Universitatea Tehnică a Moldovei

Facultatea *Calculatoare, Informatică și Microelectronică*

Specialitatea *Tehnologii Informaționale*

Raport

la lucrarea de laborator nr. 4

Tema: *"Prelucrarea tablourilor bidimensionale"*

Disciplina: "Programarea Calculatorului"

Varianta 4

A efectuat:	Student grupa TI-231 FR	Apareci Aurica
A verificat:	Asistent universitar	Mantaluță Marius

Chișinău 2023

Cuprins

1. Cadrul teoretic

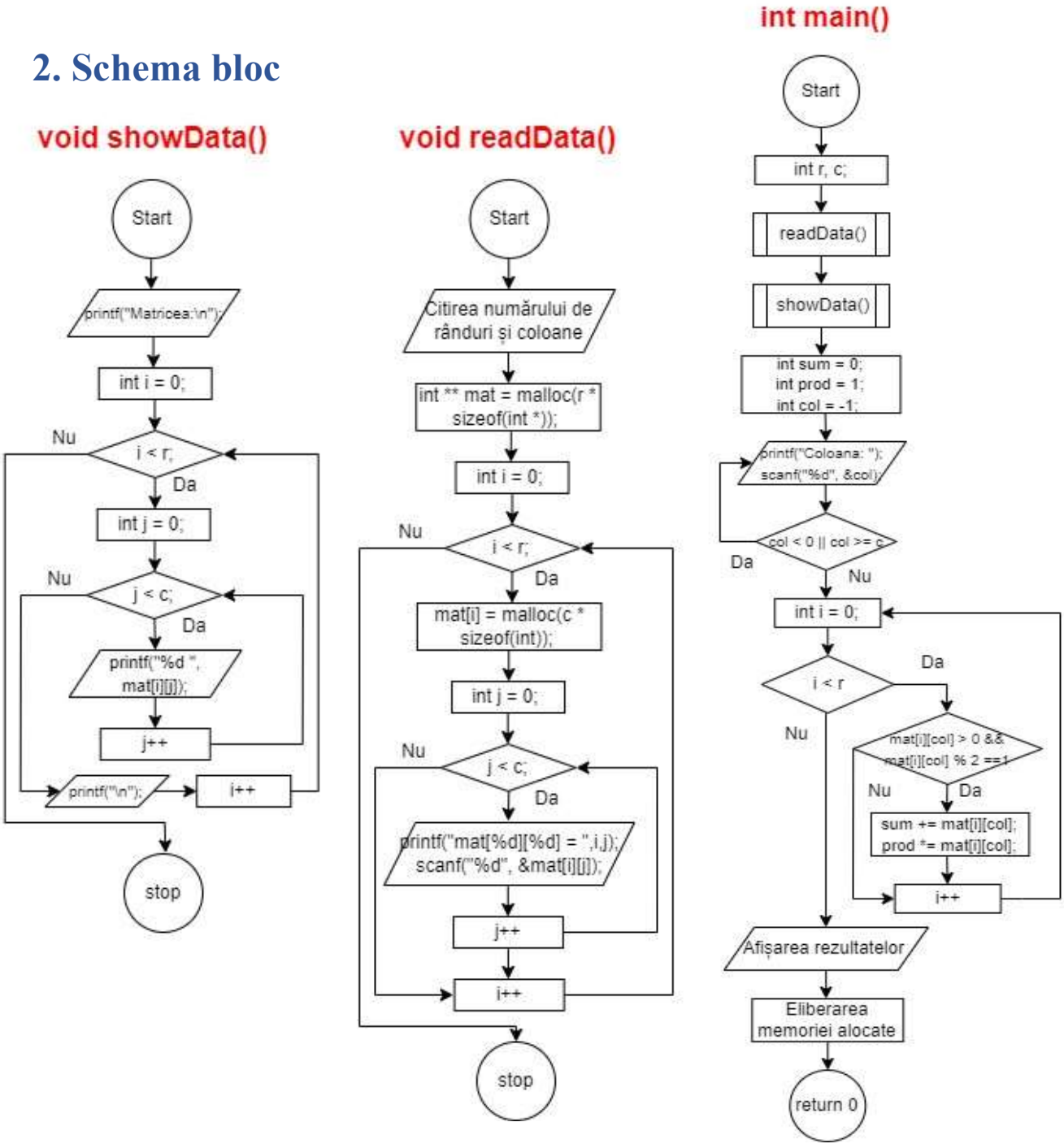
Tema: *Prelucrarea tablourilor bidimensionale.*

Scopul lucrării: Studierea posibilităţilor şi mijloacelor limbajului C pentru programarea algoritmilor cu structură ramificată şi ciclică la prelucrarea tablourilor bidimensionale.

Sarcina: Să se elaboreze schema bloc și programul ce va prelucra un tablou bidimensional cu **N** linii şi **M** coloane, urmând condițiile din **Tabelul 1.**

C(N, N)	De calculat suma și produsul elementelor pozitive impare de pe coloana indicată de utilizator.

2. Schema bloc



3. Listingul programului

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

int r, c;
int **mat;
```

void readData()

```
{
    printf("Numarul de linii: ");
    scanf("%d", &r);
    printf("Numarul de coloane: ");
    scanf("%d", &c);
    mat = malloc(r * sizeof(int *));
    for(int i = 0; i < r; i++)
    {
        mat[i] = malloc(c * sizeof(int));
        for (int j = 0; j < c; j++)
        {
            printf("mat[%d][%d] = ", i, j);
            scanf("%d", &mat[i][j]);
        }
    }
}
```

void showData()

```
{
     printf("Matricea:\n");
    for(int i = 0; i < r; i++)
    {
        for (int j = 0; j < c; j++)
        {
            printf("%d ", mat[i][j]);
        }
        printf("\n");
    }
}
```

int main()

```
{
    readData();
    showData();
    int sum = 0;
    int prod = 1;
    int col = -1;
    do
    {
        printf("Coloana: ");
        scanf("%d", &col);
    } while (col < 0 || col >= c);
    for(int i = 0; i < r; i++)
    {
        if(mat[i][col] > 0 && mat[i][col] % 2 == 1)
        {
            sum += mat[i][col];
            prod *= mat[i][col];
        }
    }
    printf("Suma elementelor pozitive impare de pe coloana %d: %d\n", col, sum);
    printf("Produsul elementelor pozitive impare de pe coloana %d: %d\n", col,
prod);
    for(int i = 0; i < r; i++)
    {
        free(mat[i]);
    }
    free(mat); }
```

4. Testarea aplicației

Nr.	Input	Output
1.	Numarul de linii: 5 Numarul de coloane: 5 mat[0][0] = 1 mat[0][1] = 2 mat[0][2] = 3 mat[0][3] = 4 mat[0][4] = 5 mat[1][0] = 6 mat[1][1] = 7 mat[1][2] = 8 mat[1][3] = 9 mat[1][4] = 1 mat[2][0] = 2 mat[2][1] = 3 mat[2][2] = 4 mat[2][3] = 5 mat[2][4] = 6 mat[3][0] = 7 mat[3][1] = 8 mat[3][2] = 9 mat[3][3] = 1 mat[3][4] = 2 mat[4][0] = 3 mat[4][1] = 4 mat[4][2] = 5 mat[4][3] = 6 mat[4][4] = 1	Matricea: 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 1 Coloana: 3 Suma elementelor pozitive impare de pe coloana 3: 15 Produsul elementelor pozitive impare de pe coloana 3: 45 *Explicație:* Suma = 9+5+1=15 Produs = 9*5*1 = 45
2.	Numarul de linii: 3 Numarul de coloane: 4 mat[0][0] = 15 mat[0][1] = 21 mat[0][2] = 12 mat[0][3] = 33 mat[1][0] = 14 mat[1][1] = 16 mat[1][2] = 32 mat[1][3] = 19 mat[2][0] = 21 mat[2][1] = 19 mat[2][2] = 35 mat[2][3] = 53	Matricea: 15 21 12 33 14 16 32 19 21 19 35 53 Coloana: 1 Suma elementelor pozitive impare de pe coloana 1: 40 Produsul elementelor pozitive impare de pe coloana 1: 399 *Explicație:* Suma = 21+19=40 Produs = 21*19 = 399
3.	Numarul de linii: 2 Numarul de coloane: 2 mat[0][0] = 1 mat[0][1] = 2 mat[1][0] = 3 mat[1][1] = 4	Matricea: 1 2 3 4 Coloana: 0 Suma elementelor pozitive impare de pe coloana 0: 4 Produsul elementelor pozitive impare de pe coloana 0: 3

5. Concluzii

În cadrul acestei lucrări de laborator pe tema prelucrării tablourilor bidimensionale în limbajul de programare C, am dezvoltat un program care permite utilizatorului să introducă dimensiunile unei matrice, să completeze matricea cu valori, după care calculează suma și produsul elementelor pozitive impare de pe coloana indicată de utilizator. Realizarea sarcinii a acoperit mai multe aspecte tehnice importante, precum:

1. Alocarea dinamică de memorie. Una dintre cele mai importante aspecte tehnice a fost alocarea dinamică de memorie pentru matricea bidimensională. Am folosit funcția *`malloc`* pentru a aloca spațiul necesar pentru matrice, în funcție de dimensiunile specificate de utilizator (rânduri și coloane). De asemenea, am eliberat memoria alocată folosind funcția *`free`* pentru a evita scurgerile de memorie.

2. Lucrul cu pointeri la pointeri. Pentru a gestiona matricea bidimensională, am folosit un pointer la pointeri *(`int **mat;`).* Acest lucru ne-a permis să creăm un vector de pointeri la linii, fiecare pointer la linie indicând un vector de intregi, corespunzând coloanelor.

3. Modularizare. Am împărțit codul în funcții separate (funcțiile *`readData()`* și *`showData()`),* care au fost create pentru a gestiona citirea datelor și afișarea matricei.

4. Validarea datelor de intrare. Am inclus verificări pentru datele introduse de utilizator, cum ar fi asigurarea că numărul de linii și coloane este pozitiv și validarea coloanei introduse de utilizator pentru a se asigura că aceasta este în intervalul corect.

5. Calcularea sumei și produsului. Am calculat suma și produsul elementelor pozitive impare dintr-o coloană specificată de utilizator. Această funcționalitate a implicat parcurgerea matricei și aplicarea condițiilor specifice asupra valorilor pentru a realiza calculele.

În concluzie, această lucrare de laborator a abordat aspecte tehnice esențiale pentru lucrul cu matricele bidimensionale în limbajul de programare C, precum gestionarea memoriei, modularizarea codului și manipularea datelor. Aceste concepte sunt fundamentale pentru dezvoltarea programelor eficiente și robuste în C și pot fi aplicate într-o varietate de scenarii de programare.