

Lucrarea de laborator nr. 4

TABLOURI ȘI FUNCȚII.

1. Declarația / definiția unui tablou

Tabloul este o mulțime 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. Declarația 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 declarație de tablou se termină cu ;. **După declarare, dacă nu au fost inițializate, 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 între 0 și dim-1.

Determinarea numărului de octeți rezervați 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, $10 * \text{sizeof}(\text{int})$ octeți.
`sizeof(a)` va avea același rezultat.

Se pot declara mai multe tablouri cu același tip de elemente pe aceeași linie, caz în care se folosește separatorul virgulă.

Exemplu 2.

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

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

Și 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 definiția:

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

este un tablou cu două elemente, în care fiecare element este un tablou de trei elemente de tip real dublu. (Gândiți-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[dim1][dim2]...[dimn];
```

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. Funcții

2.1. Antetul unei funcții

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

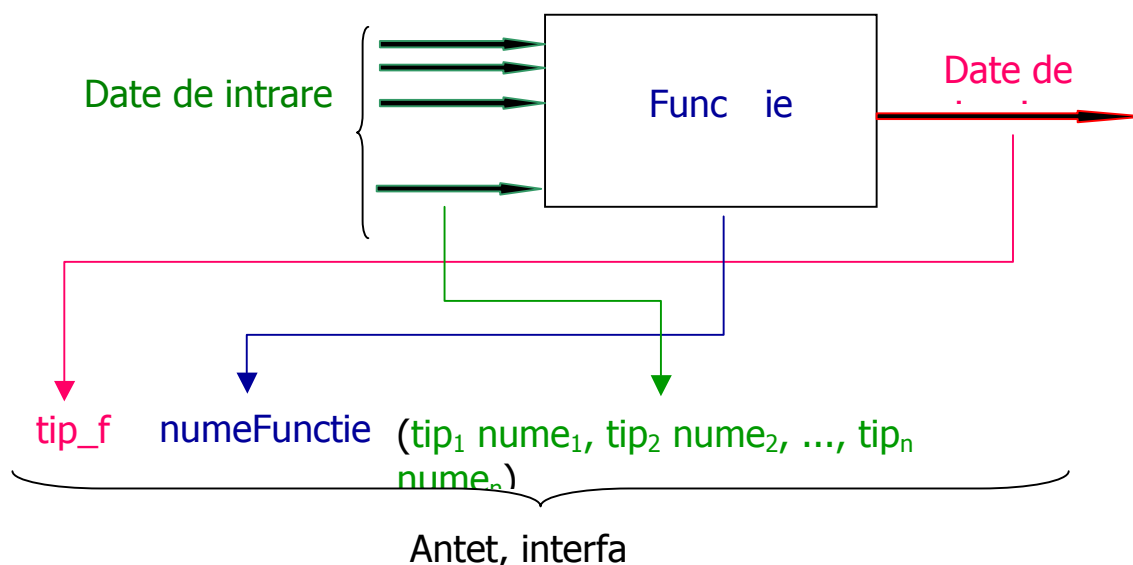


Figura 1. Diagrama asociată unei funcții

Astfel putem privi funcția (identificată prin `numeFuncție`) ca o cutie care conține un mecanism care prelucrează informațiile primite la intrare, numite și date de intrare. Lista cu tipul și, eventual, numele asociate fiecărei date de intrare sunt indicată în antetul funcției, între paranteze, după numele funcției (înscrișă cu verde în Figura 1). Lista cu tipul și numele informațiilor de la intrare se mai numește și **lista cu parametri formali** pentru funcție. O funcție poate primi la intrare una sau mai multe informații sau nicio informație. La ieșire, funcția furnizează rezultatul prelucrărilor. În antetul funcției se specifică doar tipul informației care "iese" din funcție (`tip_f` înscriș cu roșu în Figura 1). La ieșire funcția poate furniza cel mult o informație. Tipul informației furnizate de funcție în exterior dă **tipul funcției**.

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

Observație: o funcție nu poate furniza în exterior un tablou, deci nu poate avea tipul tablou.

Exemplu 3.

a) antetul unei funcții al cărui nume este **suma** și care primește din exterior două informații de tip real dublu și furnizează în exterior un rezultat de tip întreg:

`int suma(double a, double b)`

b) antetul unei funcții al cărui nume este **lama** și care nu primește nici o informație din exterior și furnizează în exterior un rezultat de tip char:

```
char lama(void)
```

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

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

2.2. Declarația (prototipul) unei funcții

Declarația unei funcții se găsește într-un fișier header (sau antet) și este compusă din antetul funcției (Figura 1) urmat de semnul de caracterul **;**. Prototipul unei funcții nu "spune" cum prelucrază funcția informațiile primite, ci indică numai legaăturile sale cu exteriorul.

Exemplu 4.

Prototipurile funcțiilor din Exemplu 3. sunt:

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

2.3. Definiția unei funcții

Definiția unei funcții se va găsi întotdeauna într-un fișier cu cod sursă și este formată din antetul funcției urmat de un bloc în care se găsește algoritmul, exprimat în limbajul de programare **C**, care descrie modul în care funcția realizează prelucrările informației primite din exterior. Blocul începe întotdeauna cu caracterul **{** și se încheie cu **}**.

Variabilele (de orice tip) definite în interiorul unei funcții sunt locale funcției respective și **nu sunt recunoscute în afara funcției**, iar modificările făcute asupra variabile din interiorul funcției, cât și asupra parametrilor de intrare se vor pierde la ieșirea din funcție. De aceea, rezultatul pe care funcția trebuie să-l trimită în exterior trebuie returnat prin folosirea instrucțiunii **return**.

Forma generală a aceste instrucțiuni este:

```
return expresie;
```

Observație: modificările făcute asupra elementelor unui tablou se păstrează și după ieșirea din funcție.

Exemplu 5.

Dacă funcția cu prototipul din Exemplu 4.a). trebuie să calculeze partea întreagă a sumei celor doi reali primiți care parametri de intrare atunci definiția respectivei funcții este:

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

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

Observație: nu putem avea același nume pentru o funcție și pentru o variabilă.

Pentru rezolvarea corectă a problemelor ce urmează vă rog să citiți cu atenție fiecare enunț de la început până la sfârșit.

Rezolvările se fac folosind proiecte și fără variabile globale.

TEMA 1

Problema 1.1.

Să se scrie un program care citește de la tastatură un număr întreg n și două seturi de câte n elemente de tip întreg, fiecare set fiind stocat într-un tablou unidimensional cu maxim 30 de elemente, afișează cele două tablouri, calculează tabloul (vectorul) sumă și afișează vectorul rezultat. Rezolvarea trebuie să conțină funcții scrise pentru: citirea de la tastatură a unui vector, afișarea pe monitor a unui vector, calculul vectorului sumă. (se scrie funcție pentru citirea unui vector, care se apelează în această problemă de două ori).

Observație: problema se bazează pe exemplele de la curs.

Problema 1.2.

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

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

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

și se găsește în MATH.H.

Se vor scrie funcții 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 citește de la tastatură un tablou de maximum 30 de numere întregi, afișează pe monitor vectorul citit, ordonează elementele vectorului crescător prin metoda bulelor și afișează rezultatul pe monitor. Se vor scrie funcții pentru citirea unui vector de numere întregi, scrierea unui vector de numere întregi și ordonare prin metoda bulelor.

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

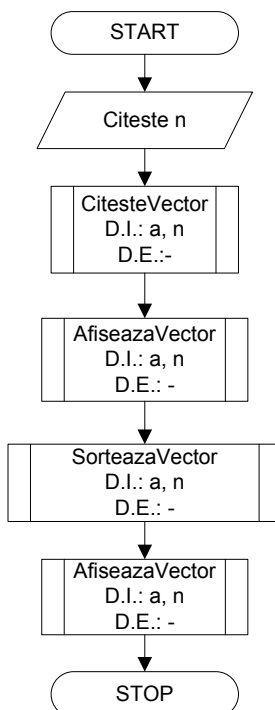


Figura 2. Schema logică a programului de ordonare (sortare) în sens crescător a elementelor unui vector

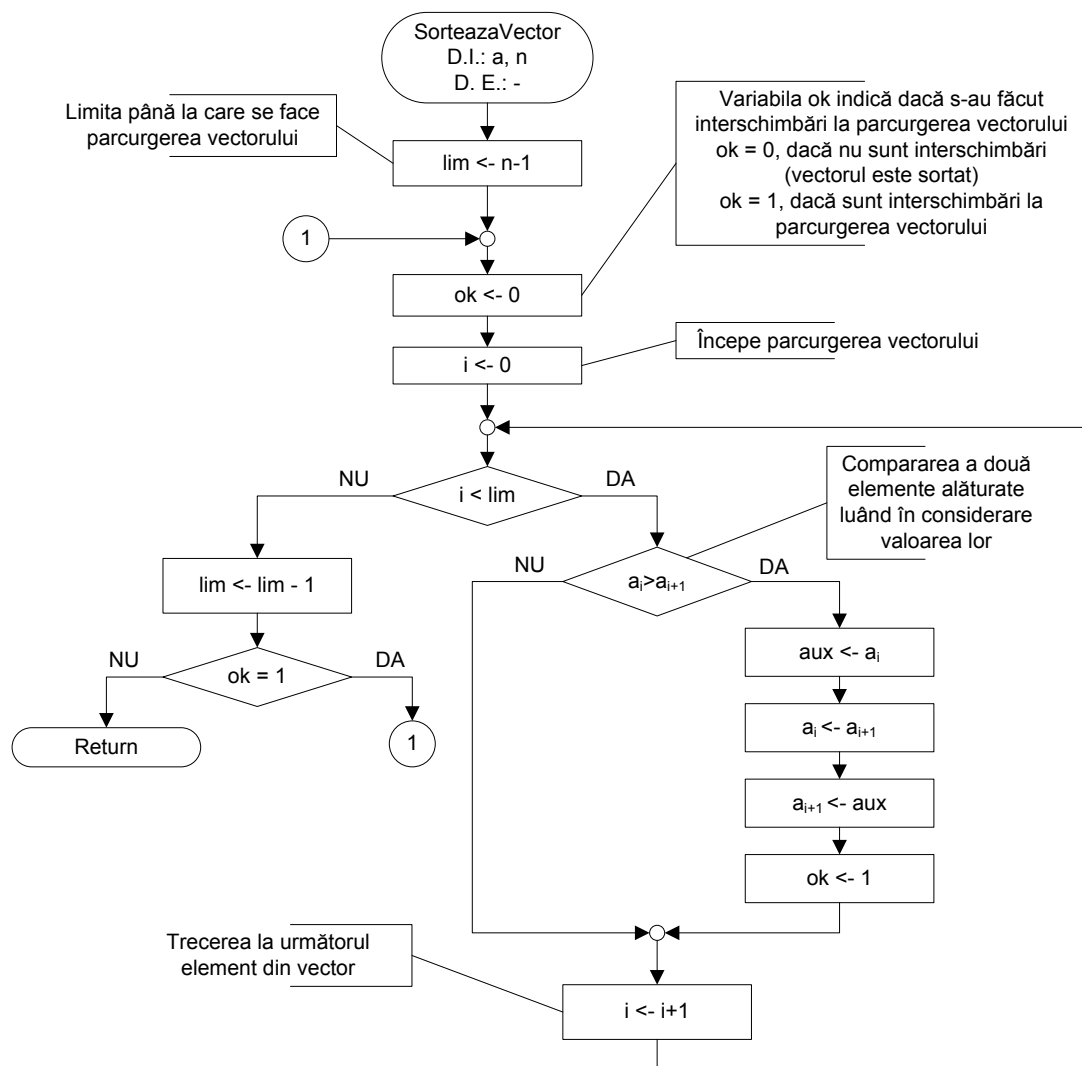


Figura 3. Schema logică a funcție de sortare în sens crescător a elementelor unui vector (folosește sortarea prin metoda bulelor)

TEMA 2

Problema 2.1.

În urma unor măsurători făcute în cadrul unui experiment rezultă un șir 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. afișarea numărului de componente care nu depășesc valoarea medie
- d. crearea unui alt vector **y** cu elementele din **x** mai mari decât valoarea medie și afișarea vectorului **y** cu câte 5 elemente pe o linie.

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

1. citirea unui șir de numere cu prototipul:

```
void citesteRVector(double a[], int n);
```
2. afișarea unui șir de numere cu prototipul:

```
void scrieRVector(double a[], int n);
```
3. fiecare din cele 4 tipuri de prelucrări (câte o funcție pentru fiecare prelucrare).
 Aceste funcții vor returna rezultatul calculat, scrierea rezultatului pe monitori făcându-se în funcția **main**.

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

Prototipurile funcțiilor se vor găsi într-un fișier header după modelul dat la curs.

Problema 2.2.

Se citește de la tastatură un număr natural **n** (**n** cel mult egal cu 100) și un șir de **n** numere întregi.. Șirul de numere se memorează într-un tablou.

Să se afișeze elementele distincte.

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

Se vor scrie funcții pentru:

1. citirea unui șir de numere cu prototipul (de la problema anterioară):

```
void citireIVector(int a[], int n);
```

```
// funcția returnează numărul de elemente efectiv citite
```
2. afișarea unui șir de numere cu prototipul (de la problema anterioară):

```
void afisareIVector(int a[], int n);
```
3. determinarea șirului cu elemente distincte cu prototipul:


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

// funcția are ca parametri de intrare șirul inițial (**a**), șirul final (cu elemente distincte **b**) și numărul de elemente din șirul inițial **n** și returnează numărul de elemente din șirul final.

4. funcție care verifică dacă un element se găsește sau nu într-un vector. Funcția returnează valoarea 1 dacă elementul se găsește în vector și 0 în caz contrar. Prototipul funcției 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 și **el** este elementul care se caută.

Prototipurile funcțiilor se vor găsi într-un fișier 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 concentrația unei anumite componente în proba de laborator analizată. Pentru aceasta se notează pe o fișă concentrația și momentul de timp corespunzător (considerând momentul de început al analizei ca moment zero). Se obțin astfel două șiruri de valori: (i) șirul cu valorile momentelor de timp și (ii) șirul cu valorile concentrațiilor la aceste momente de timp.

Se cere să se scrie un program care citește cele două șiruri de valori (care sunt numere reale) și afișează coeficienții dreptei de regresie sau afișează un mesaj corespunzător dacă aceștia nu pot fi calculați.

Dreapta de regresie $x = a \cdot t + b$ 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.

Coeficienții a și 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 funcții:

- funcție pentru citirea unui vector
- funcție pentru afișarea unui vector

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

c. funcție pentru rezolvarea unui sistem de două ecuații cu două necunoscute.

Această funcție are doi parametri:

- un tablou cu 6 elemente conținând coeficienții celor două ecuații;
- un tablou cu două elemente conținând soluțiile sistemului

și returnează rezultatul:

- 1 – dacă sistemul este compatibil determinat
- 2 – dacă sistemul este compatibil nedeterminat
- 3 – dacă sistemul este incompatibil

d. funcție care calculează produsul scalar a doi vectori a și b . Funcția are ca parametri cei doi vectori și numărul de elemente al unui vector și returnează valoarea produsului scalar. Pentru calculul sumei valorilor elementelor unui vector folosind funcția produs scalar unul din vectori va fi un vector inițializat cu unități.

Observație:

Toate sumele se vor calcula folosind funcția produs scalar.

Prototipurile funcțiilor se vor găsi într-un fișier header după modelul dat la curs.