Limbaje de Programare
Curs 4 – Tablouri

Dr. Casandra Holotescu

Universitatea Politehnica Timișoara

Ce discutăm azi...

1. Generalități
2. Declarare și parcurgere
3. Adrese
4. Tablouri ca parametri la funcții
5. Tablouri multidimensionale. Matrici

Tablouri

Tablou (vector) = o secvență de elemente de același tip de date

$$\{x_0, x_1, x_2, x_3, \ldots, x_{n-2}, x_{n-1}\}$$

O **valoare $\mathbf{x_k}$** este asociată cu un **indice k**.

ATENȚIE: în C, pentru un tablou de n elemente, indicii vor fi în intervalul $\overline{0, n-1}$, deci **de la** 0 **la** $n-1$.

Obs.: Putem avea și tablouri multidimensionale (bidimensionale–matrice, etc.).

Declarare și inițializare

Declarare:

tip nume-tablou [nr-elemente]

Dimensiunea tabloului: *nr-elemente* = o *constantă* pozitivă.

Exemple:

```
double x[20]; //tablouri unidimensionale
long a[100];
int m[10][20]; //tablou bidimensional
```

Inițializare – între acolade, cu virgulă:

```
int y[4] = {0, 1, 4, 7};
```

Elementele unui tablou

Accesarea unui element:

nume-tablou[indice]

Pentru *indice* putem folosi orice expresie cu valoare întreagă.

```
double v = x[4];
int n = a[i] + 5;
a[i] = a[2*i+j] + 5;
```

Un **element de tablou** poate fi folosit **ca orice variabilă** de tipul respectiv: are o valoare, poate fi folosită în expresii, i se poate atribui o valoare, etc.

Indexarea tablourilor în C

În C, pentru un tablou de **n** elemente, indicii tabloului vor fi **de la 0 la n − 1**.

Indicii încep de la 0, deci pentru:

```
int y[4] = {0, 1, 4, 7};
```

vom avea y[0]==0, y[1]==1, y[2]==4 și y[3]==7

Nu avem elementul y[4] în tabloul y!!

Obs.: Numele tabloului e adresa de la care începe memorarea elementelor.

Citirea tablourilor

Pentru parcurgerea tablourilor folosim de regulă un ciclu **for**:

```
...
int i=0, a[30];
unsigned n;

do
  scanf("%d", &n); //citim nr de elemente
while(n>30);  //repetam cat timp nr citit
//e mai mare decat dimensiunea tabloului

for(i=0;i<n;i++){
    scanf("%d", &a[i]);
    //citim elementul de indice i
}
```

Tipărirea tablourilor

```
...
int i=0;

for(i=0;i<n;i++){
    printf(" %d ", a[i]);
    //tiparim elementul de indice i
}

putchar('\n'); // linie noua
```

Când lucrăm cu tablouri e bine sa scriem funcții specializate pentru citirea/tipărirea tablourilor, pentru a nu repeta inutil secvențe foarte asemănătoare de program.

Adrese

Orice variabilă are o **adresă**, la care se memorează valoarea ei.

Operatorul prefix **&** dă adresa operandului:
&x e adresa variabilei x.

& se poate folosi cu orice **lvalue**(=destinație validă de atribuiri): **variabile, elemente de tablou**.

Nu se poate folosi cu: expresii, constante (nu au adresă!).

În cazul tablourilor, numele unui tablou este chiar adresa sa.

Exemplu: int z[4];

Numele z reprezintă **adresa de început** a tabloului (nu toate elementele sale, împreună, doar adresa sa în memorie!).

Adrese

O adresă poate fi tipărită (în hexazecimal – baza 16) folosind formatul %p în apelul funcției *printf*:

```
#include <stdio.h>
int main(void) {
    double d; int a[6];

    printf("Adresa lui d: %p\n", &d);
    // folosim operatorul &

    printf("Adresa lui a: %p\n", a);
    // a e deja adresa, nu e nevoie de &

    return 0;
}
```

Tablouri ca parametri la funcții

Chiar dacă declarația tabloului alocă și memorie pentru elementele acestuia, numele său nu reprezintă tabloul ca tot unitar, ci doar adresa sa.

⇒ Numele tabloului nu poartă informații despre numărul său de elemente.

Dimensiunea tabloului cu sizeof – dimensiunea totală în octeți:

sizeof(nume-tablou) = nr-elem * sizeof(tip-elem)

⇒ Funcțiile care lucrează cu tablouri au nevoie de:

- adresa de început a tabloului
- numărul de elemente al tabloului

pentru fiecare tablou.

Tablouri ca parametri la funcții

ATENȚIE: numărul de elemente nu se dă între [ ], de exemplu [4], nu va fi luat în considerare! Se trimite ca parametru separat!

```
#include <stdio.h>
void printtab(int t[], unsigned len) {
    for (int i = 0; i < len; ++i) //C99
        printf("%d ", t[i]);
}

int main(void) {
    int prim[10]
      = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29 };
    printtab(prim, 10);
    // ATENTIE: NU prim[10], NU prim[]
    return 0;
}
```

Tablouri ca parametri la funcții

În C, transmiterea parametrilor se face întotdeauna **prin valoare**!

Un tablou dat ca parametru la o funcție se transmite **prin valoarea adresei sale**.

Astfel, având **adresa de început** a tabloului, funcția poate accesa (**citi și scrie**) **elementele tabloului**.

Inițializarea tablourilor:

- tablourile **neinițializate** au elemente de valoare necunoscută
- tablourile **inițializate parțial** au restul elementelor nule

Tablouri ca parametri la funcții

```
#define LEN 3 // macro pt. constanta reutilizata

void sumvect(double a[],
     double b[], double r[], unsigned len) {

   for (unsigned i = 0; i < len; ++i) //C99
      r[i] = a[i] + b[i];
}

int main(void) {
   double a[LEN] = {0, 1.41, 1};
   double b[LEN] = {1, 1.73, 2}, c[LEN];
   sumvect(a, b, c, LEN);
   return 0;
}
```

Tablouri multidimensionale (matrici)

Tablouri multidimensionale = tablouri cu elemente care sunt la rândul lor tablouri.

Declarare:

tip nume-tablou [dim1][dim2]..[dimN]

La fel ca pt. tablouri unidimensionale, dimensiunile sunt constante.

Exemple:

```
int m[10][20]; //tablou bidimensional
double matd[4][5][6]; // 3 dimensiuni
```

Elementele unui tablou multidimensional

Accesarea unui element:

nume-tablou[indice1][indice2]..[indiceN]

Elementele tabloului sunt dispuse în memorie unul după altul, chiar și in cazul tablourilor multidimensionale.

Pentru tabloul:

```
int m[LIN][COL];
```

elementul **m[i][j]** se află pe poziția **i*COL+j**, deci la i*COL+j elemente distanță de începutul tabloului.

Citirea matricilor

```
int i=0, j=0;

for(i=0; i < n; i++) { //matrice cu n linii
    for(j=0; j < m; j++) {  // si m coloane

      scanf("%d", &a[i][j]);
        // citim elementul de indici i si j
        // dintr-o matrice a de elemente int
    }
}
```

Tipărirea matricilor

```
int i=0, j=0;

for(i=0; i < n;i++) { //matrice cu n linii
   for(j=0; j < m; j++) {  // si m coloane

     printf(" %d ", a[i][j]);
       // tiparim elementul de indici i si j
       // dintr-o matrice a de elemente int
    }

   putchar('\n'); //linie noua
}
```

Matrici ca parametri la funcții

Într-un tablou bidimensional **m[LIN][COL]**

elementul **m[i][j]** se află pe poziția **i*COL+j**

⇒ pentru a accesa elem. m[i][j] trebuie să cunoaștem **COL**

Într-un tablou multidimensional **tab[dim1][dim2]...[dimN]**

elementul **tab[i1][i2]..[iN]** se află pe poziția

i1*dim2+i2*dim3+..+i(N-1)*dimN+iN

⇒ pentru a accesa elementul tab[i1][i2]..[iN] trebuie să cunoaștem **dim2, dim3, .., dimN**

(toate dimensiunile tabloului în afară de prima).

Exemple – Înmulțirea matricilor

```
void matmul(double a[][10],
  double b[][6],double c[][6],int lin) {
//doar pentru matrici cu dim. 10 si 6
    for (int i = 0; i < lin; ++i)
        for (int j = 0; i < 6; ++j) {
            c[i][j] = 0;
            for (int k = 0; k < 10; ++k)
                c[i][j] += a[i][k]*b[k][j];
        }
    }
  ... //in main
  double m1[8][10], m2[10][6], m3[8][6];
  ...
  matmul(m1, m2, m3, 8);
  // NU: m1[][], NU: m2[][6], NU: m3[8][6]
```