Vasile Denisa-Georgiana
Grupa 204
Informatica ZI
Anul I
Facultatea de Matematica si Informatica

Tema

1. Se citeste o matrice patratica cu n linii si n coloane (n<=100), cu elemente numere naturale din intervalul [0,1000], avand elementele distincte pe fiecare dintre cele doua diagonale. Interschimbati elementul maxim de pe diagonala principala cu elementul minim de pe diagonala secundara. Afisati matricea rezultata.

```
#include <iostream>

using namespace std;

void Citire_Matrice_Patratica(int numar_de_linii_si_de_coloane, int matrice[][100])
{
    for (int index1 = 0; index1 < numar_de_linii_si_de_coloane; index1++)
           for (int index2 = 0; index2 < numar_de_linii_si_de_coloane; index2++)
                  cin >> matrice[index1][index2];
}

void Gasire_Element_Maxim_de_pe_Diagonala_Principala(int numar_de_linii_si_de_coloane,
int matrice[][100], int& element_maxim, int& index_element)
{
    for (int index = 1; index < numar_de_linii_si_de_coloane; index++)
           if (element_maxim < matrice[index][index])
           {
                  element_maxim = matrice[index][index];
                  index_element = index;
           }
}

void Gasire_Element_Minim_de_pe_Diagonala_Secundara(int numar_de_linii_si_de_coloane,
int matrice[][100], int& element_minim, int& index_element)
{
    for (int index = 1; index < numar_de_linii_si_de_coloane; index++)
           if (element_minim > matrice[index][numar_de_linii_si_de_coloane - 1 -
index])
           {
                  element_minim = matrice[index][numar_de_linii_si_de_coloane - 1 -
index];
                  index_element = index;
           }
}
```

```
void Modificare_Matrice(int numar_de_linii_si_de_coloane, int matrice[][100])
{
    int element_maxim = matrice[0][0], element_minim =
matrice[0][numar_de_linii_si_de_coloane - 1], index_element_maxim = 0,
index_element_minim = 0;
    Gasire_Element_Maxim_de_pe_Diagonala_Principala(numar_de_linii_si_de_coloane,
matrice, element_maxim, index_element_maxim);
    Gasire_Element_Minim_de_pe_Diagonala_Secundara(numar_de_linii_si_de_coloane,
matrice, element_minim, index_element_minim);
    matrice[index_element_maxim][index_element_maxim] = element_minim;
    matrice[index_element_minim][numar_de_linii_si_de_coloane - 1 -
index_element_minim] = element_maxim;
}

void Afisare_Matrice_Patratica(int numar_de_linii_si_de_coloane, int matrice[][100])
{
    Modificare_Matrice(numar_de_linii_si_de_coloane, matrice);
    for (int index1 = 0; index1 < numar_de_linii_si_de_coloane; index1++)
    {
           for (int index2 = 0; index2 < numar_de_linii_si_de_coloane; index2++)
                  cout << matrice[index1][index2]<<" ";
           cout << endl;
    }
}

int main()
{
    int n, matrice[100][100];
    cin >> n;
    Citire_Matrice_Patratica(n, matrice);
    Afisare_Matrice_Patratica(n, matrice);
    return 0;
}
```

2. Se citeste o matrice patratica cu n linii si n coloane (n<=100), cu elemente numere naturale din intervalul [0,1000]. Sa se determine sumele elementelor celor 4 triunghiuri determinate de diagonala principala si cea secundara.

```
#include <iostream>

using namespace std;

void Citire_Matrice_Patratica(int numar_de_linii_si_de_coloane , int matrice[][100])
{
    for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
          for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
                cin >> matrice[index_linie][index_coloana];
}

void Aflare_Sume(int numar_de_linii_si_de_coloane, int matrice[][100], int& suma_sus,
int& suma_jos, int& suma_stanga, int& suma_dreapta)
{
    for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
          for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
          {
                if ((index_coloana + index_linie < numar_de_linii_si_de_coloane - 1)
&& (index_linie < index_coloana))
                      suma_sus = suma_sus + matrice[index_linie][index_coloana];
                else
                {
                      if ((index_coloana + index_linie >
numar_de_linii_si_de_coloane - 1) && (index_linie < index_coloana))
                            suma_dreapta = suma_dreapta +
matrice[index_linie][index_coloana];
                      else
                      {
                            if ((index_coloana + index_linie >
numar_de_linii_si_de_coloane - 1) && (index_linie > index_coloana))
                                  suma_jos = suma_jos +
matrice[index_linie][index_coloana];
                            else
                            {
                                  if ((index_coloana + index_linie <
numar_de_linii_si_de_coloane - 1) && (index_linie > index_coloana))
                                        suma_stanga = suma_stanga +
matrice[index_linie][index_coloana];
                            }
                      }
                }
          }
}

void Afisare_Sume(int numar_de_linii_si_de_coloane, int matrice[][100])
```

```
{
    int suma_sus = 0, suma_jos = 0, suma_stanga = 0, suma_dreapta = 0;
    Aflare_Sume(numar_de_linii_si_de_coloane, matrice, suma_sus, suma_jos,
suma_stanga, suma_dreapta);
    cout << suma_sus << " " << suma_dreapta << " " << suma_jos << " " << suma_stanga;
}

int main()
{
    int n, matrice[100][100];
    cin >> n;
    Citire_Matrice_Patratica(n, matrice);
    Afisare_Sume(n, matrice);
    return 0;
}
```

3. Se citeste o matrice cu n linii si m coloane, (n, m<=100) cu elemente numere naturale din intervalul [0,1000]. Afisati liniile cu numar maxim de de elemente disticte. Sa se stearga dupa aceea liniile cu numar maxim de elemente distincte.

```
#include <iostream>

using namespace std;

void Citire_Matrice(int numar_de_linii, int numar_de_coloane, int matrice[][100])
{
	for (int index_linie = 0; index_linie < numar_de_linii; index_linie++)
		for (int index_coloana = 0; index_coloana < numar_de_coloane;
index_coloana++)
			cin >> matrice[index_linie][index_coloana];
	cout << endl;
}

int Verificare_Numar_de_Elemente_Distince(int numar_coloane, int matrice[][100], int
linie)
{
	int nr_elemente_distincte = 0, vector[100];
	for (int index = 0; index < numar_coloane; index++)
		vector[index] = matrice[linie][index];
	for (int index = 0; index < numar_coloane; index++)
	{
		int conditie = 1;
		for (int index2 = 0; index2 < numar_coloane && conditie == 1; index2++)
			if (index2 != index && vector[index] == vector[index2])
				conditie = 0;
		if (conditie) nr_elemente_distincte++;
	}
	return nr_elemente_distincte;
}

int Numarul_Maxim_de_Elemente_Distincte(int numar_coloane, int matrice[][100])
{
	int maxim = -1;
	for (int index = 0; index < numar_coloane; index++)
	{
		if (Verificare_Numar_de_Elemente_Distince(numar_coloane, matrice, index) >
maxim)
		{
			maxim = Verificare_Numar_de_Elemente_Distince(numar_coloane,
matrice, index);
		}
	}
	return maxim;
}

void Eliminare_Linii(int& numar_linii,int numar_coloane, int matrice[][100], int pozitie)
{
	for (int index1 = pozitie; index1 < numar_linii; index1++)
		for (int index2 = 0; index2 < numar_coloane; index2++)
			matrice[index1][index2] = matrice[index1 + 1][index2];
	numar_linii--;
```

```
}

void Afisare_Linii_cu_Nr_Maxim_de_Elemente_Distincte(int numar_linii, int numar_coloane,
int matrice[][100])
{
	int maxim_aparitii_distincte = Numarul_Maxim_de_Elemente_Distincte(numar_coloane,
matrice);
	for (int index = 0; index < numar_linii; index++)
	{
		if (Verificare_Numar_de_Elemente_Distince(numar_coloane, matrice, index) ==
maxim_aparitii_distincte)
		{
			for (int index2 = 0; index2 < numar_coloane; index2++)
				cout << matrice[index][index2] << " ";
			cout << endl;
			Eliminare_Linii(numar_linii,numar_coloane, matrice, index);
			index--;
		}
	}
}

int main()
{
	int n, m, matrice[100][100];
	cin >> n >> m;
	Citire_Matrice(n, m, matrice);
	Afisare_Linii_cu_Nr_Maxim_de_Elemente_Distincte(n, m, matrice);
	return 0;
}
```

4. Se citeste o matrice patratica cu n linii si n coloane (n<=100), cu elemente numere naturale din intervalul [0,1000]. Ordonati crescator elementele de pe diagonala principala prin interschimbari de linii si coloane. Afisati matricea rezultata.

```
#include <iostream>

using namespace std;

void Citire_Matrice_Patratica(int numar_de_linii_si_de_coloane, int matrice[][100])
{
    for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
           for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
                  cin >> matrice[index_linie][index_coloana];
}

void Interschimbare(int& numar1, int& numar2)
{
    int numar_auxiliar = numar1;
    numar1 = numar2;
    numar2 = numar_auxiliar;
}

void Ordonare_Elemente_de_pe_Diagonala_Principala(int numar_de_linii_si_de_coloane,
int matrice[][100])
{
    for (int index1 = 0; index1 < numar_de_linii_si_de_coloane-1; index1++)
    {
           for (int index2 = index1; index2 < numar_de_linii_si_de_coloane; index2++)
           {
                  if (matrice[index1][index1] > matrice[index2][index2])
                         Interschimbare(matrice[index1][index1],
matrice[index2][index2]);
           }
    }
}

void Afisare_Matrice(int numar_de_linii_si_de_coloane, int matrice[][100])
{
    Ordonare_Elemente_de_pe_Diagonala_Principala(numar_de_linii_si_de_coloane,
matrice);
    for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
    {
           for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
                  cout << matrice[index_linie][index_coloana] << " ";
           cout << endl;
    }

}
```

```
int main()
{
    int n, matrice[100][100];
    cin >> n;
    Citire_Matrice_Patratica(n, matrice);
    Afisare_Matrice(n, matrice);
    return 0;
}
```

5. Sa se roteasca o matrice patratica, cu n linii si n coloane, cu 90 de grade in sensul acelor de ceas.

```
#include <iostream>

using namespace std;

void Citire_Matrice_Patratica(int numar_de_linii_si_de_coloane, int matrice[][100])
{
    for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
           for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
                  cin >> matrice[index_linie][index_coloana];
}

void Rotire_Matrice(int numar_de_linii_si_de_coloane, int matrice[][100])
{
    int matrice_auxiliara[100][100];
    for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
           for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
                  matrice_auxiliara[index_linie][index_coloana] =
matrice[index_linie][index_coloana];
    for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
           for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
                  matrice[index_linie][index_coloana] =
matrice_auxiliara[numar_de_linii_si_de_coloane - 1 - index_coloana][index_linie];
}

void Afisare_Matrice(int numar_de_linii_si_de_coloane, int matrice[][100])
{
    Rotire_Matrice(numar_de_linii_si_de_coloane, matrice);
    for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
    {
           for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
                  cout << matrice[index_linie][index_coloana] << " ";
           cout << endl;
    }

}

int main()
{
    int n, matrice[100][100];
    cin >> n;
    Citire_Matrice_Patratica(n, matrice);
    Afisare_Matrice(n, matrice);
return 0;
}
```

6. Se citeste o matrice patratica cu n linii si n coloane (n<=100) cu elemente numere naturale din intervalul [0,1000]. Sa se interschimbe elementele simetrice fata de diagonala principala care au aceeasi paritate si sa se afiseze matricea rezultata.

Exemplu: 4
3 4 1 6
3 4 2 1
5 6 5 7
2 4 3 6
=> 3 4 5 2
3 4 6 1
1 2 5 3

6 4 7 6

```
#include <iostream>

using namespace std;

void Citire_Matrice_Patratica(int numar_de_linii_si_de_coloane, int matrice[][100])
{
	for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
		for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
			cin >> matrice[index_linie][index_coloana];
}

void Interschimbare(int& numar1, int& numar2)
{
	int numar_auxiliar = numar1;
	numar1 = numar2;
	numar2 = numar_auxiliar;
}

void Interschimbare_Elemente_Simetrice_de_acceasi_Paritate(int
numar_de_linii_si_de_coloane, int matrice[][100])
{
	for (int index1 = 0; index1 < numar_de_linii_si_de_coloane ; index1++)
	{
		for (int index2 = 0; index2 < numar_de_linii_si_de_coloane; index2++)
		{
			if (matrice[index1][index2] % 2 == matrice[index2][index1] % 2 &&
index1 != index2 && index1<index2)
				Interschimbare(matrice[index1][index2],
matrice[index2][index1]);
		}
	}
}

void Afisare_Matrice(int numar_de_linii_si_de_coloane, int matrice[][100])
```

```
{
	Interschimbare_Elemente_Simetrice_de_acceasi_Paritate(numar_de_linii_si_de_coloane
, matrice);
	cout << endl;
	for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
	{
		for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
			cout << matrice[index_linie][index_coloana] << " ";
		cout << endl;
	}

}

int main()
{
	int n, matrice[100][100];
	cin >> n;
	Citire_Matrice_Patratica(n, matrice);
	Afisare_Matrice(n, matrice);
	return 0;
}
```

7. Construiti si afisati o matrice patratica de ordin n dupa modelul de mai jos pentru care n=5:

1 2 3 4 5
2 1 2 3 4
3 2 1 2 3
4 3 2 1 2
5 4 3 2 1

```
#include <iostream>

using namespace std;

void Generare_Matrice(int numar_de_linii_si_de_coloane, int matrice[][100])
{
	for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
		for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
			if (index_linie <= index_coloana)
				matrice[index_linie][index_coloana] = (index_coloana -
index_linie) + 1;
			else
				matrice[index_linie][index_coloana] = (index_linie -
index_coloana) + 1;
}

void Afisare_Matrice(int numar_de_linii_si_de_coloane, int matrice[][100])
{
		Generare_Matrice(numar_de_linii_si_de_coloane, matrice);
		for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
		{
			for (int index_coloana = 0; index_coloana <
numar_de_linii_si_de_coloane; index_coloana++)
				cout << matrice[index_linie][index_coloana] << " ";
			cout << endl;
		}

}

int main()
{
	int n, matrice[100][100];
	cin >> n;
	Afisare_Matrice(n, matrice);
	return 0;
}
```

8. Se citeste o matrice patratica de ordin n. Parcurgeti si afisati elementele din matrice incepand cu elementul din coltul stanga sus, mergand paralel cu diagonala secundara, ca in exemplu.

Exemplu

n=4, matricea:

1 3 4 10
2 5 9 11
6 8 12 15
7 13 14 16

In urma parcurgerii se vor afisa numerele: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

```
#include <iostream>

using namespace std;

void Citire_Matrice_Patratica(int numar_de_linii_si_de_coloane, int matrice[][100])
{
	for (int index_linie = 0; index_linie < numar_de_linii_si_de_coloane;
index_linie++)
		for (int index_coloana = 0; index_coloana < numar_de_linii_si_de_coloane;
index_coloana++)
			cin >> matrice[index_linie][index_coloana];
}

void Afisare_Numere(int numar_de_linii_si_de_coloane, int matrice[][100])
{
	for(int index=0;index<2*numar_de_linii_si_de_coloane;index++)
		for (int index2 = 0; index2 < index; index2++)
		{
			if (index - 1 - index2 < numar_de_linii_si_de_coloane && index2 <
numar_de_linii_si_de_coloane)
			{
				if (index % 2 == 0)
					cout << matrice[index - 1 - index2][index2] << " ";
				else cout << matrice[index2][index - 1 - index2] << " ";
			}
		}
}

int main()
{
	int n, matrice[100][100];
	cin >> n;
	Citire_Matrice_Patratica(n, matrice);
	Afisare_Numere(n, matrice);
	return 0;
}
```

9. Se citeste un numar natural n patrat perfect si apoi n numere naturale. Sa se creeze o matrice patratica care sa contina toate cele n numere citite, in care elementele sa fie completate in spirala in sens invers al acelor de ceas (in sens trigonometric). Sa se afiseze matricea construita.

Exemplu:

numere.in

9

2 6 7 3 7 1 7 1 5

numere.out

2 1 7

6 5 1

7 3 7

```
#include <iostream>
#include <fstream>
#include <cmath>

using namespace std;

ifstream in("numere.in");
ofstream out("numere.out");

void Construire_Matrice(int marime_matrice, int matrice[][100])
{
	int index = 0;
	while (index < marime_matrice / 2)
	{
		int linie = index - 1, coloana = index;
		while (linie < marime_matrice - 1 - index)
			in >> matrice[++linie][coloana];
		while (coloana < marime_matrice - 1 - index)
			in >> matrice[linie][++coloana];
		while (linie > index)
			in >> matrice[--linie][coloana];
		while (coloana > index + 1)
			in >> matrice[linie][--coloana];
		index++;
		if (marime_matrice % 2 == 1)
			in >> matrice[marime_matrice / 2][marime_matrice / 2];
	}
}

void Afisare_Matrice(int numar)
{
```

```
	int matrice[100][100], marime_matrice = sqrt(numar);
	Construire_Matrice(marime_matrice, matrice);
	for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
	{
		for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
			out << matrice[index_linie][index_coloana] << " ";
		out << endl;
	}
}

int main()
{
	int n;
	in >> n;
	Afisare_Matrice(n);
	return 0;
}
```

10. Se da o matrice patratica de dimensiune n, matrice care trebuie sa contina toate numerele intre 0 si n^2-1.
a. sa se verifice daca matricea data respecta conditia ceruta

```
#include <iostream>

using namespace std;

void Citire_Matrice_Patratica(int marime_matrice, int matrice[][100])
{
	for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
			cin >> matrice[index_linie][index_coloana];
}

bool Verificare_Matrice(int marime_matrice, int matrice[][100])
{
	Citire_Matrice_Patratica(marime_matrice, matrice);
	int vector_frecventa[10000];
	for (int index = 0; index <= marime_matrice * marime_matrice - 1; index++)
		vector_frecventa[index] = 0;
	for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
			vector_frecventa[matrice[index_linie][index_coloana]]++;
	for (int index = 0; index <marime_matrice * marime_matrice; index++)
		if (vector_frecventa[index] == 0)
			return false;
	return true;
}

int main()
{
	int n, matrice[100][100];
	cin >> n;
	if (Verificare_Matrice(n, matrice)) cout << "Matricea data contine toate numerele
intre 0 si n^2-1. ";
	else cout << "Matricea data nu contine toate numerele intre 0 si n^2-1. ";
	return 0;
}
```

b. sa se localizeze punctul cu valoarea 0 (pentru matricile valide) si sa se scrie o functie care returneaza toate cele maxim 4 matrici care se pot forma mutand 0 N, S, E V.

```
#include <iostream>

using namespace std;

void Citire_Matrice_Patratica(int marime_matrice, int matrice[][100])
{
	for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
			cin >> matrice[index_linie][index_coloana];
	cout << endl;
}

void Interschimbare(int& numar1, int& numar2)
{
	int numar_auxiliar = numar1;
	numar1 = numar2;
	numar2 = numar_auxiliar;
}

bool Verificare_Matrice(int marime_matrice, int matrice[][100])
{
	Citire_Matrice_Patratica(marime_matrice, matrice);
	int vector_frecventa[10000];
	for (int index = 0; index <= marime_matrice * marime_matrice - 1; index++)
		vector_frecventa[index] = 0;
	for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
			vector_frecventa[matrice[index_linie][index_coloana]]++;
	for (int index = 0; index < marime_matrice * marime_matrice; index++)
		if (vector_frecventa[index] == 0)
			return false;
	return true;
}

void Generare_Matrici(int marime_matrice, int matrice[][100], int linie, int coloana)
{
	if (linie - 1 >= 0)
	{
		Interschimbare(matrice[linie][coloana], matrice[linie - 1][coloana]);
		for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		{
			for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
				cout << matrice[index_linie][index_coloana] << " ";
			cout << endl;
		}
		cout << endl;
		Interschimbare(matrice[linie][coloana], matrice[linie - 1][coloana]);
```

```
	}
	if (linie + 1 < marime_matrice)
	{
		Interschimbare(matrice[linie][coloana], matrice[linie + 1][coloana]);
		for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		{
			for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
				cout << matrice[index_linie][index_coloana] << " ";
			cout << endl;
		}
		cout << endl;
		Interschimbare(matrice[linie][coloana], matrice[linie + 1][coloana]);
	}
	if (coloana - 1 < marime_matrice)
	{
		Interschimbare(matrice[linie][coloana], matrice[linie][coloana - 1]);
		for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		{
			for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
				cout << matrice[index_linie][index_coloana] << " ";
			cout << endl;
		}
		cout << endl;
		Interschimbare(matrice[linie][coloana], matrice[linie][coloana - 1]);
	}
	if (coloana + 1 >= 0)
	{
		Interschimbare(matrice[linie][coloana], matrice[linie][coloana + 1]);
		for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		{
			for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
				cout << matrice[index_linie][index_coloana] << " ";
			cout << endl;
		}
		cout << endl;
		Interschimbare(matrice[linie][coloana], matrice[linie][coloana + 1]);
	}
}

void Cautare_Zero(int marime_matrice, int matrice[][100])
{
	int linie_zero, coloana_zero;
	for (int index_linie = 0; index_linie < marime_matrice; index_linie++)
		for (int index_coloana = 0; index_coloana < marime_matrice;
index_coloana++)
			if (matrice[index_linie][index_coloana] == 0)
			{
				linie_zero = index_linie;
				coloana_zero = index_coloana;
			}
	Generare_Matrici(marime_matrice, matrice, linie_zero, coloana_zero);
}

int main()
{
```

```
        int n, matrice[100][100];
        cin >> n;
        if (Verificare_Matrice(n, matrice)) Cautare_Zero(n, matrice);
        else cout << "Matricea data nu contine toate numerele intre 0 si n^2-1. ";
        return 0;
}
```