

Programarea calculatoarelor

Limbajul C



CURS 4



Tablouri



Tablouri unidimensionale

- o colecție finită de elemente de același tip (tip de bază al tabloului) care ocupă un spațiu continuu de memorie.

- Sintaxa:

tip_bază nume_tablou [dimensiune] = {const0, const1, ...} ;

- dimensiune

- numărul de elemente ale tabloului
- expresie întreagă **constantă** (se precizează de obicei printr-o constantă simbolică).

- Memoria ocupată de tablou:

`dimensiune * sizeof(tip_baza)`

Exemple

- `int tab[10];`
*//definește un tablou de 10 elemente întregi, care
// ocupă 40 octeți*
- `#define N 10`
`int tab[N];` *// definiție echivalentă cu cea de mai sus*
- `float v[60]`
- `char c[4];`
- `int v[];` *//Incorect!!*

Inițializare tablou

- Tabloul poate fi inițializat la definire prin precizarea constantelor de inițializare.
- Dacă numărul acestora:
 - ==dimensiune - elementele tabloului se inițializează cu constantele precizate
 - <dimensiune - constantele neprecizate sunt implicit 0
 - >dimensiune - apare eroare la compilare.
- Dacă este prezentă partea de inițializare, dar lipsește dimensiune, aceasta este implicit egală cu numărul constantelor.

Exemple

```
int a[] = {- 1, 0, 4, 7}; // echivalent cu  
int a[4] = {- 1, 0, 4, 7};
```

```
char s[] = "un sir"; // echivalent cu  
char s[7] = {' u', 'n', ' ', 's', 'i', 'r', '\ 0'};
```

```
#define NR_ELEM 5  
float t[NR_ELEM]={1.2,5,3};  
/*primele trei elemente se inițializează cu constantele  
precizate, următoarele două cu 0*/
```

Prelucrare tablou

- *Selectarea unui element de tablou:*
 - se foloseste operatorul de indexare []
 - *nume_tablou[indice]*
 - indice - expresie întreagă cu valori între 0 și dimensiune -1
- Un element de tablou poate fi prelucrat ca orice variabilă având tipul de bază.

Observație

- Nici compilatorul, nici mediul de execuție nu verifică valorile indicilor; programatorul trebuie să codifice astfel încât indicele să ia valori în intervalul

0 .. dimensiune - 1

- pot apare erori imprevizibile
 - modificarea nedorită a altor variabile:

```
#define N 10
int tab[N];
tab[N]=5; //se modifică zona de 4 octeți următoare tabloului
tab[-10]=6;
/* se modifică o zonă de 4 octeți situată la o adresă cu 40 octeți
inferioară tabloului */
```

Prelucrarea tablourilor

- Pentru a realiza o prelucrare asupra tuturor elementelor tabloului se folosește instrucțiunea *for* cu o variabilă contor care să ia toate valorile indicilor (între 0 și dimensiune -1):
- ```
#define N 10
int tab[N],i;
for(i=0; i<N; i++)
 //prelucrare tab[i]
```
- ```
int a[10], i, n, suma=0;
for( i = 0; i < n; i++) suma += a[i];
```


Exemplu

```
#define N 10  
  
....  
int a[N], i, n;  
  
....  
for ( i = 0; i < n; i++) {  
    printf("elem[%d]=",i);  
    scanf("%d", &a[i]);  
}  
for ( i = 0; i < n; i++)  
    printf ("a[%d]=%d\n", i, a[i]);
```

Tablouri multidimensionale

- Definirea unui tablou multidimensional:

tip_bază

```
nume_tablou[dim1][dim2]...[dimn]={const10,...  
, {const20,...}, ..., {constn0,...}};
```

- *dim*₁, *dim*₂, ..., *dim*_{*n*} - expresii întregi constante (de obicei constante simbolice)
- memoria continuă ocupată de tablou:
*dim*₁**dim*₂* ... **dim*_{*n*}*sizeof(*tip_baza*).

Declarare matrice

- Tablourile bidimensionale se numesc matrici
 - prima dimensiune: numărul de linii
 - a doua dimensiune: numărul de coloane.
- `int m[10][5];`
`/* definește un tablou bidimensional de elemente întregi, cu 10 linii și 5 coloane, care ocupă $10 \cdot 5 \cdot 4 = 200$ octeți */`
- `#define NL 10`
`#define NC 5`
`int m[NL][NC];`
`// definiție echivalentă cu cea de mai sus`

Inițializare

- Tabloul poate fi inițializat la definire prin precizarea constantelor de inițializare.
- `int b[2][3] = {1,2,3,4,5,6};` // echivalent cu
- `int b[2][3] = {{ 1,2,3},{ 4,5,6}};` // echivalent cu
- `int b[][3] = {{ 1,2,3},{ 4,5,6}}`
- `double a[3][2]={{2},{5.9,1},{-9}};`
//elementele pe linii sunt: 2 0 / 5.9 1 / -9 0
- `double a[3][2]={2,5.9,1,-9};`
//elementele pe linii sunt: 2 5.9 / 1 -9 / 0 0

Selectare

- Selectarea unui element de tablou:

nume_tablou[ind₁][ind₂]...[ind_n]

- ind₁, ind₂, ..., ind_n - expresii întregi cu valori între 0 și dim_i-1, i=1..n.
- un element de tablou poate fi prelucrat ca orice variabilă având tipul de bază.

Exemplu

```
#define N 10
#define M 10

int main(){
    int matrice[N][M];
    int i, j, n, m;
    //citire cu validare a ordinului matricii
    do {
        printf ("Introduceti nr de linii (1..%d) si de coloane:
                (1..%d)", N,M);
        fflush(stdin);
        scanf ("%d%d", &n, &m);
    } while( n<0 || n>N || m<0 || m>M);
```

Exemplu

```
for ( i=0; i<n; i++ )
    for ( j=0; j<m; j++ )
    {
        printf( "elem[%d][%d]:", i ,j );
        scanf( "%d", &matrice[i][j] );
    }

puts("Matricea:");
for ( i=0; i<n; i++ ){
    for ( j=0; j<m; j++ )
        printf( "%d\t",matrice[i][j] );
    putchar( '\n');
}
getchar();
return 1;
}
```

Șiruri de caractere

- Declarare șiruri

- #define MAX_SIR 100

- char s[MAX_SIR];

sau:

- char s[100];

- determinarea lungimii șirului

- lg =0;

- while (s[lg]!='\0')

- lg++;

Exerciții

- Testarea proprietății de palindrom

```
i=0;
j=lg;
while (s[i]==s[j] && i<j){
    i++;
    j--;
}
if (i >=j)
    printf("\nSirul este palindrom.\n");
else
    printf("\nSirul nu e palindrom.\n");
```

Exerciții

1. Program pentru căutarea secvențială într-un vector a unei valori (prima și ultima apariție).
2. Program pentru citirea unui vector de întregi și extragerea elementelor distincte într-un al doilea vector, care se va afișa.
3. Program pentru calculul normei pe linii a unei matrice de numere reale. Norma este valoarea maximă dintre sumele valorilor absolute ale elementelor din fiecare linie.

Căutarea secvențială într-un vector a unei valori (prima și ultima apariție).

```
#include<stdio.h>
#include<stdlib.h>
#define N 10

int main(){
    int v[N], x, i, n, prim,ultim;
    printf("Introduceti nr de elemente:");
    scanf("%d", &n);
    for ( i = 0; i < n; i++) {
        printf("elem[%d]=",i);
        scanf("%d", &v[i]);
    }
    printf("Introduceti elem cautat:");
    scanf("%d", &x);
```

Continuare exemplu

```
prim=-1; ultim=-1;
for ( i = 0; i < n; i++)
    if(v[i]==x) ultim=i;

for ( i = n-1; i >= 0; i--)
    if(v[i]==x) prim=i;

if (prim!= -1) printf ("prima aparitie in pozitia %d\n", prim);
else printf("elem nu exista in sir!\n");
if (ultim!= -1) printf ("ultima aparitie in pozitia %d\n", ultim);

system("pause");
return 0;
}
```

Extragere elemente distincte dintr-un vector

```
#define MAX 30
#include <stdio.h>
void main () {
    int a[MAX];          /* un vector de intregi*/
    int b[MAX];          /* aici se pun elementele distincte din a*/
    int n, m, i, j, gasit; /* n= dimensiune vector a,
                           m=dimensiune vector b*/

    printf("n="); scanf("%d",&n); /* dimensiune vector*/
    printf ("%d numere intregi:\n",n);
        /* citire vector*/
    for (i=0;i<n;i++) scanf("%d",&a[i]);

    m=0;
```

Extragere elemente distincte dintr-un vector

```
/* caută pe a[i] in vectorul b*/
for (i=0;i<n;i++) {
    gasit=0;
    for (j=0;j<m && !gasit; j++)
        if (a[i]==b[j]) gasit=1;
    if (!gasit) {
        b[m++] =a[i];
    }
}
/* scrie vector b*/
for (j=0;j<m;j++)
    printf ("%3d",b[j]);
}
```

Norma unei matrice (cu numerotare de la 1 ptr linii si col)

```
#include <math.h>
#include <stdio.h>
#define M 20          /* dimensiuni maxime matrice*/
void main () {
    int nl,nc;        /* nl linii, nc coloane*/
    float a[M][M] ;
    int i,j; float s, norma;
    printf("nr.linii: "); scanf("%d",&nl);
    printf("nr.coloane: "); scanf("%d",&nc);
    if (nl >M || nc >M) {
        printf("Eroare: dimensiuni >M \n"); return;
    }
    for (i=0;i<nl;i++)
        for (j=0;j<nc;j++)
            scanf ("%f", &a[i][j]);
```

Norma unei matrice (cu numerotare de la 1 ptr linii si col)

```
norma=0;
for (i=0;i<nl;i++) {
    s=0;
    for (j=0;j<nc;j++)
        s=s+ fabs (a[i][j]);
    if (norma < s)
        norma=s;
}
/* afisare matrice*/
for (i=0;i<nl;i++) {
    for (j=0;j<nc;j++)
        printf ("%f ",a[i][j]);
    printf ("\n");
}
printf ("Norma este:%f \n", norma);
}
```