forma generală)

Instrucţiunea 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 aplicaţiile care urmează) în care avem o variabilă **i** care are ca valoare iniţială valoarea 0 (zero), variază prin creşterea valorii sale cu o unitate şi cu această variabilă **i** trebuie să numărăm **n** (**n** număr natural) paşi forma instrucţiunii **for** este următoarea:

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

TEMA 1

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

În acest laborator ne vom concentra asupra repetiţiei (sau iteraţiei).

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 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^{o}F – nu se vor folosi vectori.

b) Să se scrie un program care afişează tabelul corespunzător de tranformare 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 şi 400^{0}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 împărţitorul este nenul; la fiecare pas, până când restul este zero, se înlocuieşte deîmpărţitul cu împărţitorul şi împărţitorul cu restul; ultimul împărţitor nenul este c.m.m.d.c. Pentru rest se foloseşte operatorul modulo: % (exemplu: a%b înseamnă restul împărţirii lui a la b) – se va scrie o funcţie pentru calculul cmmdc.

TEMA 2

Problema nr. 2.1

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. 2.2

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. 2.3

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 $\lim_{n \to \infty} S_n = \frac{\pi}{3}$ (asigurându-na astfel existenţa unei valori pentru **n**).

Problema nr. 2.4

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

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

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

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

Indicaţii:

În afara funcţiei **main** se vor mai scrie două funcţii ale căror prototipuri se vor indica într-un fişier header:

1. o funcţie care primeşte ca parametru un număr natural şi returnează suma divizorilor numărului.
2. o funcţie având ca parametri două numere naturale şi 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.

Scrieţi un program care citeşte două valori naturale x şi y şi, în funcţie de opţiunea utilizatorului, afişează toate numerele perfecte, abundente sau deficiente din intervalul [x, y]. Programul trebuie să conţină o secvenţă de validare a datelor. Nu se vor folosi tablouri.

Indicaţii:

În afara funcţiei **main** se vor mai scrie două funcţii ale căror prototipuri se vor indica într-un fişier header:

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

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

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

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

Indicaţii:

Intre doua elemente ale şirului se pot stabili următoarele relaţii de ordine:
- mai mic
- egal
- mai mare

Se determina numărul de relaţii de ordine de fiecare tip şi în funcţie de acest număr se stabileşte tipul şirului astfel:

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

În afara funcţiei **main** se vor mai scrie două funcţii ale căror prototipuri se vor indica într-un fişier header:

1. o funcţie care nu are nici un parametru şi returnează o valoare în funcţie de tipul şirului analizat. Această funcţie citeşte şi analizează şirul.
2. o funcţie care primeşte ca parametru un număr natural reprezentând tipul şirului şi afişează mesajul corespunzător.

Problema nr. 3.2

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. 3.3

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. 3.4

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: $x^{13}=x^8x^4x$ 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. 3.5

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)