Laboratorul 7. TABLOURI DE DATE

Introducere

Cantitatea de informație prelucrată de programe poate fi mare, de asemenea tipurile datelor prelucrate pot fi aceleaşi sau diferite. Este dificil ca aceste date să fie vehiculate doar prin intermediul unor variabile simple, deoarece ar fi necesar să se folosească un număr mare de declarații, cu nume diferite. În astfel de situații se pot folosi tablouri (masive) de date care permit ca, prin folosirea unui singur nume, să se stocheze şi prelucreze un mare număr de date. Dimensiunea tablourilor este limitată doar de memoria disponibilă.

Prin folosirea tablourilor de date se construiesc şi şirurile de caractere pentru care în C/C++ nu există tip de date predefinit.

7.1. Declararea tablourilor de date

Tabloul de date (sau masiv de date) este o colecție de date de acelaşi tip, plasate într-o zonă contiguă de memorie (adresele elementelor tabloului sunt succesive).

Tipul de date tablou este **tip de date derivat**. Este construit folosindu-se tipuri de date predefinite sau definite de programator.

Tablourile de date se mai numesc *variabile compuse (structurate)*, deoarece grupează mai multe elemente.

Sintaxa declarației unui tablou cu N dimensiuni este:

tip_element nume_tablou [dim_1][dim_2]…[dim_N];

unde:
- **tip_element** – poate fi orice tip de date, fundamental sau definit de utilizator;
- **nume_tablou** – orice identificator care nu a primit altă semnificație în domeniul declarației;
- **dim_1, dim_2,...,dim_N** – sunt constante numerice întregi.

Zona de memorie rezervată conține

dim_1 x dim_2 x ... x dim_N

elemente de tipul tip_element, adică: dim_1 x dim_2 x ... x dim_N x sizeof(tip_element) octeți.

Variabilele tablou au nume, iar tipul tabloului este dat de tipul elementelor sale. Elementele tabloului pot fi referite prin numele tabloului şi indecşi (valori numerice întregi) care reprezintă poziția elementului în cadrul tabloului.

Exemple de declarare de tablouri de date:

```
float vect1[5];                 // se declară un tablou cu 5 elemente de tip float, zona de
                                // memorie alocată fiind de 5*sizeof(float) bytes
int dimensiuni[4][12];          // se declară un tablou cu 4*12 elemente de tip int, zona
                                // de memorie alocată fiind de 4*12*sizeof(int) bytes
```

Referirea unui element de tablou se face cu operatorul de indexare [ ] sub forma:

nume_tablou [indice_1][indice_2]...[indice_N];

unde: *indice_1, indice_2,...,indice_N* – poate fi orice expresie care returnează o valoare întreagă.

Declarația unui tablou se poate face cu inițializarea sa, folosind sintaxa:

tip_element nume_tablou [indice_1][indice_2]...[indice_N] ={listă_valori};

Lista de valori trebuie să conțină constante de tip compatibil cu tipul de bază al tabloului, în ordinea plasării în memorie.

Exemple de declarare a tablourilor cu inițializare:

```
float vect1[5];                          // se declară un tablou cu 5 elemente
                                         // float, fără inițializare
int vect2[10]={2,7,-1,0,9,15,-5,22,6,11}; // se declară un tablou cu 10 elemente int
                                         // cu inițializare
int mat[2][3]={{3,5,-3},{2,-1,0}};       // se declară un tablou bidimensional cu
                                         // 2*3 elemente de tip întreg, cu inițializare
```

Dacă numărul valorilor din lista de inițializare este mai mic decât dimensiunea tabloului, adică decât numărul de elemente ale tabloului, vor fi inițializate doar primele elemente în ordinea de alocare, celelalte rămânând neinițializate.

```
float vect[5] = { 1.5, 3.2, 7.1};        // se inițializează primele trei valori, două
                                         // rămânând neinițializate
```

Situația în care numărul valorilor din lista de inițializare este mai mare decât dimensiunea tabloului generează un mesaj de eroare.

```
float vect[5] = { 1.5, 3.2, 7.1, -2.5, 6, 10 };  // eroare, numărul valorilor de inițializare este mai
                                                 // mare decât numărul de elemente ale tabloului
```

Este permisă declararea tablourilor fără precizarea dimensiunii, dacă declarația este cu inițializare. În această situație, dimensiunea este stabilită în mod implicit egală cu numărul de valori din listă.

```
float vect[ ]={ 1.5, 3.2, 7.1, -2.5, 6 };  // numărul valorilor de inițializare este 5, deci
                                           // dimensiunea implicită a tabloului va fi 5
```

Indiferent că tabloul a fost declarat cu inițializare sau nu, valorile elementelor de tablou vor putea fi modificate pe parcursul execuției programului, ca în cazul oricărei variabile.

```
float vect[5];                  // se declară un tablou cu 5 elemente float, fără
                                // inițializare
vect[0] = 1.5;                  // elementului de index 0 i se atribuie valoarea 1.5
vect[1] = -2.5;                 // elementului de index 1 i se atribuie valoarea -2.5
vect[5] = 3.2;                  // deşi nu este semnalată eroare, se foloseşte index
                                // incorect indexul, valorile corecte fiind 0...4; această
                                // situație poate genera efecte necontrolate în evoluția
                                // programului
int mat[2][3]={{3,5,-3},{2,-1,0}};  // se declară un tablou bidimensional cu
                                    // 2*3 elemente de tip întreg, cu inițializare
mat[1][2]= 23;                      // elementului de indecşi 1, respectiv 2, i
                                    // se atribuie valoarea 23
```

Observații

- Numele tabloului (neînsoțit de indecşi) reprezintă adresa de memorie la care este alocat tabloul.
- Indecşii elementelor tabloului iau valori începând cu 0 (zero), deci valoarea maximă a lor este dim_n -1.
- După alocarea memoriei necesare, nici la compilare nici la execuție, nu se mai fac verificări de dimensiune, deci nu se verifică dacă indecşii utilizați au valori corecte. Dacă indecşii depăşesc dimensiunea tabloului, există riscul să se înscrie valori în zone de memorie utilizate în alt scop, distrugând date sau cod de program.
- Elementele neinițializate ale tablourilor iau valori, asemeni oricărei variabile, după cum urmează:
  - valoare zero dacă tabloul este declarat global;
  - valoare reziduală, dacă declararea tabloului este locală unui bloc de instrucțiuni.
- La declararea unui tablou, dimensiunile se specifică prin constante numerice întregi literale sau cu nume, cum ar fi:

```
#define dim1 15
const int dim2 = 99;
int vector1[20];
float vector2[dim1];
double vector3[dim2];
```

- Declarația următoare:

```
int dim = 30;
unsigned int vector4[dim];
```

  este semnalată cu mesaj de eroare, deoarece se utilizează o variabilă pentru specificarea dimensiunii tabloului.

Folosirea tablourilor uşurează scrierea programelor care prelucrează mai multe variabile de acelaşi tip, prelucrate în secvențe similare, referirea lor făcându-se cu ajutorul unui nume comun. Aceasta aduce uşurință în scrierea programului, uşurință în citirea şi depanarea acestuia. În caz contrar, ar fi necesară declararea unui număr mare de variabile, cu nume distincte, scrierea programului fiind greoaie.

În funcție de numărul dimensiunilor precizate la declararea tablourilor, tablourile pot fi grupate în **tablouri unidimensionale** (se mai denumesc vectori) şi **tablouri multidimensionale**, dintre care tablourile bidimensionale (matrice) sunt cel mai frecvent folosite.

Am văzut că, limbajele C/C++ nu au definit ca tip de date tipul şir de caractere. În aceste limbaje, şirurile de caractere sunt construite cu ajutorul tablourilor unidimensionale cu elemente de tip char.

7.2. Tablouri unidimensionale

Declarația unui tablou unidimensional se face, conform sintaxei precizată în paragrafului anterior astfel:

tip_element nume_tablou [dim];

sau cu inițializarea elementelor:

tip nume_tablou [dim] = { listă_valori };

Pentru tabloul de date cu numele **nume_tablou** se alocă **dim** elemente de tipul **tip_element** plasate succesiv într-o zonă contiguă de memorie.

Pentru referirea unui element se foloseşte sintaxa următoare:

nume_tablou[index]

unde **index** precizează poziția elementului în tablou. Acesta este reprezentat prin orice expresie întreagă pozitivă.

Exemplu de declarare a unui tablou cu 6 elemente de tip int cu inițializare:

```
int vector[6] = { 3, 5, -7, 15, 99, 43};
```

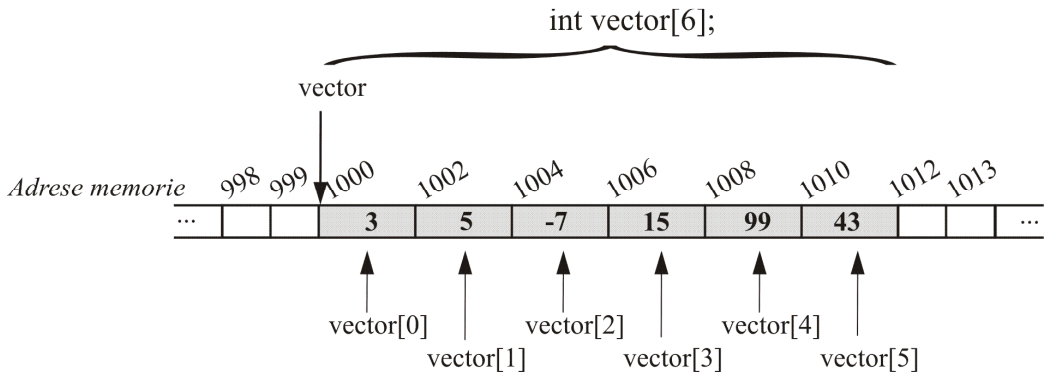

Fig. 7.1. Exemplu de alocare în memorie a unui tablou unidimensional

Alocarea tabloului în memorie se face ca în Fig. 7.1.

Numele tabloului, vector, neînsoțit de index, reprezintă adresa la care s-a făcut alocarea de memorie. Elementele tabloului sunt alocate la adrese de memorie succesive.

Se poate determina spațiul de memorie alocat pentru tablou cu operatorul sizeof.

```
printf("\nTabloul vector ocupa %d octeti in memorie", sizeof(vector));
```

Adresa de alocare este determinată prin numele tabloului. Se poate folosi şi operatorul &, dar expresiile vector şi respectiv &vector sunt echivalente. Pentru afişarea valorii se poate folosi formatul specific adreselor, %p, formatul fiind hexazecimal, sau cu formatul %u, ca întreg zecimal fără semn.

```
printf("\nTabloul vector este alocat la adresa de memorie %p", vector);
printf("\nTabloul vector este alocat la adresa de memorie %u", vector);
```

Referirea elementelor tabloului se face prin numele tabloului, operatorul [ ] şi valoarea indexului. Acesta indică poziția elementului în tablou. Un element al tabloului se poate folosi ca orice variabilă simplă de acelaşi tip.

Luând în considerare declararea tabloului vector, se pot folosii expresii care includ elemente ale tabloului ca în exemplul următor:

```
int a;
int vector[6] = { 3, 5, -7, 15, 99, 43};

vector[0] = 10;
a = (vector[1] +vector[2]) / 2;
printf("\nValoarea adresei &vector[0] = %u", &vector[0]);          // se afişează adresa la care e
                                                                   // alocat elementul de index 0
printf("\nElementul vector[0] ocupa %d octeti", sizeof(vector[0])); // se afişează nr. de
                                                                   // octeţi folosiţi pentru
                                                                   // reprezentarea elementului
 printf("\nValoarea elementului vector[0] = %d", vector[0]);      // se afişează valoarea
                                                                   // elementului
```

Observații

- Numerotarea indecşilor în C/C++ începe totdeauna cu valoarea 0 (zero) !
- Compilatorul nu verifică corectitudinea indecşilor folosiți !

În mod frecvent, dacă se doreşte prelucrarea mai multor elemente, sau chiar a tuturor elementelor tabloului, se folosesc instrucțiunile repetitive (for, while, do-while).

```
int i;
for ( i=0 ; i<5 ; i++)
     printf("\nValoarea elementului vector[%d] = %d", i, vector[i]);
```

În secvența anterioară, variabila i este folosită pentru desemnarea indexului elementelor. Prin valorile succesive de la 0 la 4 ale indicelui, se afişează pe rând valorile elementelor vector[0]... vector[4].

Observații

- Operatorii specifici limbajului pot fi aplicați elementelor tabloului în măsura în care tipul elementelor permite acest lucru.
- Nu se pot face operații cu tablouri în ansamblul lor, ci doar cu elemente ale acestora.

```
int tab1[5];
float tab[10];
// …
tab1 = tab2 ;    // eroare, nu se poate face atribuirea; tab1 şi tab2 sunt adresele
                 // la care sunt alocate tablourile şi ele nu pot fi modificate

tab1 = 100 ;     // eroare, ca si in cazul precedent, nu se poate face atribuirea,
                 // adică nu se poate modifica adresa la care este alocat tabloul

tab1+tab2 ;      // eroare, operatorul + nu este definit pentru a opera cu tablouri

tab1[0]+tab2[0];         // corect, tab1[0] şi tab2[0] sunt date de tip int, respectiv
                         // float, deci operația de adunare se poate efectua
```

7.3. Şiruri de caractere

Tablourile unidimensionale cu elemente de tip char sunt folosite pentru memorarea şirurilor de caractere. Pentru a marca sfârşitul unui şir de caractere, după ultimul caracter se adaugă un octet cu valoarea 0 (‘\0’), numit şi terminator de şir.

Declararea unui şir de caractere se face, conform sintaxei specifice tablourilor unidimensionale, prin tipul elementelor (char), numele tabloului şi dimensiune.

```
char nume[6];
```

Declarația poate fi făcută cu inițializare ca în exemplul următor:

```
char nume[6] = { ,’S’, ’M’, ’I’, ’T’, ’H’, ’\0’};
```

Tabloul este alocat în memorie ca în Fig. 7.2.

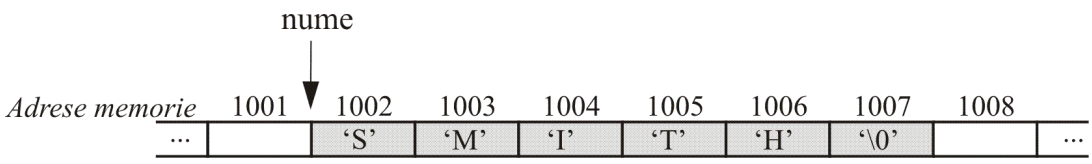

Fig.7.2. Exemplu de alocare în memorie a unui şir de caractere

Inițializarea se poate face şi folosind şiruri de caractere constante.

```
char nume[6] =”Smith”;
```

O listă de caractere încadrată între ghilimele reprezintă o constantă şir de caractere (de exemplu ”SMITH”). Pentru o astfel de constantă, compilatorul rezervă numărul de octeți pe care îi inițializează cu codurile ASCII corespunzătoare caracterelor şi un octet pe care îl inițializează cu valoarea 0 (adică terminatorul de şir ‘\0’). O constantă şir de caractere poate fi utilizată şi pentru inițializarea unei variabile şir de caractere.

Exemple de declare de tablouri:

```
char sir1[10];                  //  se declară un tablou unidimensional, cu
                                // elemente char (şir de caractere), fără inițializare
char sir3[ ]=”sir de caractere”; // se declară un tablou cu 17 elemente char (şir de
                                // caractere) cu inițializare (ultimul caracter depus
                                // în tablou este terminatorul de şir ‘\0’
sir3[0]=’S’;                    // primul caracter al şirului primeşte valoarea ‘S’
```

Observații

- În cazul unui şir vid, primul element al tabloului este chiar terminatorul de şir, de exemplu:

  nume[0]=’\0’;

- Dimensiunea precizată la declararea unui tablou şir de caractere trebuie să fie mai

mare cu o unitate decât numărul caracterelor semnificative care pot intra în alcătuirea conținutului șirului pentru a permite memorarea terminatorului.

- Terminatorul de șir permite testarea facilă a sfârșitului șirului. Funcțiile definite pentru a opera cu șiruri de caractere nu fac referire la dimensiunea acestora, ci folosesc terminatorul de șir pentru a indica sfârșitul șirurilor.

Șirurile de caractere pot fi tratate, ca orice tablou unidimensional, făcând prelucrări element cu element, deci caracter cu caracter. În plus, există definite în fișierele header standard o mulțime de funcții definite pentru prelucrarea șirurilor de caractere.

De exemplu, se definește o secvență în care se citește de la tastatură conținutul unui șir de caractere. Citirea se face caracter cu caracter până la apăsarea tastei Enter.

```
char sir [20];
int i=0;                                // variabila i se foloseşte pentru desemnarea indexului
                                        // elementului de tablou; primul element are indexul 0
do                                      // secvența repetitiva prin care se citesc succesiv caracterele
   {    fflush(stdin);
        sir[i]=getche();
        i++;
   }
while (sir[i-1]!=13);           // încheierea ciclului do-while se face la apăsarea tastei Enter care
                                // transmite două caractere cu valoarea în codul ASCII 13(trecere la
                                // rând nou '\n') şi respectiv 10 (carriage return '\r'); prima valoare
                                // este preluată de un element de tablou
sir[i-1]='\0';          // după finalizarea introducerii de caractere, se memorează terminatorul de şir
```

Secvența anterioară poate fi înlocuită cu apelul funcțiilor specifice gets() sau scanf():

```
char sir [20];
gets(sir);
```

sau

```
char sir [20];
scanf("%s", sir);
```

Pentru operații de intrare/ieșire de la consolă se pot folosi funcțiile scanf, respectiv printf, cu specificatorul de format %s sau funcțiile gets(), respectiv puts() definite tot în fișierul header stdio.h.

Exemple de folosire a funcțiilor de intrare/ieșire cu șiruri de caractere :

```
char nume[30];
scanf("%s", nume);                  // numele unui tablou este o adresă, astfel că în funcția scanf
                                    // nu a fost necesar să se utilizeze operatorul &
 printf("Numele este: %s", nume);                  // funcția printf, pentru specificatorul %s
                                                   // nu afiseaza adresa la care se află alocat
                                                   // şirul, ci afişează conținutul acestuia
printf("Adresa sirului este: %p", nume);           // funcția printf, pentru specificatorul %p
                                                   // afiseaza adresa la care se află alocat şirul
```

```
 char sir1[10];
scanf("%9s", sir1);          // în specificatorul de format s-a precizat numărul maxim de caractere
                             // care pot fi introduse pentru sir1; cel de al  10-lea octet e rezervat
                             // pentru terminatorul de şir
printf("Sirul citit de la tastatura este: %s", sir1);
```

```
gets(nume);
puts(nume);
```

Deoarece şirurile de caractere sunt construite ca tablouri unidimensionale, operatorii definiţi în mod standard nu pot opera cu acestea, ci se recurge la utilizarea de funcţii definite în fişierele disponibile existente în biblioteca de funcţii. Un fişier cu funcţii destinate lucrului cu şiruri de caractere este fişierul string.h.

Câteva funcţii uzuale definite în fişierul string.h sunt:

strcpy(sir_destinatie, sir_sursa); - copiază conţinutul şirului sursă în şirul destinaţie

strcat(sir_destinatie, sir_sursa); - adaugă conţinutul şirului sursă la şirul destinaţie (concatenează cele două şiruri)

strlen(sir); - returnează numărul de elemente ale şirului (lungimea şirului)

strcmp(sir1, sir2); - compară două şiruri, returnând valoarea 0 dacă şirurile sunt egale, şi valoare diferită de 0 dacă sunt diferite (valoarea e dată de diferenţa dintre primele caractere diferite)

strupr(sir); - transformă literele mici ale şirului în majuscule

strlwr(sir); - transformă majusculele în litere mici

Exemple de folosire a funcţiilor destinate sirurilor de caractere:

```
//declararea sirurilor de caractere
        char sir1[10], sir2[10], sir3[20];
        int i, l1, l2;

//citirea de la tastatura a sirurilor
        scanf("%9s",sir1);
        scanf("%9s",sir2);

//se determina lungimea sir1
        l1=strlen(sir1);

//copierea in sir3 a conţinutului sir1
        strcpy(sir3, sir1);

//se transformă şirul în şir cu majuscule
        strupr(sir1)
```

7.4. Tablouri multidimensionale

Tablourile multidimensionale se declară, aşa cum am văzut în paragraful 7.1, cu sintaxa:

tip_element nume_tablou [dim_1][dim_2]…[dim_N];

Zona de memorie rezervată pentru tablou conține

dim_1 x dim_2 x ... x dim_N

elemente de tipul tip_element, adică:

dim_1 x dim_2 x ... x dim_N x sizeof(tip_element)

octeți.

Un tablou multidimensional este interpretat ca fiind un tablou unidimensional, fiecare element fiind la rândul său un tablou.

În mod frecvent se folosesc tablouri cu două dimensiuni, numite în mod uzual matrice. Matricele sunt interpretate ca tablouri unidimensionale care au elemente formate la rândul lor din tablouri unidimensionale.

Se interpretează că fiecare linie a matricei reprezintă un element al tabloului, la rândul lor liniile fiind formate din tablouri unidimensionale, cu dimensiunea dată de numărul de coloane.

Declarația matricelor se poate face cu sau fără inițializare.

Exemplu de declarare a unui tablou bidimensional, cu inițializare:

```
int matrix[3][4] = {    { 10,  20,  30,  40},
                        { 50,  60,  70,  80},
                        { 90, 100, 110, 120}};
```

Declarația este făcută cu inițializarea elementelor. Alocarea de memorie se face ca în Fig. 7.3.

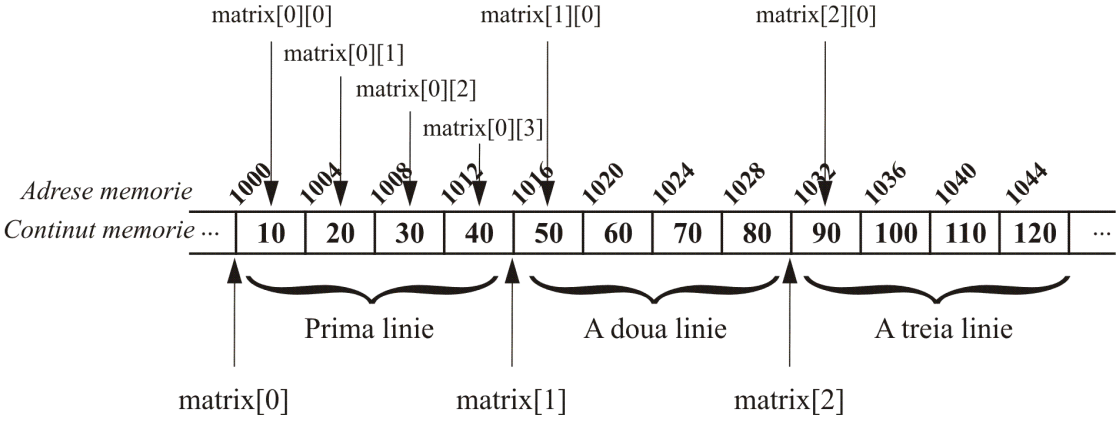

Fig.7.3. Exemplu de reprezentare a unei matrice

Matricea matrix este un tablou unidimensional cu 3 elemente, fiecare element fiind un tablou format din 4 întregi.

Numele matricei, matrix, reprezintă adresa la care este alocată matricea, adresă care coincide cu adresa primei linii a matricei, matrix[0], şi cu adresa primului element de pe prima linie, &matrix[0][0].

Numele matricei însoțit de un index reprezintă o adresă de memorie (matrix[0], matrix[1], matrix[2]), şi anume adresa la care este alocată linia corespunzătoare. O linie a matricei este formată dintr-un tablou unidimensional de 4 întregi.

Numele matricei însoțit de doi indecşi reprezintă un element al matricei şi este un int (matrix[0][0], matrix[0][1], etc.). Un element al matricei poate fi utilizat asemeni oricărei variabile de tip int, adică poate fi utilizat în orice expresie validă pentru acest tip de date.

Indecşii pot fi specificați prin orice expresie care returnează o valoare întreagă.

Dacă se doreşte prelucrarea tuturor elementelor unei matrice, în mod uzual se folosesc instrucțiuni repetitive imbricate (de regulă cicluri for). Se obține un cod compact şi uşor de citit şi depanat.

Se exemplifică folosirea elementelor unei matrice; pentru a referi elementele matricei, se folosesc doi indecşi desemnați de două variabile:

```
  int mat[3][4];          // declararea matricei
  int i, j;
// se citesc de la tastatura elementele lui mat
// variabila i se utilizează pentru desemnarea liniei matricei
// variabila j se utilizează pentru desemnarea coloanei matricei

  for(i=0 ; i<3 ; i++)
    for(j=0 ; j<4 ; j++)
      {
          printf("mat1[%d][%d]=", i, j);
          scanf("%d",&mat1[i][j]);
      }

//se afiseaza elementele matricei mat
  for(i=0 ; i<3 ; i++)
    { printf("\n");
      for(j=0 ; j<4 ; j++)
          printf("%6d", mat [i][j]);
    }
```

Exemple

```
/*************************************************************************
```
Exemplu1 7.1. - Să se întocmească un program prin care se exemplifică utilizarea tablourilor unidimensionale
```
*************************************************************************/
#include <stdio.h>
#include <conio.h>

void main()
{ //declarație tablou de dimensiune 5, fără inițializare
        int tab1[5];
 //declarație tablou de dimensiune 7 cu inițializarea elementelor
        int tab2[7] = {9, 11, 25, 7, 44, 2,14};
        int i;
```

```
 //se afişează valorile elementelor tab1 – valori reziduale
        printf("afisare elemente tab1-contine valori reziduale");
        for(i=0 ; i<5 ; i++)
                printf("\ntab1[%d]=%d", i, tab1[i]);
  //se afişează valorile elementelor tab2 – valorile folosite la initializare
        printf("\n\nafisare elemente tab2-contine valorile de initializare");
        for(i=0 ; i<7 ; i++)
                printf("\ntab2[%d] = %d", i, tab2[i]);
// se citesc de la tastatura valorile elementelor tab1
        printf("\n\ncitire de valori de la tastatura pentru tab1");
        for(i=0 ; i<5 ; i++)
        {       printf("\ntab1[%d]=", i);
                scanf("%d", &tab1[i]);
        }
//se afişează valorile elementelor tab1 – valorile sunt cele citite de la tastatura
        printf("\n\nafisare elemente tab1-contine valorile citite de la tastatura");
        for(i=0 ; i<5 ; i++)
                printf("\ntab1[%d] = %d", i, tab1[i]);
//se afişează valorile elementelor tab2 divizibile cu 3
        printf("\n\nafisarea elementelor tab2-divizibile cu 3");
        for(i=0 ; i<7 ; i++)
                if (!(tab2[i]%3))
                        printf("\ntab2[%d] = %d", i, tab2[i]);
}

/*************************************************************************
```

Exemplu1 7.2. - Să se întocmească un program prin care se exemplifică utilizarea şirurilor de caractere (tablouri unidimensionale cu elemente char). Se exemplifică atât prelucrarea element cu element, cât şi utilizarea funcţiilor specifice.

```
*************************************************************************/
#include<stdio.h>
#include<conio.h>
#include<string.h>

void main()
{
//declararea sirurilor de caractere
        char sir1[10], sir2[10], sir3[20];
        int i, l1, l2;

//citirea de la tastatura a sirurilor
        fflush(stdin);
        printf("\nNumele:");
        scanf("%9s",sir1);
        fflush(stdin);
        printf("\nPrenumele:");
        scanf("%9s",sir2);

/*************************************************************
Exemple de copiere a sirurilor caracter cu caracter
*************************************************************/
//se determina lungimea sir1
        for( l1=0 ; sir1[l1]!=0 ; l1++);
        printf("\nsir1 este format din %d caractere", l1);
```

```
//se determina lungimea sir2
        for( l2=0; sir1[l2]!=0; l2++);
        printf("\nsir2 este format din %d caractere", l2);
//preluarea in sir3 a conținutului sir1
        for(i=0;i<l1;i++)
          sir3[i]=sir1[i];
//se adaugă caracterul spațiu in sir3
        sir3[l1]='  ';
//se adaugă conținutul lui sir2 in sir3
        for(i=0 ; i<l2+1 ; i++)
                sir3[l1+1+i]=sir2[i];
//se adaugă terminatorul de sir
        sir3[l1+l2+1]='\0';
        printf("\nNumele complet este: %s", sir3);
        getch();

/*************************************************************************
Exemple de folosire a funcțiilor destinate sirurilor de caractere
*************************************************************************/
//citirea de la tastatura a sirurilor
        fflush(stdin);
        printf("\nNumele:");
        scanf("%9s",sir1);
        fflush(stdin);
        printf("\nPrenumele:");
        scanf("%9s",sir2);
//se determina lungimea sir1
        l1=strlen(sir1);
        printf("\nsir1 este format din %d caractere", l1);
//se determina lungimea sir2
        l2=strlen(sir2);
        printf("\nsir2 este format din %d caractere", l2);
//copierea in sir3 a conținutului sir1
        strcpy(sir3, sir1);
//se adaugă caracterul spațiu in sir3
        strcat(sir3, " ");
//se adaugă conținutul lui sir2 in sir3
        strcat(sir3, sir2);
        printf("\nNumele complet este: %s", sir3);
        getch();
}


/*************************************************************************
Exemplu1 7.3. - Să se întocmească un program prin care se exemplifică utilizarea tablourilor
                bidimensionale (matrice)
*************************************************************************/
#include<stdio.h>
#include <conio.h>
void main()
{// Declararea matricelor
        int mat1[2][3], mat2[3][4],  i;
        printf("Informatii despre mat1:\n");
//se afiseaza nr. de octeti alocati lui mat1
        printf("\nmat1 ocupa : %d octeti", sizeof(mat1));
```

```
//se afiseaza nr. de octeti alocati unei linii a lui mat1
        printf("\no linie a lui mat1 ocupa : %d octeti", sizeof(mat1[0]));
//se afiseaza nr. de octeti alocati unui element al lui mat1
        printf("\nun element al lui mat1 ocupa : %d octeti", sizeof(mat1[0][0]));
//se afiseaza adresele la care e alocata mat1
        printf("\n&mat1=%u", mat1);          //expresiile mat1 si &mat1 sunt echivalente
//se afiseaza adresele la care se afla liniile lui mat1
        for(i=0 ; i<2 ; i++)
        printf("\nlinia %d se afla la adresa : %u", i, mat1[i]);          // mat1[i] si &mat1[i] sunt
                                                                         // echivalente
        printf("\n\nInformatii despre mat2:\n");
//se afiseaza nr. de octeti alocati lui mat2
        printf("\nmat2 ocupa : %d octeti",sizeof(mat2));
//se afiseaza nr. de octeti alocati unei linii a lui mat2
        printf("\no linie a lui mat2 ocupa : %d octeti",sizeof(mat2[0]));
//se afiseaza nr. de octeti alocati unui element al lui mat2
        printf("\nun element al lui mat2 ocupa : %d octeti", sizeof(mat2[0][0]));
//se afiseaza adresele la care e alocata mat1
        printf("\n&mat2=%u", mat2);                              //mat2 si &mat2 sunt echivalente
//se afiseaza adresele la care se afla liniile lui mat2
        for(i=0 ; i<3 ; i++)
          printf("\nlinia %d se afla la adresa : %u", i, mat2[i]);        // mat2[i] si &mat2[i]
                                                                         // sunt echivalente
}


/*************************************************************************
```

Exemplu1 7.4. - Să se întocmească un program în care:

Se declară 3 matrice de dimensiuni egale. Se citesc de la tastatură două dintre matrice şi se calculează cea de a treia ca sumă a lor. Să se calculeze, pentru o matrice, suma elementelor pe fiecare linie si a tuturor elementelor matricei.

```
*************************************************************************/
#include<stdio.h>
#include <conio.h>

void main()
{ int mat1[3][4], mat2[3][4], mat3[3][4];              // declararea matricelor
  int  i, j;
  int suma_linie, suma_matrice;

// se citesc de la tastatura elementele lui mat1
// variabila i se utilizeaza pentru desemnarea liniei matricei
// variabila j se utilizeaza pentru desemnarea coloanei matricei

  printf("Introduceti elementele matricei mat1:\n");
  for(i=0;i<3;i++)
    for(j=0;j<4;j++)
      {   printf("mat1[%d][%d]=",i,j);
          scanf("%d", &mat1[i][j]);
      }
//se citesc de la tastatura elementele lui mat2
  printf("\nIntroduceti elementele matricei mat2:\n");
  for(i=0;i<3;i++)
    for(j=0;j<4;j++)
```

```
    {   printf("mat2[%d][%d]=", i, j);
        scanf("%d",&mat2[i][j]);
    }
//se calculeaza matricea mat3 ca suma dintre mat1 si mat2
 for(i=0;i<3;i++)
   for(j=0;j<4;j++)
     mat3[i][j]=mat1[i][j] + mat2[i][j];
//se afiseaza matricea mat3 obținuta
printf("\nmat3:");
  for(i=0;i<3;i++)
    { printf("\n");
     for(j=0;j<4;j++)
         printf("%6d", mat3[i][j]);
    }
//se calculeaza pentru mat3 pe fiecare linie si suma tuturor elementelor
  suma_matrice=0;
  for(i=0;i<3;i++)
    {suma_linie=0;
     for(j=0;j<4;j++)
         {suma_linie+=mat3[i][j];
          suma_matrice+=mat3[i][j];
         }
     printf("\npentru linia %d suma elementelor este: %d", i, suma_linie);
    }
   printf("\nsuma tuturor elementelor matricei mat3 este: %d", suma_matrice);
  getch();
}
```

Întrebări. Exerciții. Probleme.

1. Se citesc de la tastatură elementele unui tablou unidimensional. Să se înlocuiască toate valorile negative cu valoarea 0. Se afişează elementele tabloului.

2. Se citesc de la tastatură elementele unui tablou unidimensional. Să se afişeze, în ordine crescătoare, valorile introduse şi numărul lor de apariţii.

3. Se citesc de la tastatură elementele a două tablouri unidimensionale. Se are în vedere ca o valoare să nu se regăsească de mai multe ori într-un tablou. Să se afişeze valorile care se regăsesc în ambele tablouri şi indecşii corespunzători acestora.

4. Se citeşte de la tastatură un şir de caractere format numai din cifre. Să se convertească în valoarea numerică întreagă corespunzătoare.

5. Să se întocmească un program în care se citeşte de la tastatură un întreg în baza 10. Se afişează numărul reprezentat în baza 2.
   Nota:
   Cifrele binare obținute se vor memora în elementele unui tablou de întregi.

6. Sa se scrie un program in care se declara o matrice cu 4 x 5 reali. Se citesc de la tastatura valori pentru elementele matricei şi se afişează. Se determină şi se afişează maximele de pe fiecare linie a matricei care se memorează într-un tablou unidimensional (dimensiunea=nr. de linii ale matricei) şi maximul dintre toate elementele matricei.