8. Aplicaţii cu tablouri în C/C++. (C/C++ applications using arrays)

1. Obiective:

- Înţelegerea tablourilor unidimensionale şi multidimensionale din C/C++
- Înţelegerea modului în care se lucrează cu tablourilor unidimensionale şi multidimensionale din C/C++: declarare, iniţializare, accesul la elemente, diferite operaţii
- Scrierea şi rularea de programe simple în care sunt folosite tablouri

1'. Objectives:

- Understanding the unidimensional and multidimensions arrays in C/C++
- Understanding the arrays' operating mode: declaration, initialization, elements' access, various operations
- Writing and running some simple programs that use arrays

2. Breviar teoretic

Un tablou este o secvență de elemente de același tip stocate contigu. Dimensiunea tabloului poate fi o expresie constantă întreagă pozitivă care specifică numărul de elemente. Tipul elementelor este tipul tabloului.
Orice tablou are un nume (care acționează ca un pointer constant la primul element, dar nu este efectiv un pointer). Deci, un nume de tablou este un identificator. Poate fi convenabil să-l considerăm ca un pointer constant în anumite contexte, deoarece comportamentul este în mare parte același și este implicit convertibil catre pointer, dar nu este, în sine, un pointer.

2.1. Tablouri unidimensionale

Declarare: *tip nume_tablou[dimensiune];*

- Pentru referirea unui element se foloseşte operatorul de indexare [], precizând numele tabloului şi poziţia elementului în tablou (*indexul sau indicele*): `nume_tablou[index],`
  `unde index=0,1,2...dimensiune-1.`

2.2. Tablouri multidimensionale

Declarare: `tip nume_tablou[dimens`$_1$`][dimens`$_2$`]...[dimens`$_N$`];`

- Un tablou cu mai multe dimensiuni este de fapt un tablou unidimensional având ca elemente alte tablouri, adica un tablou N-dimensional poate fi privit ca un tablou unidimensional, având ca elemente alte tablouri N-1 dimensionale.
- Referirea unui element se face astfel: `nume_tablou[index`$_1$`][index`$_2$`]...[index`$_N$`];`

2.3. Elemente de tablou

Un element de tablou poate să apară oriunde poate să apară o variabilă simplă cu acelaşi tip ca şi tipul tabloului : atribuiri, citiri de la consolă, afişări, calcule (expresii), apeluri de funcţii.

2.4. Iniţializarea tablourilor la declarare

Iniţializarea se poate face la declarare, prin utilizarea unei liste de valori separate prin virgulă şi delimitată de acolade.

a) tablouri unidimensionale: `tip nume_tablou[dimensiune] = {ec`$_0$`, ..., ec`$_i$`};`

b) tablouri multidimensionale (matrice):
```
tip nume_tablou[n][m] = {{ec11,...,ec1m},
                         {ec21,...,ec2m},...,
                         {ecn1,...,ecnm}};
```

2.5. Prelucrarea tuturor elementelor unui tablou

Se utilizează cel mai frecvent instrucţiunea ciclică *for( )* standard pentru a prelucra secvenţial toate elementele. Variabila de ciclare este folosită ca şi index în tablou, iar o altă variabilă sau constantă simbolică ce conţine dimensiunea tabloului se foloseşte în condiţia asociată instrucţiunii ciclice. Cand nu se face distinctie intre elemente se prefera a folosi o instructiune de tip for_range, introdusa ulterior in C++.

2.6. Transmiterea tablourilor către funcţii

Se poate face prin declararea unui argument de tip tablou, si poate fi fără prima dimensiune. Uzual, se mai transmite şi dimensiun/ea/ile tabloului ca şi argument/e separat.
Fiind o formă de apel prin adresă, orice modificare făcută în funcţie, a elementelor unui tablou transmis ca parametru, se face asupra tabloului iniţial. Dacă însă se foloseşte modificatorul *const* pentru tablou, atunci in functie nu se mai pot modifica valorile elementelor tabloului original.

În cazul tablourilor multidimensionale, prima dimensiune din stanga poate lipsi, celelalte sunt obligatorii.
De exemplu:

```
int tab_fcn(char tablou[][25][50]);
```

2.7. Căutarea în tablouri

De obicei se caută poziţia unui element cu o anumită valoare.
Dacă tabloul nu este sortat (elementele nu sunt aranjate într-o anumită ordine) se aplică *"căutarea liniară"* care constă în parcurgerea secvenţială (unul după altul) a elementelor tabloului şi compararea acestora cu valoarea de căutată. Dacă este găsit un element ce are acea valoare, se returnează indicele elementului, altfel se returnează valoarea -1. Acest algoritm permite găsirea primului element ce are o anumită valoare, însă pot fi mai multe elemente cu aceeaşi valoare.

2.8. Inserarea în tablouri

În cazul tablourilor nesortate, pentru a insera elemente noi într-un tablou trebuie să ţinem cont de următoarele:
- trebuie să existe poziţii libere în tablou;
- pentru elementul nou se crează loc prin deplasarea altor elemente, spre dreapta.

În cazul tablourilor sortate trebuie găsită mai întâi poziţia în care se va face inserarea.

2.9. Ştergerea din tablouri

Ca şi inserarea, ştergerea presupune deplasarea unor elemente, spre stânga.
Se presupune că se cunoaşte poziţia în care se face ştergerea (dacă nu se face căutarea elementului de tablou cu o anumită valoare).

3. Exemple:

Exemplul 1: program ce citeşte elementele unui tablou bidimensional, după care determina suma elementelor de pe diagonala principală.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define MAX     20

int main(void)
{
        int i, j, n, sum = 0, tab[MAX][MAX];
        printf("\nDimensiunea matricei (linii = coloane): ");
        scanf("%d", &n);
        if (n <= 0) {
                printf("\n\tDimensiunea data este negativa !");
                return 1;
        }//end if

        printf("\nIntroduceti elementele matricei:");
        for (i = 0; i<n; i++) {
                printf("\n\t Linia %d: \n", i + 1);
```

```
		for (j = 0; j<n; j++) {
			printf("\t\ttab[%d,%d] = ", i + 1, j + 1);
			scanf("%d", &tab[i][j]);
		}
	}//end for
	for (i = 0; i < n; i++)
		for (j = 0; j < n; j++)
			if (i == j)
				sum += tab[i][j];
	printf("\nSuma elementelor de pe diagonala principala este: %d\n ", sum);
	 return 0;
}
```

Exemplul 2: program pentru determinarea valorii medii a elementelor unui tablou unidimensional.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define DIM 20

float valMed(int x[], int dim);

int main(void)
{
	int i, dim;
	int x[DIM];
	printf("\nIntroduceti dimensiunea tabloului: ");
	scanf("%d", &dim);
	if (dim > DIM) {
		printf("\n Dimensiune prea mare !");
		return 1;		}

	printf("\n Introduceti elementele tabloului:\n");
	for (i = 0; i<dim; i++)
	{
		printf("\tx[%d] = ", i);
		scanf("%d", &x[i]);
	}

	printf("\n Valoarea medie este: %.3f\n", valMed(x, dim));
	return 0;
}

// Functia determina valoarea medie a elementelor unui tablou
float valMed(int x[], int n)
{
	int i;
	int sum = 0;
	for (i = 0; i<n; i++)
		sum += x[i];
	return (float) sum / n;
}
```

Exemplul 3: program pentru căutarea directă într-un tablou unidimensional.

```
#define _CRT_SECURE_NO_WARNINGS
```

```
#include <stdio.h>

#define DIM 100

int cautDir(int x[], int dim, int val);

int main(void)
{
	int i, dim, val, poz;
	int x[DIM];
	printf("\nIntroduceti dimensiunea tabloului( <= %d): ", DIM);
	scanf("%d", &dim);
	if (dim > DIM) {
		printf("\n Dimensiune prea mare !");
		return 1;
	}

	printf("\n Introduceti elementele tabloului:\n");
	for (i = 0; i<dim; i++)
	{
		printf("\tx[%d] = ", i);
		scanf("%d", &x[i]);
	}

	printf("\n Introduceti numarul de cautat: ");
	scanf("%d", &val);
	poz = cautDir(x, dim, val);
	if (poz < 0)
		printf("\n Numarul %d nu a fost gasit in tablou !\n", val);
	else
		printf("\n Numarul %d a fost gasit in pozitia %d.\n", val, poz);
	return 0;
}

// Functie ce cauta o valoare intr-un tablou
// Returneaza pozitia in caz de suuces sau valoarea -1 in caz de esec
int cautDir(int x[], int n, int val)
{
	for (int i = 0; i <= n - 1; i++)
	{
		if (x[i] == val)
			return i;
	}
	return -1;
}
```

Exemplul 4: program pentru inserarea unei valori într-un tablou unidimensional.

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define DIM 5

void inserare(int tab[], int n, int idx, int val);

int main(void)
{
	int i, dim, val, poz;
```

```
	int x[DIM] ;
	printf("\nIntroduceti dimensiunea tabloului: ");
	scanf("%d", &dim);
	if (dim > DIM) {
		printf("\n Dimensiune prea mare !");
		return 1;
	}

	printf("\n Introduceti elementele tabloului:\n");
	for (i = 0; i < dim; i++)
	{
		printf("\tx[%d] = ", i);
		scanf("%d", &x[i]);
	}

	printf("\n Introduceti numarul de inserat: ");
	scanf("%d", &val);
	printf("\n Introduceti pozitia in care se va face inserarea: ");
	scanf("%d", &poz);


	if (dim == DIM) {
		printf("\n Nu mai este spatiu de inserare in tablou !\n");
		return 1;
	}
	if (poz<0 || poz>dim ) {
		printf("\n Nu e o pozitie valida pentru inserare !\n");
		return 1;

	}

	inserare(x, dim, poz, val);
	dim++;
	printf("\nNumerele din tablou :\n");
	for (i = 0; i < dim; i++)
		printf("\t%d", x[i]);//end for
	printf("\n");
	return 0;
}//end main

void inserare(int arr[], int n, int idx, int val)
{
	if (idx == -1)
		arr[n] = val;  // adaugare
	else {
		for (int i = n; i > idx; i--)//deplasare dreapta
			arr[i] = arr[i - 1];
		arr[idx] = val;
	}
}
```

Exemplul 5: program pentru afisarea unui tablou tridimensional din perspectiva xOy.

```
//3D matrix display
#include<stdio.h>

#define Max1 3 //last limit - planes
#define Max2 2 //midle limit - rows
```

```
#define Max3 4 //first limit - columns

int main( ) {
   int i, j, k;
   double arr[Max1][Max2][Max3] = {
   {{-0.1, 0.22, 0.3, 4.3}, {2.3, 4.7, -0.9, 2}},
    {{0.9, 3.6, 4.5, 4}, {1.2, 2.4, 0.22, -1}},
    {{8.2, 3.12, 34.2, 0.1}, {2.1, 3.2, 4.3, -2.0}}
   };

   printf(":::3D Array Elements:::\n\n");
   for (i = 0; i < Max1; i++)
   {
      for (j = 0; j < Max2; j++)
      {
         for (k = 0; k < Max3; k++)
         {
            printf("%6.2lf\t", arr[i][j][k]);
         }
         printf("\n");
      }
      printf("\n");
   }
}//main
```

Exemplul 6: program ce returneaza un tablou static declarat intr-o functie.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX 10

/* function to generate and return random numbers */
int * getRandom( );

/* main function to call the getRandom( ) function */
int main( )
{
  /* a pointer to an int */
  int *p;
  int i;
  p = getRandom( );
  for ( i = 0; i <MAX; i++ )
     printf("\nFrom main p[ %d] :  %d", i, p[i]);
}

int * getRandom( ){
 static int  r[MAX];//must be static
 int i;
 /* set the seed */
 srand( (unsigned)time( NULL ) );
 for(i = 0; i < MAX; ++i)
 {
   r[i] = rand( );//generate random numbers
   printf( "From function r[%d] = %d\n", i, r[i]);
 }
```

```
  return r;
}
```

4. Întrebări:

- Cum se pot accesa elementele unui tablou ?
- Cum se face iniţializarea unui tablou ?
- Ce presupune inserarea, respectiv ştergerea unui element dintr-un tablou ?
- Cum se realizează transmiterea unui tablou într-o funcţie?

5. Teme:

1. Scrieţi un program pentru determinarea valorii medii a elementelor pozitive/negative dintr-un tablou unidimensional.
2. Scrieţi un program pentru determinarea celui mai mic element pozitiv dintr-un tablou unidimensional.
3. Scrieţi o aplicatie C/C++ care citeşte de la tastatură un tablou de 10 valori intregi. Definiţi o funcţie care primeste tabloul ca parametru şi apoi îl afişeaza ordonat crescător.
4. Scrieţi o aplicaţie C/C++ care defineşte o parolă (in format şir de caractere). Programul citeşte în mod repetat şirurile de caractere introduse de la tastatură, până când utilizatorul scrie parola corectă. Să se afişeze numărul de incercări până la introducerea parolei corecte.
5. Scrieţi o aplicatie C/C++ care citeşte de la tastatura două siruri de caractere reprezentand numele si prenumele unei persoane. Afisati-le cu majuscula, separate prin două spatii de tabulare.
6. Scrieţi o aplicatie C/C++ care defineşte două matrici de valori intregi. Dimensiunea si elementele matricilor sunt citite de la tastatura. Scrieţi functiile care:
   a) afişează poziţiile elementelor pare din fiecare matrice;
   b) afişează suma elementelor impare din ambele matrice;
   c) afişează suma matricelor;
7. Citiţi de la tastatura elementele unei matrice cu elemente de tip float, cu dimensiunea 3x3. Rearanjaţi coloanele matricei astfel ca suma elementelor de pe o anumită coloană sa fie mai mica decat suma elementelor de pe coloana următoare.
8. Scrieţi un program care preia de la tastatură "n" valori reale într-un tablou unidimensional, calculează cu o funcţie valoarea medie a valorilor introduse si afişează cu o altă funcţie doar valorile din tablou mai mari decît valoarea medie calculată.
9. Să se scrie o aplicaţie C/C++ în care se citesc într-un tablou unidimensional "n" valori întregi şi se determină numărul elementelor negative impare. Să se afişeze acest număr şi elementele respective.
10. Scrieţi programul care citeşte elementele întregi ale unui tablou unidimensional şi construieşte într-o funcţie un alt tablou unidimensional în care se vor stoca resturile împărţirii elementelor primului tablou la numărul elementelor pozitive din acesta.
11. Se citeşte de la tastatură un şir de caractere. Scrieţi funcţia care inversează şirul şi apoi formează un alt şir de caractere ce va conţine elementele de pe poziţiile pare ale şirului inversat. Afişaţi şirurile obţinute.
12. Să se citească de la tastatură elementele întregi ale unei matrice de dimensiune *m x n.* Dacă matricea este pătratică să se afişeze elementele diagonalei secundare, altfel să se afişeze suma elementelor de pe o coloană dată, c. Valorile m, n şi c se citesc de la tastatură şi se vor scrie funcţii pentru operaţiile cerute.
13. Pornind de la exemplul 5 în care se arată modul de parcurgere a unei matrice tridimensionale din perspectiva *xOy*, să se scrie un program care face parcurgerea matricei considerând perspectiva *xOz* și *yOz*. Afișați datele precizând din ce perspectivă au fost parcurse.
14. Afisați fiecare apariție a fiecărui caracter din alfabetul englez dintr-un șir de testare, "The quick brown fox jumps over the lazy dog.", folosit în vechea telegrafie (cod morse), pe o linie va fi un triplet al caracterului căutat (codul morse), offsetul din interiorul șirului unde a fost găsit, (pozitia relativa fata de inceput) și caracterul potrivit. Afisați șirul de test initial cu prima apariție a fiecărui caracter convertit catre caracterele majuscule daca e cazul (nu caracterele majuscule sau altele diferite de litere).

5'. Homework

1. Write a C/C++ code, that determines the mean value of the negative/positive elements from a one-dimensional array.
2. Write a C/C++ program that determines the value of the smallest positive element of a one-dimensional array.
3. Write a C/C++ application that reads from the keyboard an array of 10 integer values. Define and implement a function that receives the array as parameter and then displays its elements, ordered increasingly.
4. Write a C/C++ application that defines a password (as a string of characters). The program reads repeatedly the entries from the keyboard until the user enters the right password. Display the number of trials the user entered wrong passwords.
5. Write a C/C++ application that that reads from the keyboard two arrays of characters representing the name and surname of a person. Display them with capital letters, separated by a <Tab> space.
6. Write a C/C++ application that defines two matrices of integer values. The matrices' dimensions and their elements are read from the keyboard. Write the functions that display:
   a) the positions of the odd elements in each matrix;
   b) the sum of all even elements in both matrices;
   c) the sum of the two initial matrices;
7. Read from the keyboard the elements of a 3x3 float matrix. Rearrange the columns so that the sum of each column's elements is smaller than the sum of the elements of the next column to the right.
8. Write a program that reads from the keyboard n integer values into a one-dimensional array. Implement a function that calculates their average value. Another function will display the elements from the initial array that are greater than the computed average.
9. Write a program that reads from the keyboard n integer values into a one-dimensional array. Display the number of negative odd elements and their values.
10. Write a program that reads the integer elements of a one-dimensional array. Implement a function that builds another array containing the remainders obtained by dividing the values in the initial array to the number of the positive elements.
11. Read from the keyboard an array of characters. Define the function that reverses the array and populates another array with the elements from the even positions of the reverted array. Display the results.
12. Read from the keyboard the elements of an *m x n* integer matrix.
    If the matrix is square, display the elements from the secondary diagonal. If not, print the sum of all the elements from a certain column, c.
    The values of m, n and c are read from the keyboard. All the operations should be implemented in specific functions.
13. Starting from the example 5 which shows how to display a three-dimensional matrix from the perspective of *xOy*, write a program that goes through the matrix considering the perspective *xOz* and *yOz*. Display the data specifying from which perspective they were browsed.

14. Display each occurrence of each character in the English alphabet in a test string, "The quick brown fox jumps over the lazy dog." (Morse code), on a line there will be a triplet of the searched character (Morse code), the offset inside the string where it was found, (relative position to the beginning) and the appropriate character. Display the initial test string with the first occurrence of each character converted to uppercase if necessary (not uppercase or other non-alphabetical characters).