DECLARAŢII ŞI DEFINIŢII DE VARIABILE, VECTORI ŞI FUNCŢII. CITIRE DIN FIŞIER. SCRIERE ÎN FIŞIER

În limbajul **C** operăm cu entităţi scalare (care se referă la un singur element) şi entităţi agregate (sau structurate) care conţin mai multe entităţi de acelaşi tip sau de tipuri diferite: tablouri şi structuri.

Înainte de a fi folosite într-un program, funcţie sau bloc (un bloc este o secvenţă de cod cuprinsă între **{** şi **}**) orice entitate trebuie declarată sau definită. Prin declaraţie, se face cunoscut programului că va fi folosită o anumită entitate, fără a i se rezerva acesteia vreo zonă de memorie. Prin definiţie, se face cunoscut programului că va fi folosită o anumită entitate şi **se rezervă spaţiu de memorie pentru entitatea respectivă.**

Într-un program, pentru o entitate putem avea mai multe declaraţii, dar o singură definiţie. O definiţie poate fi şi o declaraţie, dar nu şi invers. Declaraţiile entităţilor sunt fi plasate întotdeauna în fişierul header, pe când definiţiile se vor găsi **numai** în corpul funcţiilor. Modul de declarare şi definire al diferitelor entităţi întâlnite într-un program au fost prezentate la curs.

Într-un program avem declaraţii şi/sau definiţii de variabile şi de funcţii.

1. Declaraţia/definiţia unei variabile scalare (simple)

În general pentru o variabilă scalară, declaraţia şi definiţia coincid. La definire se rezervă in memorie un număr de octeţi în care se va păstra valoarea variabilei respective. Numărul de octeţi rezervaţi este dat de tipul variabilei şi poate depinde şi compilator.

Definiţia unei variabile are forma:

T lista_var;

unde **T** este unul din tipurile predefinite ale limbajului (prezentate la curs), iar lista_var este lista de variabile de tipul **T** care vor fi folosite în programul, funcţia sau blocul respectiv Lista de variabile poate fi formată din una sau mai multe variabile, fiecare variabilă fiind specificată prin identificatorul asociat. În cazul în care lista cuprinde mai multe variabile, identificatorii vor fi separaţi prin virgulă. La sfârşitul declaraţiei trebuie să fie prezent caracterul **;** (punct şi virgulă).

Deşi o declaraţie se poate referi la mai multe variabile, se recomandă, gruparea variabilelor de declarat în funcţie de modul de utilizare (date de intrare, date de ieşire, date cu semnificaţii speciale). În anumite cazuri, se impune chiar declararea unei singure

variabile pe o linie pentru a putea introduce comentarii referitoare la utilizarea în cod a variabilei respective.

În limbajul **C** toate declaraţiile de variabile trebuie să apară înaintea oricărei instrucţiuni din program, funcţie sau bloc.

Observaţie: după definire valoarea variabilei este nedeterminată, nefăcându-se o iniţializare implicită.

Exemplu 3.1.

```
a) int i, j;            /* i şi j sunt date întregi cu semn */
b) char c;              /* c este o variabilă de tip caracter */
c) long double x;       /* x este o dată de tip real în precizie extinsă*/
```

Dimensiunea (în octeţi) a unui tip sau a unei variabile se determină utilizând operatorul **sizeof**. Forma generală este:

sizeof(**T**)

sau

sizeof(**var**)

în care **T** este tipul căruia îi determinăm dimensiunea, iar **var** este o variabilă a cărei dimensiune vrem să o determinăm, indiferent dacă aceasta este o variabilă scalară sau agregată.

Exemplu 3.2.

a) Expresia **sizeof(long long int)** are ca rezultat dimensiunea în octeţi a tipului de dată long long int.

b) În cazul în care avem definiţia

long double x;

expresia **sizeof(x)** are ca rezultat dimensiunea în octeţi a variabilei **x** (câţi octeţi au fost rezervaţi pentru memorarea valorilor variabilei **x**).

2. 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 (care coincide cu definiţia) unui tablou unidimensional este de forma:

T nume[dim];

unde :

T este unul din tipurile existente în limbaj sau definite de programator

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 3.3.

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 3.4.

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 dimensiuni, 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.

3. Funcţii

3.1. Antetul unei funcţii

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

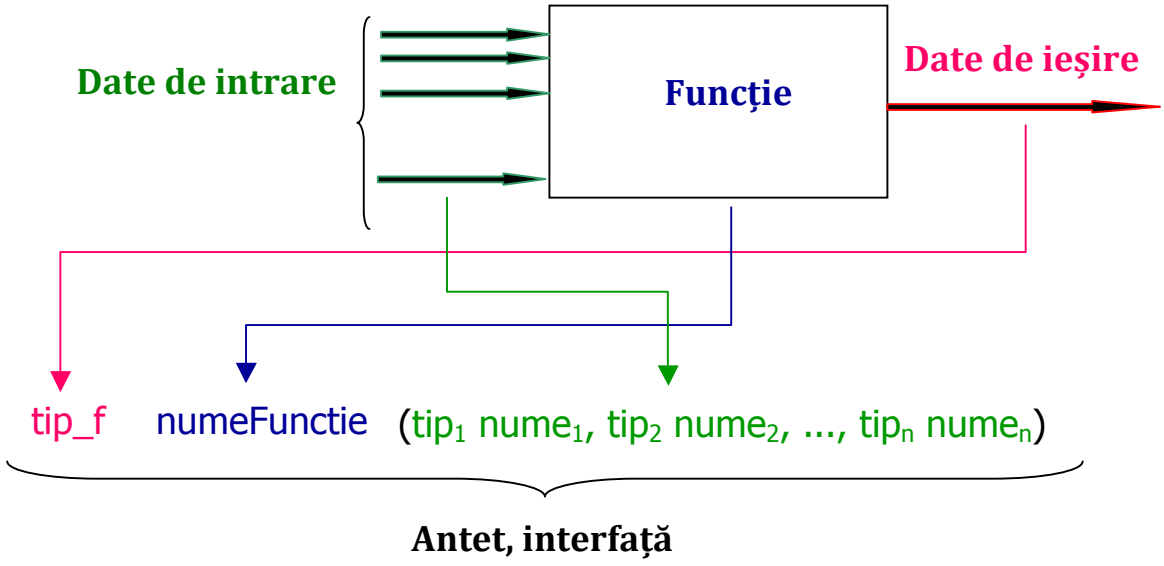


Figura 1. Diagrama asociată unei funcţii

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 indicate 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 3.5.

a) antetul unei funcţii al cărui nume este **suma** şi care primeşte din exterior două informaţii de tip real în dublă precizie 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 cu elemente reale în dublă precizie şi nu furnizează în exterior niciun rezultat:

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

3.2. 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 3.6.

Prototipurile funcţiilor din Exemplu 3.5. sunt:

a) int suma(double a, double b)**;**

b) char lama(void);

c) void citesteVector(int n, double a[]);

3.3. 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 3.7.

Dacă funcţia cu prototipul din Exemplu 3.6.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 a limbajului).

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

4. Operaţii de intrare – ieşire în fişiere

Pentru a putea lucra cu fişiere mai întâi trebuie declarată o variabilă care va fi asociată fişierului cu care lucrăm la deschiderea acestuia.

Declaraţia variabilei respective are forma generală:

```
FILE * numeVariabila;
```

unde **numeVariabila** este numele variabilei care va fi asociată unui fişier la deschiderea acestuia. Vom avea astfel de declaraţii pentru fiecare fişier cu care vrem să lucrăm.

Deoarece pot apare erori la deschiderea unui fişier, trebuie verificat dacă, după deschiderea fişierului, variabila numeVariabila are valoarea egală cu zero. În acest caz fişierul nu s-a putut deschide şi programul trebuie încheiat.

Secvenţa de cod pentru deschiderea unui fişier pentru citire (vrem să citim din el) în cazul în care am avut declaraţia:

```
FILE *in;
```

este

```
in = fopen("intrare.dat","r");
if (in == 0)
{
    fprintf(stderr, "Eroare la deschiderea fişierului intrare.dat.\n");
    exit(EXIT_FAILURE);
}
```

unde **intrare.dat** este numele fişierului din care citim datele.

Secvenţa de cod pentru deschiderea unui fişier pentru scriere (vrem să scriem din el) în cazul în care am avut declaraţia:

```
FILE *out;
```

este

```
out = fopen("iesire.dat","w");
if (out == 0)
{
    fprintf(stderr, "Eroare la deschiderea fişierului iesire.dat.\n");
    exit(EXIT_FAILURE);
}
```

unde **iesire.dat** este numele fişierului în care scriem rezultatele.

După ce am terminat lucrul cu un anumit fişier trebuie să-l închidem. Închiderea fişierului se face folosind funcţia **fclose**, astfel:

```
fclose(in);      /* Inchide fisierul pentru citire */
fclose(out);     /* Inchide fisierul pentru scriere */
```

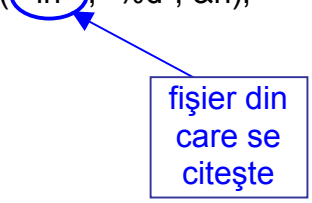
Citirea dintr-un fişier se face prin folosirea funcţiei **fscanf**, iar scrierea într-un fişier se face folosind funcţia **fprintf**.

Cele două funcţii sunt asemănătoare funcţiilor **scanf** şi **printf**, doar că înaintea specificării formatului de citire sau scriere, trebuie să specificăm fişierul din care citim sau în care scriem.

Exemplu 3.8.

a) Vrem să citim din fişierul care are asociată variabila **in** valoarea variabile **n** care este de tip întreg. Instrucţiunea va fi:

```
fscanf ( in , "%d", &n);
```

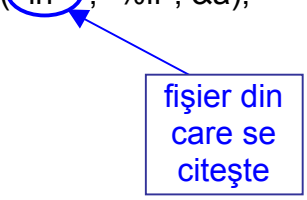


b) Citirea unui real dublu pentru care avem declaraţia

```
double a;
```

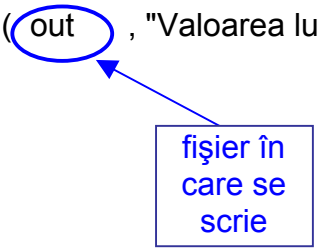
se va face cu instrucţiunea:

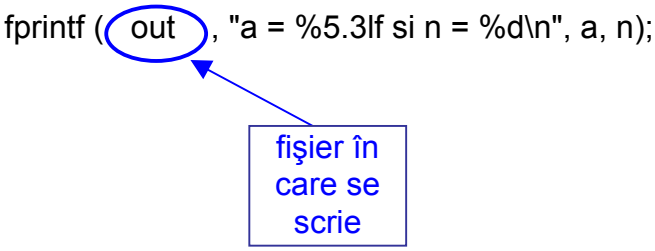
```
fscanf ( in , "%lf", &a);
```



c) Scrierea în fişierul care are asociată variabila **out**:

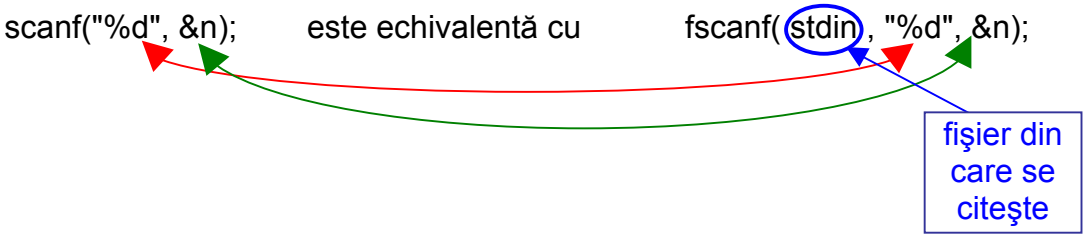
```
fprintf ( out , "Valoarea lui n este %d.|n", n);
```



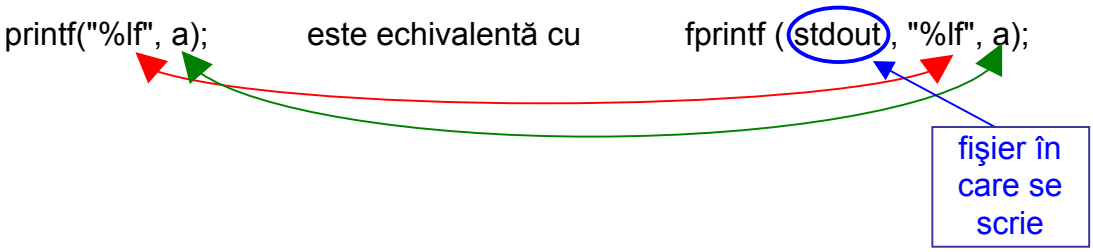


Toate regulile de scriere enunţate pentru funcţiile **printf** şi **scanf** rămân valabile şi pentru **fscanf** şi **fprintf**.

Implicit tastatura (dispozitivul standard de intrare) are asociată variabila **stdin**, iar monitorul (dispozitivul standard de ieşire) are asociată variabila **stdout**. Astfel instrucţiunea:



iar instrucţiunea



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.

Într-o staţie meteorologică se măsoară, în fiecare zi la ora 7, temperatura la 1,5 m deasupra solului. La anumite intervale de timp, măsurate în zile, se realizează rapoarte care cuprind: temperatura minimă, temperatura maximă şi temperatura medie din intervalul de timp considerat, precum şi media geometrică a valorilor absolute ale temperaturilor măsurate în intervalul de timp dat (această valoare va fi folosită apoi în alte prelucrări necesare pentru o analiză meteorologică completă).

Intervalul de timp pentru care se fac prelucrările poate fi de maxim 31 de zile (raportul se face numai pentru zile din aceeaşi lună).

Să se scrie un program care citeşte informaţiile necesare de la tastatură şi generează raportul cerut (ca în exemplul de mai jos) în fişierul **raport.dat**.

Valorile temperaturilor măsurate (se iau în considerare valori întregi pentru temperaturile măsurate) se stochează într-un vector, iar, pentru individualizarea prelucrărilor, în raport se va indica şi luna pentru care se fac prelucrările.

Indicaţie: *Pentru media geometrică se va folosi funcţia pow, iar pentru determinarea valorii absolute se va folosi funcţia abs.*

Se vor scrie funcţii adecvate pentru citirea unui vector, scrierea unui vector, calcularea valorii maxime a elementelor unui vector, calcularea valorii minime a elementelor unui vector, calcularea mediei aritmetice a valorilor elementelor unui vector, calcularea mediei geometrice a valorilor elementelor unui vector.

Exemplu de raport care trebuie generat de program (textul scris cu albastru reprezintă mesajele scrise de program, textul scris cu verde reprezintă valorile datelor de intrare, iar textul scris cu roşu reprezintă valorile rezultatelor calculate de program).

```
Raport de temperatura pentru ultimele 3 zile din luna februarie.
Temperaturile citite sunt: -7, 1, -5.
        Valoarea maxima a temperaturii este: 1.
        Valoarea minima a temperaturii este: -7.
        Valoarea medie a temperaturii este: 3.667.
        Media geometrica a temperaturilor este 3.2711.

Final raport.
```

Atenţie: în acest exemplu culorile sunt folosite doar a evidenţia de unde provin informaţiile citite. Programul nu va folosi culori diferite pentru scrierea raportului şi nici nu va genera un chenar.

Problema 1.2.

Firma "Trioda" comercializează componente electronice şi are două magazine în **Pădurea cu alune**. Cele două magazine se numesc **Trioda1** şi **Trioda2** şi

comercializează aceleaşi sortimente de componente electronice. La sfârşitul fiecărei luni se face un inventar al stocurilor de componente electronice din cele două magazine pentru a determina valoarea stocurilor de componente electronice şi valoarea totală a mărfii. După inventar trebuie generat un raport ca în exemplul următor (convenţiile de culoare sunt cele de la problema 1.1):

```
Raport inventar pentru firma Trioda.

Firma Trioda comercializează 5 tipuri de componente electronice.
Pret / bucata pentru fiecare componenta electronica = (125.3, 3.15, 574, 98.35, 4.25)
        Stocuri magazin Trioda1 = (23, 675, 12, 5, 170)
        Stocuri magazin Trioda2 = (12, 25, 3, 25, 30)
                Total stocuri firma = (35, 700, 15, 30, 200)
        Valoare stocuri magazin Trioda1 = (2881.90, 2126.25, 6888.00, 491.75, 722.50)
        Valoare stocuri magazin Trioda2 = (1503.60, 78.75, 1722.00, 2458.75, 127.50)
                Valoare stocuri firma =  (4385.50, 2205.00, 8610.00, 2950.50, 850.00)
Valoarea totală marfă = 19001.00 lei.

Sfarsit raport inventar.
```

Să se scrie un program care citeşte informaţiile necesare şi

- generează un raport de inventar pentru firma **Trioda** după modelul dat;
- calculează valoarea stocurilor firmei şi valoarea totală a mărfii după o lună de la inventarul iniţial considerând că stocurile au rămas aceleaşi (ceea ce s-a vândut a fost imediat completat), dar toate preţurile au crescut cu o anumită valoare procentuală, valoare care va fi citită de la tastatură (de exemplu, creşterea preţurilor a fost de 0.7%).

Firma **Trioda** poate avea pe stoc la un moment dat maxim 50 de tipuri de componente electronice.

Se vor scrie funcţii adecvate pentru: citirea unui vector, afişarea unui vector (conform modelului dat), înmulţirea a doi vectori. suma a doi vectori, suma valorilor elementelor a doi vectori, înmulţirea unui vector cu un scalar.

Datele de intrare se găsesc în fişierul **trioda.dat**, iar rezultatele (raportul de inventar iniţial şi noile valori ale stocurilor) vor fi afişate pe ecran.

Problema 1.3.

Să se scrie un program care citeşte un tablou de maximum 30 de numere întregi dintr-un fişier numit **inP12.dat**, afişează pe monitor vectorul citit, ordonează crescător, prin metoda bulelor, elementele vectorului 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. (*Observaţie se pot folosi funcţiile scrise în problemele anterioare*).

Schema logică a programului este dată în **Figura 2**, iar schema logică a funcţiei pentru sortarea, în ordine cresc prin metoda bulelor este dată în **Figura 3**.

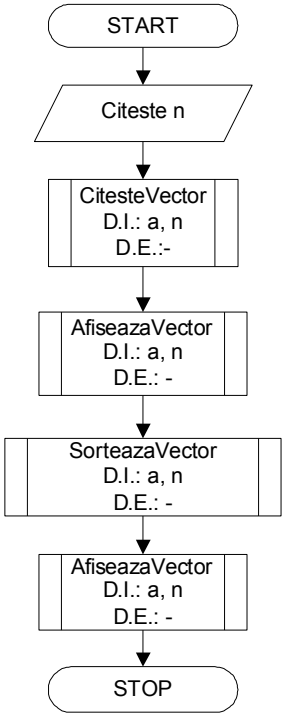


Figura 2. Schema logică a programului de ordonare (sortare) în sens crescător a elementelor unui vector

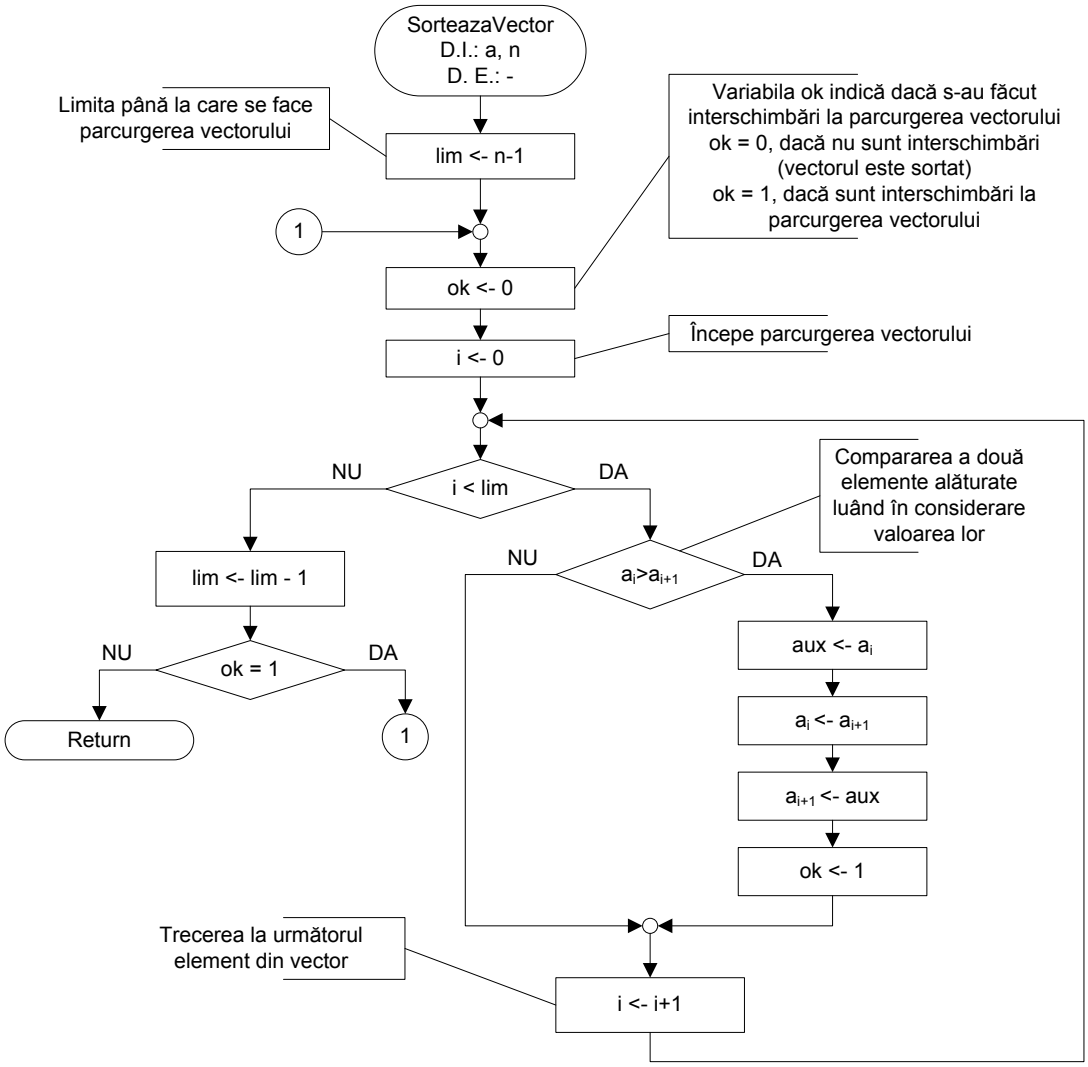


Figura 3. Schema logică a funcţie de sortare în sens crescător a elementelor unui vector (foloseşte sortarea prin metoda bulelor)

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:

   int citesteVector(FILE *in, double a[]);

   // funcţia returnează numărul de elemente efectiv citite
2. afişarea unui şir de numere cu prototipul:

   void scrieVector(FILE *out, 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 în fişier făcându-se din 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.

Datele vor fi citite din fişierul **inP21.dat**, iar rezultatele vor fi scrise în fişierul **rezP21.dat**.

Problema 2.2.

Se citeşte de la tastatură un şir de cel mult 100 numere întregi, până la întâlnirea valorii **0**. 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:

   int citire(FILE * in, int a[]);

   // funcţia returnează numărul de elemente efectiv citite
2. afişarea unui şir de numere cu prototipul (de la problema anterioară):

   void afisare(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.

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.