***Матриця змійкою***

void snake(int\*\* matrix, int rows, int cols) {

int first\_number = -10;

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

if (i % 2 == 0) {

matrix[i][j] = first\_number++;

}

else {

matrix[i][j] = first\_number + cols - 1 - j;

}

}

if (i % 2 != 0) first\_number += cols;

}

}

***Матриця спіралью***

void spiral(int\*\* matrix, int n) {

int number = 1;

for (int i = 0; i < n / 2; i++) {

for (int j = i; j < n - i; j++) matrix[i][j] = number++;

for (int j = 1; j < n - i - i; j++) matrix[j + i][n - i - 1] = number++;

for (int j = n - 2 - i; j >= i; j--) matrix[n - i - 1][j] = number++;

for (int j = n - i - 2; j > i; j--) matrix[j][i] = number++;

}

if (n % 2 != 0) matrix[n / 2][n / 2] = number;

}

***Сума ромба в матриці***

int get\_rhomb\_sum(int\*\* matrix, int size) {

int sum = 0;

int correction = 0;

for (int i = 3; i <= size; i += 2)

correction++;

if (size % 2 == 0)

correction++;

for (int i = 0; i < size; i++)

for (int j = 0; j < size; j++)

if (i >= size - j - 1 - correction && i <= j + correction && i >= j - correction && i <= size - j - 1 + correction)

sum += matrix[i][j];

return sum;

}

***Сума лівого і правого трикутників***

int get\_triangle\_sum(int\*\* matrix, int size) {

int sum = 0;

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

if ((i >= j && i <= size - j - 1) || (i <= j && i >= size - j - 1)) {

sum += matrix[i][j];

}

}

}

return sum;

}

***Сума нижнього і верхнього трикутників***

int get\_triangle\_sum(int\*\* matrix, int size) {

int sum = 0;

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

if ((i >= j && i >= size - j - 1) || (i <= j && i <= size - j - 1)) {

sum += matrix[i][j];

}

}

}

return sum;

}

***Піднести матрицю до квадрату***

int\*\* matrix\_squared(int\*\* matrix,int n) {

int\*\* new\_matrix = new int\*[n];

for (int i = 0; i < n; i++) {

new\_matrix[i] = new int[n];

for (int j = 0; j < n; j++) new\_matrix[i][j]=0;

}

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

for (int k = 0; k < n; k++) {

new\_matrix[i][j] += (matrix[i][k]) \* (matrix[k][j]);

}

}

}

return new\_matrix;

}

***Створення матриці у вигляді***

double\*\* get\_matrix(int size) {

double\*\* matrix = new double\*[size];

for (int i = 0; i < size; i++) {

matrix[i] = new double[size];

}

for (int j = 0; j < size; j++) {

cin >> matrix[0][j];

}

int correction;

for (int i = 1; i < size; i++) {

for (int j = 0; j <size; j++) {

if (j != size - 1) matrix[i][j] = matrix[i - 1][j + 1];

else matrix[i][j] = matrix[i-1][0];

}

}

return matrix;

}

***Транспонування матриці***

int\*\* matrix\_transposition(int\*\* matrix, int rows,int cols) {

int\*\* new\_matrix = new int\* [cols];

for (int i = 0; i < cols; i++) {

new\_matrix[i] = new int[rows];

}

for (int i = 0; i < rows; i++) {

for (int j = 0; j < cols; j++) {

new\_matrix[j][i] = matrix[i][j];

}

}

return new\_matrix;

}

***Видалити певний рядок***

void delete\_row(int \*\*&matrix, int rows, int cols, int row) {

int\*\* new\_matrix = new int\* [rows - 1];

for (int i = 0; i < rows - 1; i++) {

new\_matrix[i] = new int[cols];

}

row--;

int correction = 0;

for (int i = 0; i < rows - 1; i++) {

if (i == row) {

correction++;

}

for (int j = 0; j < cols; j++) {

new\_matrix[i][j] = matrix[i+correction][j];

}

}

for (int i = 0; i < rows; i++) {

delete[]matrix[i];

}

matrix = new\_matrix;

}

***Видалити певний стовпець***

void delete\_col(int\*\*& matrix, int rows, int cols, int col) {

int\*\* new\_matrix = new int\* [rows];

for (int i = 0; i < rows; i++) {

new\_matrix[i] = new int[cols-1];

}

col--;

int correction = 0;

for (int j = 0; j < cols - 1; j++) {

if (j == col) correction++;

for (int i = 0; i < rows; i++) {

new\_matrix[i][j] = matrix[i][j+correction];

}

}

for (int i = 0; i < rows; i++) {

delete[]matrix[i];

}

matrix = new\_matrix;

}

***Сортування матриці***

void sort\_matrix(int\*\* matrix, int rows,int cols) {

int temp\_j;

int row;

for (int i = 0; i < rows \* cols - 1; i++) {

for (int j = 0; j < rows \* cols - i - 1; j++) {

row = 0;

temp\_j = j;

while (temp\_j >= cols) {

temp\_j -= cols;

row++;

}

if (temp\_j + 1 < cols && matrix[row][temp\_j] > matrix[row][temp\_j + 1])

swap(matrix[row][temp\_j], matrix[row][temp\_j + 1]);

else if (temp\_j + 1 == cols && matrix[row][temp\_j] > matrix[row + 1][0])

swap(matrix[row][temp\_j], matrix[row + 1][0]);

}

}

}