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;

}

1. 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;

}

1. 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;

}

1. 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;

}

1. 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;

}

1. 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;

}

1. 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;

}

1. 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;

}

1. 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;

}

1. Se da o matrice patratica de dimensiune n, matrice care trebuie sa contina toate numerele intre 0 si n^2-1.
2. 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;

}

1. 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;

}