Laborator 9 – Tablouri

Un tablou în C poate fi definit ca nu număr de locaţii de memorie, fiecare putând memora același tip de date și care pot fi referite prin același nume de variabilă.

Un tablou este un nume colectiv dat unui grup similar de cantităţi. Ca exemplu, aceste cantităţi pot fi notele unor studenţi, numărul de scaune dintr-o casă, numele angajaţilor unei companii, etc. Aceste elemente pot fi de tipul întreg, float (double), caracter, etc. Regula generală este că toate elementele unui șir de caractere sunt de același tip (de exemplu nu poate exista un tablou cu 10 numere, 5 întregi și 5 de tip float).

Tablouri multidimensionale

Tablourile trebuie declarate înainte de a fi utilizate. Forma generală de declaraţie a unui tablou este:

tip nume_tablou[dimens1][dimens2]…[dimensn];

Exemple

double height[10];
float width[20];
int min[9];
char nume[20];

tip reprezintă un tip de date recunoscut în C;
dimensi este o expresie constantă şi reprezintă limita superioară a indicelui nr. i al tabloului: indicele i ia valori de la 0 la dimensi-1.
Limita exactă este determinată de compilatorul folosit.

În limbajul C numele unui tablou are ca valoare adresa primului său element. La întâlnirea unei declaraţii de tablou se alocă de către compilator o zonă de memorie necesară pentru a păstra valorile elementelor sale. Numele tabloului respectiv poate fi utilizat în diferite expresii şi valoarea lui este adresa de început a zonei de memorie care i-a fost alocată.

Cantitatea de memorie necesară pentru stocarea unui tablou depinde de tipul şi dimensiunea acestuia. Cantitatea de memorie necesară pentru stocarea unui tablou se calculează astfel:

Total octeţi= sizeof(tip)* dimens1*dimens2*… dimensn

Exemple

1. int vect[10][15];

Declaraţia definește tabloul vect de 10*15=150 elemente și are tipul int. Pentru acest tablou se alocă 10*15*4=600 de octeţi. Vect este un simbol a cărui valoare este adresa primului său element, adică adresa lui vect[0][0]. Deci vect[0][0] are ca valoare adresa primului element al tabloului, iar vect are ca valoare adresa acestui element.

2. char tab[50];

Tabloul tab este un tablou unidimensional de tip char, care are 50 de elemente. I se alocă 50 de octeţi și tab are ca valoare adresa adresa elementului tab[0].

3. double dmat[10][50];

Tabloul dmat este un tablou bidimensional de tip double. El reprezintă o matrice de 10 linii a 50 de coloane fiecare. Compilatorul rezervă pentru acest tablou 10*50*8=4000 de octeţi.

Tablouri unidimensionale

Forma generală de declaraţie a unui tablou unidimensional este:

tip nume_tablou[dimens];

Exemple

1. int vector[10];
2. char s[15];

În exemplele de mai sus, efectul declaraţiilor este următorul:
1. Se declară un tablou unidimensional vector de 10 elemente, toate elementele fiind de tipul int.
2. Se declară un tablou de caractere s cu 15 elemente, de la s[0] la s[14].
Elementele tabloului sunt stocate în locaţii învecinate, într-o ordine precizată prin index.

De multe ori, numărul elementelor unui tablou este specificat printr-o constantă simbolică introdusă cu ajutorul directivei #define a preprocesorului. Această metodă are avantajul că programul sursă este mai uşor de citit şi întreţinut, iar modificarea dimensiunii tabloului afectează un singur punct din programul sursă.

Observaţie:
Limbajul C nu are posibilitatea de verificare a depăşirii limitelor pentru tablouri. Programatorul are datoria de a asigura verificarea depăşirii limitelor acolo unde este nevoie.

Şirurile de caractere sunt cele mai des utilizate tablouri unidimensionale. Prin definiţie, un şir în limbajul C este un tablou de caractere care se încheie printr-un caracter nul (zero), specificat prin ‘\0’.

Astfel pentru un şir care poate memora 15 caractere trebuie făcută următoarea declaraţie:

char s[16];

Se va aloca spaţiu pentru caracterul nul de la sfârşitul tabloului.

În limbajul C deşi nu există un tip de dată de tip şir, este permisă existenţa constantelor şir. O constantă şir de caractere este tratată de compilator ca şi cum ar fi numele unui tablou de caractere iniţializat cu respectiva constantă. Pentru un şir conţinând n caractere, compilatorul rezervă n+1 locaţii. În primele n sunt depuse caracterele din şir, iar ultima conţine terminatorul ‘\0’.

Tablouri bidimensionale

Forma cea mai simplă a unui tablou multidimensional este tabloul bidimensional. Un tablou bidimensional este în esenţă un tablou de tablouri unidimensionale. Forma generală de declarare a unui tablou bidimensional este:

tip nume_tablou[dimens1][dimens2];

Exemple
double matrice[10][30];

Se declară un tablou bidimensional matrice de tip double. El reprezintă o matrice cu 10 linii a 30 coloane fiecare. Memoria utilizată este de 8*10*30=2400 octeţi. În acest caz matrice este o matrice dreptunghiulară fiindcă numărul liniilor este diferit de numărul coloanelor.

char chmat[4][3];

Se declară un tablou bidimensional chmat de tip char.
În declaraţia tabloului parantezele pătrate se asociază de la stânga la dreapta.
Tabloul este stocat într-o matrice al cărei prim indice reprezintă rândul şi al doilea reprezintă coloana.

Matrice pătratice

În tabelul 1 sunt prezentate relaţiile care se pot scrie între indicii i şi j în funcţie de poziţia elementului a_{ij} faţă de cele două diagonale.

Tab. 1.

Poziţia elementelor în matrice	Relaţii
Elemente situate pe diagonala principală	i = j
Elemente situate strict deasupra diagonalei principale	i < j
Elemente situate deasupra şi pe diagonala principală	i ≤ j
Elemente situate strict sub diagonala principală	i > j
Elemente situate sub şi pe diagonala principală	i ≥ j
Elemente situate pe diagonala secundară	i+j = n-1
Elemente situate strict deasupra diagonalei secundare	i+j < n-1
Elemente situate deasupra şi pe diagonala secundară	i+j ≤ n-1
Elemente situate strict sub diagonala secundară	i+j > n-1
Elemente situate sub şi pe diagonala secundară	i+j ≥ n-1

Cu ajutorul acestor relaţii se pot efectua toate operaţiile necesare cu elementele unei matrice pătratice.

O altă modalitate de tratare a cazurilor de mai sus este aceea de a stabili limitele pentru cei doi indici i şi j. Astfel se va putea parcurge doar porţiunea de matrice care interesează şi numărul operaţiilor se reduce corespunzător.

În tabelul 2. ce urmează sunt stabilite limitele inferioară şi superioară precum şi pasul de variaţie pentru indicii i şi j în cazurile prezentate mai sus.

Tab. 2.

Poziţia elementelor în matrice	Relaţii	Element
Elemente situate pe diagonala principală	i=0,n-1, pas=1	a_{ii}
Elemente situate strict deasupra diagonalei principale	i=0,n-2, pas=1; j=i+1,n-1, pas=1	a_{ij}
Elemente situate deasupra şi pe diagonala principală	i=0,n-1, pas=1; j=i,n-1, pas=1	a_{ij}
Elemente situate strict sub diagonala principală	i=1,n-1, pas=1; j=0,i-1, pas=1	a_{ij}
Elemente situate sub şi pe diagonala principală	i=0,n-1, pas=1; j=0,i, pas=1	a_{ij}
Elemente situate pe diagonala secundară	i=0,n-1, pas=1	$a_{i,n-1-i}$
Elemente situate strict deasupra diagonalei secundare	i=0,n-2, pas=1; j=0,n-i-2, pas=1	a_{ij}
Elemente situate deasupra şi pe diagonala secundară	i=0,n-1, pas=1; j=0,n-i-1, pas=1	a_{ij}
Elemente situate strict sub diagonala secundară	i=1,n-1, pas=1; j=n-i,n-1 pas=1	a_{ij}
Elemente situate sub şi pe diagonala secundară	i=0,n-1, pas=1; j=n-i-1,n-1, pas=1	a_{ij}

Iniţializarea unui tablou

Limbajul C oferă posibilitatea iniţializării unui tablou în momentul declarării acestuia. Forma generală de iniţializare a tablourilor este similară cu cea a iniţializării unor variabile. Forma generală de iniţializare a unui tablou:

tip nume_tablou[dimensiune1][dimensiune2]…[dimensiunen]={lista_valori};

lista_valori este o listă de constante separate prin virgulă, cu un tip compatibil cu tip. Prima constantă este plasată pe prima poziţie din tablou, a doua constantă în cea de-a doua poziţie etc.
Dacă constantele de iniţializare au un tip diferit de cel al tabloului, atunci valorile lor se convertesc spre tipul tabloului înainte de a fi atribuite elementelor pe care le iniţializează.

Exemple

int i[10]={1,2,3,4,5,6,7,8,9,10};

Tabloul unidimensional i este iniţializat cu numerele de la 1 la 10. i[0] va avea valoarea 1 şi i[9] primeşte valoarea 10.

char sir[12]=”Imi place C”;

Tablou sir păstrează şirul de caractere “Imi place C”. Fiindcă în C toate șirurile se încheie cu un zero, tabloul care se declară trebuie să fie suficient de lung pentru a include și zero-ul. Din acest motiv sir are 12 caractere, deși fraza ”Imi place C” nu are decât 11 caractere. Când se folosește constanta șir, compilatorul inserează automat caracterul zero de încheiere.

Tablourile multidimensionale se iniţializează asemănător celor unidimensionale:

```
int matrice[4][5] = {
1,2,3,6,7,
5,9,7,6,1,
3,2,4,1,8,
9,8,2,6,5
}
```

Exemple

1. Să se scrie un program care determină elementul maxim şi poziţia acestuia dintr-un şir de elemente reale dat.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 1000
void main(void)
{
int i,n,imax;
double x[MAX],vmax;
printf("Valoarea lui n=");
scanf("%d",&n);
while(n<=0||n>MAX){
  printf("Dimensiune eronata:%d\n",n);
  printf("Introduceti alt n:");
  scanf("%d",&n);
                        }
for(i=0;i<n;i++){
        printf("Elementul x[%d]=",i);
        scanf("%lf",&x[i]);
                }
vmax=x[0];
imax=0;

for(i=1;i<n;i++)

        if(vmax<x[i]){
                vmax=x[i];
                imax=i;
                        }
printf("Valoarea maxima din sir este x[%d]=%lf\n",imax,vmax);
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

2. Să se scrie un program care citeşte doi vectori de numere întregi cu dimensiunea n şi calculează produsul lor scalar.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 1000
void main(void)
{
int i,n;
int x[MAX],y[MAX];
double suma;
printf("Valoarea lui n=");
scanf("%d",&n);
while(n<=0||n>MAX){
  printf("Dimensiune eronata:%d\n",n);
  printf("Introduceti alt n:");
  scanf("%d",&n);
                        }
for(i=0;i<n;i++){
        printf("Elementul x[%d]=",i);
        scanf("%d",&x[i]);
                }
for(i=0;i<n;i++){
        printf("Elementul y[%d]=",i);
        scanf("%d",&y[i]);
                }

for(suma=0,i=0;i<n;i++)
  suma+=x[i]*y[i];
printf("Produsul scalar al celor doi vectori (x,y)=%lf\n",suma);
printf("Apasati orice tasta pentru continuare\n");
getch();
}
```

3. Să se scrie un program care ordonează în ordine descrescătoare un şir de n numere întregi.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAX 1000
void main(void)
{
int i,n,ind=1,aux;
int x[MAX];
printf("Valoarea lui n=");
scanf("%d",&n);
while(n<=0||n>MAX){
  printf("Dimensiune eronata:%d\n",n);
  printf("Introduceti alt n:");
```

```
  scanf("%d",&n);
                              }
for(i=0;i<n;i++){
          printf("Elementul x[%d]=",i);
          scanf("%d",&x[i]);
                        }
printf("Elementele sirului initial neordonat sunt:\n");
for(i=0;i<n;i++){
          printf("%d    ",x[i]);
                         }
while(ind){
  ind=0;
  for (i=0;i<n-1;i++){
  if(x[i]<x[i+1]){
        aux=x[i];
        x[i]=x[i+1];
        x[i+1]=aux;
  ind=1;
                        }
                              }
                }
printf("\nElementele sirului ordonat descrescator sunt:\n");
for(i=0;i<n;i++){
          printf("%d    ",x[i]);
                      }
printf("\nApasati orice tasta pentru continuare\n");
getch();
}
```

4. Să se scrie un program care calculează produsul a două matrice dreptunghiulare a_{mxn} şi b_{nxl} rezultatul fiind depus în matricea dreptunghiulară p_{mxl}.

```
#include<stdio.h>
#define MAX 30
#include<conio.h>
void main(void)
{
int i,j,k,m,n,l;
double a[10][10], b[10][10],p[10][10];
printf ("Introduceti numarul de linii al matricei a m<10,m="),
scanf("%d",&m);
printf ("Introduceti numarul de coloane al matricei a n<10,n="),
scanf("%d",&n);
printf ("Introduceti numarul de coloane al matricei b l<10,l="),
scanf("%d",&l);
for(i=0;i<m;i++){
  for (j=0;j<n;j++) {
  printf("Elementul a(%d,%d)=",i,j);
  scanf("%lf",&a[i][j]);
                              }
```

```
                    }
for(i=0;i<n;i++){
  for (j=0;j<l;j++) {
  printf("Elementul b(%d,%d)=",i,j);
  scanf("%lf",&b[i][j]);
                          }
                        }

for(i=0;i<m;i++){
  for (j=0;j<l;j++) {
  p[i][j]=0;
     for (k=0;k<n;k++)
            p[i][j]=p[i][j]+a[i][k]*b[k][j];
                          }
                         }
printf("\nElementele matricei a:\n");
 for(i=0;i<m;i++){
  for (j=0;j<n;j++) {
   printf("%15.6lf",a[i][j]);
                        }
    printf("\n");
                          }
printf("Elementele matricei b:\n");
 for(i=0;i<n;i++){
  for (j=0;j<l;j++) {
   printf("%15.6lf",b[i][j]);
                        }
    printf("\n");
                          }

printf("\nElementele matricei produs p:\n");
 for(i=0;i<m;i++){
  for (j=0;j<l;j++) {
   printf("%15.6lf",p[i][j]);
                          }
    printf("\n");
                          }
printf("\nApasati orice tasta pentru continuare\n");
getch();
}
```

5. Să se scrie un program care calculează media aritmetică a elementelor strict negative situate deasupra şi pe diagonala secundară a unei matrice pătratice a_{nxn} având elemente numere întregi.

```
#include<stdio.h>
#include<conio.h>
void main(void)
{int i,j,n,nrneg,a[30][30];
double s,ma;
```

```
nrneg=0;
s=0;
printf ("Introduceti dimensiunea matricei a n <30,n=");
scanf("%d",&n);
for(i=0;i<n;i++){
  for (j=0;j<n;j++) {
  printf("Elementul a(%d,%d)=",i,j);
  scanf("%d",&a[i][j]);
                              }
                          }
for(i=0;i<=n-1;i++){
  for (j=0;j<=n-i-1;j++) {
    if(a[i][j]<0){
          s=s+a[i][j];
          nrneg=nrneg+1;
                    }
                              }
                          }
printf("\nElementele matricei a:\n");
 for(i=0;i<n;i++){
  for (j=0;j<n;j++) {
   printf("%10d",a[i][j]);
                          }
    printf("\n");
                        }
if(nrneg==0)
          printf("Media aritmetica nu se poate calcula\n");
else
          {
          ma=s/nrneg;
          printf("Media arimetica:%lf\n",ma);
          }

getch();
}
```

6. Să se scrie un program care calculează transpusa unei matrice pătratice a_{nxn} care să ocupe acelaşi loc în memorie.

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
int i,j,n,a[30][30],aux;
printf ("Introduceti dimensiunea matricei a n <30,n=");
scanf("%d",&n);
for(i=0;i<n;i++){
  for (j=0;j<n;j++) {
  printf("Elementul a(%d,%d)=",i,j);
  scanf("%d",&a[i][j]);
```

```
                        }
                    }
printf("\nElementele matricei a:\n");
 for(i=0;i<n;i++){
  for (j=0;j<n;j++) {
   printf("%10d",a[i][j]);
                        }
    printf("\n");
                    }
for(i=0;i<n;i++){
  for (j=i+1;j<n;j++) {
          aux=a[i][j];
          a[i][j]=a[j][i];
          a[j][i]=aux;
                        }
                    }
printf("\nTranspusa matricei a:\n");
 for(i=0;i<n;i++){
  for (j=0;j<n;j++) {
   printf("%10d",a[i][j]);
                        }
    printf("\n");
                    }
getch();
}
```

Probleme propuse:

7. Să se scrie un program care determină numărul elementelor pare dintr-un șir de numere întregi.

8. Să se scrie un program care calculează suma elementelor impare dintr-un șir de numere reale.

9. Se citește de la tastatură o matrice oarecare (A_{mxn}). Să se scrie un program care să verifice dacă există elemente nule și să se afișeze poziţia lor în matrice.

10. Se citește de la tastatură o matrice oarecare (A_{mxn}). Să se scrie un program care să calculeze media aritmetică a elementelor de pe cele două diagonale.

11. **TEMA!!!** Se citește de la tastatură o matrice oarecare (A_{mxn}). Să se scrie un program care să verifice dacă matricea este simetrică iar în cazul în care nu este, să se schimbe cu 0 toate elementele nesimetrice. (Simetria matricei se verifică faţă de diagonala principală).