14.12.2021

FUNDAMENTELE PROGRAMĂRII
– LABORATOR NR. 9 –

1. Se citește un număr natural n. Să se construiască și să se afișeze următoarele matrice pătratice (exemplele sunt pentru cazul $n = 4$):

a)
$$\begin{matrix} 0 & 1 & 2 & 3 \\ 7 & 6 & 5 & 4 \\ 8 & 9 & 10 & 11 \\ 15 & 14 & 13 & 12 \end{matrix}$$

b)
$$\begin{matrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 2 & 2 \\ 0 & 1 & 2 & 3 \end{matrix}$$

c)
$$\begin{matrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{matrix}$$

Rezolvare:

```
#include <stdio.h>

int main()
{
    int m[100][100], i, j, k, n, ct;

    printf("n = ");
    scanf("%d", &n);

    //Punctul a)
    ct = 0;
    for(i=0; i<n; i++)
        if(i%2 == 0)
            for(j=0; j<n; j++)
            {
                m[i][j] = ct;
                ct++;
            }
        else
            for(j=n-1; j>=0; j--)
            {
                m[i][j] = ct;
                ct++;
            }
```

```
printf("\nMatricea de la punctul a):\n");
for(i=0; i<n; i++)
{
    printf("\n");
    for(j=0; j<n; j++)
        printf("%4d ", m[i][j]);
}
printf("\n");

//Punctul b) - varianta 1
for(i = 0; i < n; i++)
{
    for(j = i; j < n; j++)
        m[i][j] = i;
    for(k = i; k < n; k++)
        m[k][i] = i;
}

printf("\nMatricea de la punctul b) - varianta 1:\n");
for(i=0; i<n; i++)
{
    printf("\n");
    for(j=0; j<n; j++)
        printf("%4d ", m[i][j]);
}
printf("\n");

//Punctul b) - varianta 2
for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
        m[i][j] = i < j ? i : j;

printf("\nMatricea de la punctul b) - varianta 2:\n");
for(i=0; i<n; i++)
{
    printf("\n");
    for(j=0; j<n; j++)
        printf("%4d ", m[i][j]);
}
printf("\n");

//Punctul c) - varianta 1
for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
        if(i == j)
            m[i][j] = 0;
        else
            m[i][j] = -1;
```

```
for(ct = 0; ct < n; ct++)
    for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
            if(m[i][j] == ct)
            {
                if(m[i][j+1] == -1)
                    m[i][j+1] = m[i][j] + 1;
                if(m[i+1][j] == -1)
                    m[i+1][j] = m[i][j] + 1;
            }

printf("\nMatricea de la punctul c) - varianta 1:\n");
for(i=0; i<n; i++)
{
    printf("\n");
    for(j=0; j<n; j++)
        printf("%4d ", m[i][j]);
}

printf("\n");

//Punctul c) - varianta 2
for(i = 0; i < n; i++)
{
    m[i][i] = 0;
    for(j = i+1; j < n; j++)
        m[i][j] = m[i][j-1] + 1;
    for(k = i+1; k < n; k++)
        m[k][i] = m[k-1][i] + 1;
}

printf("\nMatricea de la punctul c) - varianta 2:\n");
for(i=0; i<n; i++)
{
    printf("\n");
    for(j=0; j<n; j++)
        printf("%4d ", m[i][j]);
}

printf("\n");

//Punctul c) - varianta 3
for(i = 0; i < n; i++)
    for(j = 0; j < n; j++)
        m[i][j] = i > j ? i-j : j-i;

printf("\nMatricea de la punctul c) - varianta 3:\n");
for(i=0; i<n; i++)
{
    printf("\n");
```

```
        for(j=0; j<n; j++)
            printf("%4d ", m[i][j]);
    }

    printf("\n");

    return 0;
}
```

2. Scrieți un program care să construiască în memorie și să afișeze o matrice pătratică M de dimensiune n, având forma următoare:

- prima coloană conține numerele $1, 2, 3, \dots, n$;
- ultima linie conține numerele $n, n-1, \dots, 2, 1$;
- restul elementelor se calculează ca sumă a elementelor vecine de la vest și sud.

Exemplu: pentru $n = 4$ matricea cerută este:

$$M = \begin{pmatrix} 1 & 9 & 25 & 50 \\ 2 & 8 & 16 & 25 \\ 3 & 6 & 8 & 9 \\ 4 & 3 & 2 & 1 \end{pmatrix}$$

M[0][0] = 1 = 0 + 1 M[1][0] = 2 = 1 + 1 M[2][0] = 3 = 2 + 1 M[3][0] = 4 = 3 + 1 M[i][0] = i+1	M[n-1][0] = 4 = n-0 M[n-1][1] = 3 = n-1 M[n-1][2] = 2 = n-2 M[n-1][3] = 1 = n-3 M[n-1][j] = n - j
Restul elementelor: M[i][j] = M[i][j-1] + M[i+1][j]	

Rezolvare:

```
#include <stdio.h>

int main()
{
    //declar matricea patratica M de dimensiune maxima 100
    int M[100][100];

    //n = numarul de linii si de coloane
    int n, i, j;
```

```
    printf("n =  ");
    scanf("%d", &n);

    //completăm prima coloana
    for(i = 0; i < n; i++)
        M[i][0] = i+1;

    //completăm ultima linie
    for(j = 0; j < n; j++)
        M[n-1][j] = n - j;

    //calculam restul elementelor, de la penultima linie spre prima,
    //iar pe fiecare linie de la a doua coloana spre ultima
    for(i = n-2; i >= 0; i--)
        for(j = 1; j < n; j++)
            M[i][j] = M[i][j-1] + M[i+1][j];

    //afisam matricea
    printf("\n\nMatricea:\n");
    for(i = 0; i < n; i++)
    {
        for(j = 0; j < n; j++)
            printf("%5d ", M[i][j]);
        printf("\n");
    }

    return 0;
}
```

3. Gigel a găsit pe stradă o matrice cu m linii, n coloane și elemente numere întregi. Curios, el își propune să determine, pentru fiecare linie, cea mai mică valoare care se poate obține adunând elementele de pe linia respectivă, cu excepția unuia singur. Scrieți un program care să-l ajute pe Gigel să-și satisfacă curiozitatea cât mai repede!

Exemplu: pentru $m = 4$ și $n = 5$ matricea cerută este:

$M = \begin{pmatrix} 1 & -9 & 25 & 50 & 37 \\ 2 & 8 & 16 & 25 & 35 \\ -3 & -6 & -8 & -9 & -1 \\ 4 & 3 & 12 & 1 & 7 \end{pmatrix}$	`Linia 0: 1+(-9)+25+37 = 54`
	`Linia 1: 2+8+16+25 = 51`
	……

Rezolvare:

Pe fiecare linie, se va calcula suma obținută prin eliminarea maximului de pe linia respectivă!