Lucrarea de laborator nr. 4
TABLOURI ŞI FUNCŢII.

1. Declaraţia / definiţia unui tablou

Tabloul este o mulţime ordonată de elemente la care ne putem referi cu ajutorul unor indici. Orice tablou are un nume, iar tipul comun al elementelor este tipul tabloului. În limbajul **C** există doar tablouri unidimensionale. Tablourile multidimensionale sunt considerate tot tablouri unidimensionale în care fiecare element este la rândul său un tablou. Declaraţia unui tablou unidimensional este de forma:

T nume[dim];

unde :

T	este unul din tipurile existente qin limbaj
nume	este numele tabloului
dim	este o valoare **constantă** care reprezintă numărul de elemente din tablou.

Orice declaraţie de tablou se termină cu **;**. **După declarare, dacă nu au fost iniţializate, elementele tabloului au valori nedefinite.**

La definirea unui tablou se alocă o zonă de memorie necesară pentru a memora elementele tabloului. La primul element ne vom referi prin nume[0], la al doilea prin nume[1]...la ultimul prin nume[dim-1], deci indicele de parcurgere a tabloului variază intre 0 şi dim-1.

Determinarea numărului de octeţi rezervaţi tabloul identificat prin **nume** se face prin utilizarea expresiei:

sizeof(nume)

Rezultatul acestei expresii trebuie să coincidă cu valoarea dată de expresia:

dim * sizeof(**T**)

Exemplu 1.

Definirea unui tablou unidimensional cu 10 elemente de tip întreg cu semn al cărui nume este **a**.

Int a[10];

Pentru acest tablou se vor rezerva, la compilare, 10*sizeof(int) octeţi.

sizeof(a) va avea acelaşi rezultat.

Se pot declara mai multe tablouri cu acelaşi tip de elemente pe aceeaşi linie, caz în care se foloseşte separatorul virgulă.

Exemplu 2.

Definirea a trei tablouri cu elemente de tip real simplu (float).

```
float a[15], b[5], vector[20];
```

Şi în cazul definirii tablourilor se recomandă definirea unui singur tablou pe o linie pentru a da posibilitatea inserării de comentarii pentru buna documentare a programului.

Tablourile care au ca elemente alte tablouri sunt de fapt tablouri multidimensionale, astfel definiţia:

```
double mat[2][3];
```

este un tablou cu două elemente, în care fiecare element este un tablou de trei elemente de tip real dublu. (Gândiţi-vă că acest tablou este o matrice cu două linii, iar fiecare linie are trei elemente de tip real dublu).

Pentru cazul general

T nume[dim_1][dim_2]...[dim_n];

unde **nume** este numele unui tablou cu dim_1 elemente, în care fiecare din cele dim_1 este un tablou cu dim_2 elemente, în care fiecare din cele dim_2 elemente este un tablou ..., iar **T** este tipul elementelor din tablou de dimensiune dim_n.

2. Funcţii

2.1. Antetul unei funcţii

O funcţie este o entitate care prelucrează informaţie într-un program. Pentru o funcţie putem construi următoare diagramă:

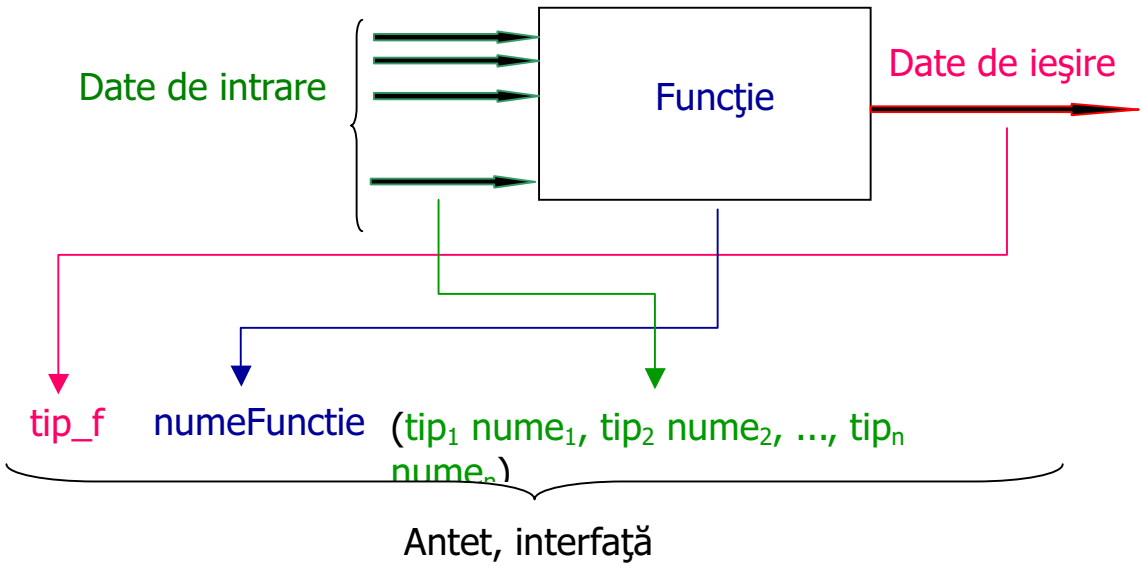

Figura 1. Diagrama asociată unei funcţii

Astfel putem privi funcţia (identificată prin numeFunctie) ca o cutie care conţine un mecanism care prelucrează informaţiile primite la intrare, numite şi date de intrare. Lista cu tipul şi, eventual, numele asociate fiecărei date de intrare sunt indicată în antetul funcţiei, între paranteze, după numele funcţiei (înscrisă cu verde în Figura 1). Lista cu tipul şi numele informaţiilor de la intrare se mai numeşte şi **lista cu parametri formali** pentru funcţie. O funcţie poate primi la intrare una sau mai multe informaţii sau nicio informaţie. La ieşire, funcţia furnizează rezultatul prelucrărilor. În antetul funcţiei se specifică doar tipul informaţiei care "iese" din funcţie (tip_f înscris cu roşu în Figura 1). La ieşire funcţia poate furniza cel mult o informaţie. Tipul informaţiei furnizate de funcţie în exterior dă **tipul funcţiei**.

În cazurile în care fie funcţia nu primeşte informaţii din exterior (lista de parametri este vidă), fie funcţia nu furnizează informaţii în exterior se specifică acest lucru prin folosirea cuvântului cheie **void**.

Observaţie: o funcţie nu poate furniza în exterior un tablou, deci nu poate avea tipul tablou.

Exemplu 3.

a) antetul unei funcţii al cărui nume este **suma** şi care primeşte din exterior două informaţii de tip real dublu şi furnizează în exterior un rezultat de tip întreg:

```
int suma(double a, double b)
```

b) antetul unei funcţii al cărui nume este **lama** şi care nu primeşte nici o informaţie din exterior şi furnizează în exterior un rezultat de tip char:

```
char lama(void)
```

c) antetul unei funcţii al cărui nume este **citesteVector** şi care primeşte din exterior două informaţii una referitoare la un întreg cu semn şi cealaltă referitoare la un tablou de reali dubli şi nu furnizează în exterior niciun rezultat:

```
void citesteVector(int n, double a[])
```

2.2. Declaraţia (prototipul) unei funcţii

Declaraţia unei funcţii se găseşte într-un fişier header (sau antet) şi este compusă din antetul funcţiei (Figura 1) urmat de semnul de caracterul **;**. Prototipul unei funcţii nu "spune" cum prelucrază funcţia informaţiile primite, ci indică numai legaăturile sale cu exteriorul.

Exemplu 4.

Prototipurile funcţiilor din Exemplu 3. sunt:

a) int suma(double a, double b);
b) char lama(void);
c) void citesteVector(int n, double a[]);

2.3. Definiţia unei funcţii

Definiţia unei funcţii se va găsi întotdeauna într-un fişier cu cod sursă şi este formată din antetul funcţiei urmat de un bloc în care se găseşte algoritmul, exprimat în limbajul de programare **C**, care descrie modul în care funcţia realizează prelucrările informaţiei primite din exterior. Blocul începe întotdeauna cu caracterul **{** şi se încheie cu **}**.

Variabilele (de orice tip) definite în interiorul unei funcţii sunt locale funcţiei respective şi **nu sunt recunoscute în afara funcţiei**, iar modificările făcute asupra variabile din interiorul funcţiei, cât şi asupra parametrilor de intrare se vor pierde la ieşirea din funcţie. De aceea, rezultatul pe care funcţia trebuie să-l trimită în exterior trebuie returnat prin folosirea instrucţiunii **return**.

Forma generală a aceste instrucţiuni este:

```
return expresie;
```

Observaţie: modificările făcute asupra elementelor unui tablou se păstrează şi după ieşirea din funcţie.

Exemplu 5.

Dacă funcţia cu prototipul din Exemplu 4.a). trebuie să calculeze partea întreagă a sumei celor doi reali primiţi care parametri de intrare atunci definiţia respectivei funcţii este:

```
int suma(double a, double b)
{
    int s;
    s = floor(a+b);
    return s;
}
```

În exemplul dat mai sus partea scrisă cu roşu reprezintă blocul (sau corpul funcţiei), iar **floor** este o funcţie care calculează aproximarea prin lipsă a unui real (funcţie existentă în biblioteca matematică).

Observaţie: nu putem avea acelaşi nume pentru o funcţie şi pentru o variabilă.

Pentru rezolvarea corectă a problemelor ce urmează vă rog să citiţi cu atenţie fiecare enunţ de la început până la sfârşit.
Rezolvările se fac folosind proiecte şi fără variabile globale.

TEMA 1

Problema 1.1.

Să se scrie un program care citeşte de la tastatură un număr întreg n şi două seturi de câte **n** elemente de tip întreg, fiecare set fiind stocat într-un tablou unidimensional cu maxim 30 de elemente, afişează cele două tablouri, calculează tabloul (vectorul) sumă şi afişează vectorul rezultat. Rezolvarea trebuie să conţină funcţii scrise pentru: citirea de la tastatură a unui vector, afişarea pe monitor a unui vector, calculul vectorului sumă. (se scrie funcţie pentru citirea unui vector, care se apelează în această problemă de două ori).

Observaţie: problema se bazează pe exemplele de la curs.

Problema 1.2.

Să se scrie un program care citeşte de la tastatură un tablou de maximum 20 de numere întregi, determină maximul, minimul, media aritmetică şi media geometrică şi

afişează rezultatele. Pentru media geometrică se va folosi funcţia **pow** (al cărui prototip este:

double pow(double x, double y); /* returnează x la puterea y */

şi se găseşte în MATH.H.

Se vor scrie funcţii adecvate pentru citirea unui vector, scrierea unui vector, calcularea maximului dintre elementele unui vector, calcularea minimului dintre elementele unui vector, calcularea mediei aritmetice a valorilor elementelor unui vector, calcularea mediei geometrice a valorilor elementelor unui vector.

Problema 1.3.

Să se scrie un program care citeşte de la tastatură un tablou de maximum 30 de numere întregi, afişează pe monitor vectorul citit, ordonează elementele vectorului crescător prin metoda bulelor şi afişează rezultatul pe monitor. Se vor scrie funcţii pentru citirea unui vector de numere întregi, scrierea unui vector de numere întregi şi ordonare prin metoda bulelor.

Schema logică a programului care ordonează crescător un vector este dată în Figura 2, iar schema logică a funcţiei pentru sortarea prin metoda bulelor este dată în Figura 3.

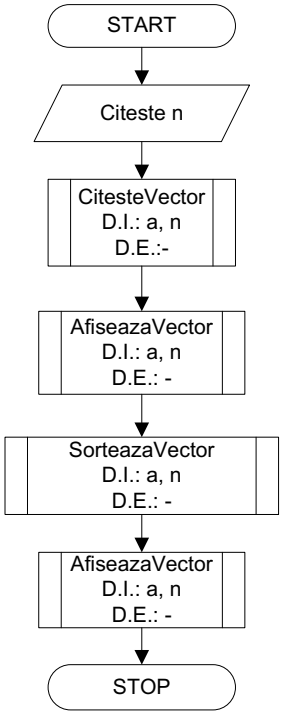

Figura 2. Schema logică a programului de ordonare (sortare) în sens crescător a elementelor unui vector

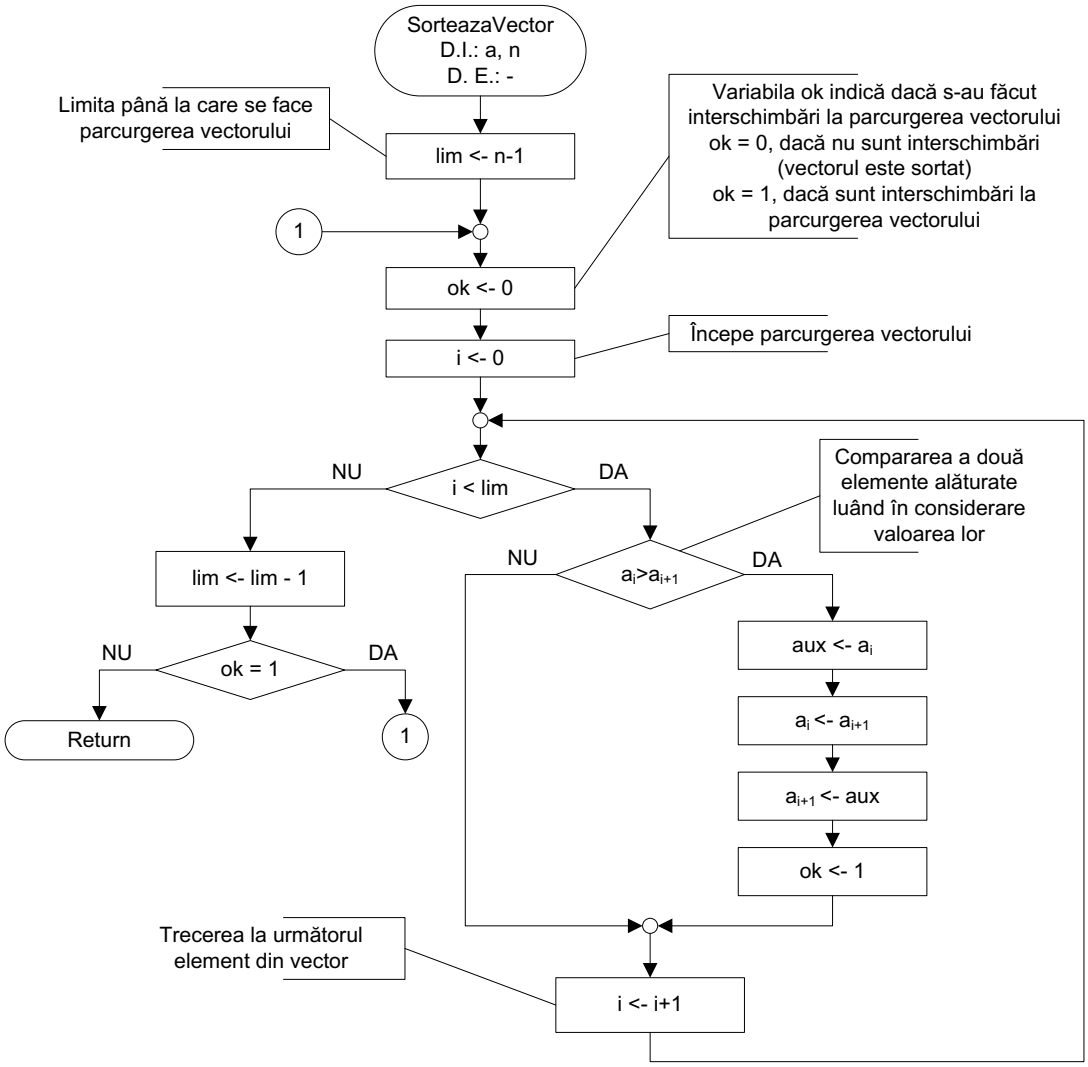

Figura 3. Schema logică a funcţie de sortare în sens crescător a elementelor unui vector (foloseşte sortarea prin metoda bulelor)

TEMA 2

Problema 2.1.

În urma unor măsurători făcute în cadrul unui experiment rezultă un şir de **n** date experimentale (de tip real) care se memorează într-un vector (tablou unidimensional) **x**. Prelucrarea acestor date experimentale înseamnă:

a. calcularea valorii medii cu formula

$$x_m = \frac{1}{n}\sum_{i=0}^{n-1} x_i$$

b. calcularea abaterii medii pătratice cu formula:

$$x_p = \sqrt{\frac{\sum_{i=0}^{n-1}(x_i - x_m)^2}{n(n-1)}}$$

c. afişarea numărului de componente care nu depăşesc valoarea medie

d. crearea unui alt vector **y** cu elementele din **x** mai mari decât valoarea medie şi afişarea vectorului **y** cu câte 5 elemente pe o linie.

Să se scrie un program pentru a putea prelucra datele experimentale obţinute. Pentru rezolvarea problemei se vor folosi funcţii pentru:

1. citirea unui şir de numere cu prototipul:
   void citesteRVector(double a[], int n);
2. afişarea unui şir de numere cu prototipul:
   void scrieRVector(double a[], int n);
3. fiecare din cele 4 tipuri de prelucrări (câte o funcţie pentru fiecare prelucrare). Aceste funcţii vor returna rezultatul calculat, scrierea rezultatului pe monitori făcându-se în funcţia **main**.

Pentru punctul **d** se vor scrie două funcţii: una pentru crearea vectorului **y** având ca date de intrare vectorul **x**, numărul de elemente din **x** şi vectorul **y** şi ca date de ieşire numărul de elemente din **y** şi cea de a doua pentru scrierea unui vector cu câte 5 elemente pe o linie.

Prototipurile funcţiilor se vor găsi într-un fişier header după modelul dat la curs.

Problema 2.2.

Se citeşte de la tastatură un număr natural **n** (**n** cel mult egal cu 100) şi un şir de **n** numere întregi.. Şirul de numere se memorează într-un tablou.

Să se afişeze elementele distincte.

Exemplu: în şirul **2 2 5 4 5 1 2** elementele distincte sunt **2 5 4 1**.

Se vor scrie funcţii pentru:

1. citirea unui şir de numere cu prototipul (de la problema anterioară):
   void citireIVector(int a[], int n);
   // funcţia returnează numărul de elemente efectiv citite
2. afişarea unui şir de numere cu prototipul (de la problema anterioară):
   void afisareIVector(int a[], int n);
3. determinarea şirului cu elemente distincte cu prototipul:

int distinct(int a[], int b[], int n);

// funcţia are ca parametri de intrare şirul iniţial (**a**), şirul final (cu elemente distincte **b**) şi numărul de elemente din şirul iniţial **n** şi returnează numărul de elemente din şirul final.

4. funcţie care verifică dacă un element se găseşte sau nu într-un vector. Funcţia returnează valoarea 1 dacă elementul se găseşte în vector şi 0 în caz contrar. Prototipul funcţiei este:

int gasit(int a[], int n, int el);

în care **a** este vectorul în care se caută, **n** numărul de elemente din aceste vector şi **el** este elementul care se caută.

Prototipurile funcţiilor se vor găsi într-un fişier header după modelul dat la curs.

Problema 2.3.

În cadrul unui laborator de analize medicale se fac măsurători care privesc modul în care evoluează în timp concentraţia unei anumite componente în proba de laborator analizată. Pentru aceasta se notează pe o fişă concentraţia şi momentul de timp corespunzător (considerând momentul de început al analizei ca moment zero). Se obţin astfel două şiruri de valori: (i) şirul cu valorile momentelor de timp şi (ii) şirul cu valorile concentraţiilor la aceste momente de timp.

Se cere să se scrie un program care citeşte cele două şiruri de valori (care sunt numere reale) şi afişează coeficienţii dreptei de regresie sau afişează un mesaj corespunzător dacă aceştia nu pot fi calculaţi.

Dreapta de regresie $x = a * t + b$ are proprietatea că pentru punctele din plan valoarea expresiei

$$E = \sum_{i=0}^{n-1} (x_i - a \cdot t_i - b)^2$$

este minimă, unde *n* este numărul de măsurători făcute.

Coeficienţii *a* şi *b* se determină prin rezolvarea sistemului:

$$\begin{cases} n \cdot a + b \cdot \sum_{i=0}^{n-1} (t_i) = \sum_{i=0}^{n-1} (x_i) \\ a \cdot \sum_{i=0}^{n-1} (t_i) + b \cdot \sum_{i=0}^{n-1} (t_i \cdot t_i) = \sum_{i=0}^{n-1} (t_i \cdot x_i) \end{cases}$$

Pentru rezolvarea problemei se vor scrie următoarele funcţii:

a. funcţie pentru citirea unui vector
b. funcţie pentru afişarea unui vector

(Aceste două funcţii sunt cele scrise pentru rezolvarea problemei 1)

c. funcţie pentru rezolvarea unui sistem de două ecuaţii cu două necunoscute. Această funcţie are doi parametri:

- un tablou cu 6 elemente conţinând coeficienţii celor două ecuaţii;
- un tablou cu două elemente conţinând soluţiile sistemului

şi returnează rezultatul:

1 – dacă sistemul este compatibil determinat

2 – dacă sistemul este compatibil nedeterminat

3 – dacă sistemul este incompatibil

d. funcţie care calculează produsul scalar a doi vectori *a* şi *b*. Funcţia are ca pearametri cei doi vectori şi numărul de elemente al unui vector şi returnează valoarea produsului scalar. Pentru calculul sumei valorilor elementelor unui vector folosind funcţia produs scalar unul din vectori va fi un vector iniţializat cu unităţi.

Observaţie:

Toate sumele se vor calcula folosind funcţia produs scalar.

Prototipurile funcţiilor se vor găsi într-un fişier header după modelul dat la curs.