МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпропетровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Технічна Кібернетика»

**Лабораторна робота №4**

**з дисципліни «Основи програмування»**

**на тему: «**Використання вкладених циклів в задачах обробки матриць.**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О.И.

Дніпро, 2019

**Тема.** Використання вкладених циклів в задачах обробки матриць.

**Мета.** Навчитися використовувати багатомірні [масиви](http://lider.diit.edu.ua/mod/lesson/view.php?id=40269) та вкладені цикли в розв’язанні задач обробки матриць.

# **Постановка завдання.**

Варіант 5, рівень Б) Дано дійсну квадратну матрицю порядку 2N. Отримати нову матрицю, переставляючи її блоки N x N за годинниковою стрілкою.

**Вимоги до програми:**

- вхідні дані вводяться з клавіатури;

- передбачити перевірку вхідних даних на відповідність діапазону значень і некоректні символи;

- результати роботи програми виводяться на екран;

**Вимоги до тексту програми:**

- коментарі щодо призначення програми, її вхідних і вихідних даних;

- коментарі щодо призначення кожного блоку програми, дій окремих операторів для пояснення алгоритму;

- самодокументованість коду: всі ідентифікатори повинні мати назви, що відповідають суті змінних.

# **Зовнішні специфікації програми.**

*Формат вхідних даних*:

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| Порядок матриці | M | Натуральне число, кратне двум. | 4 |
| Елементи матриці | Matrix\_in[i][j] | Дійсне число. | 54354 |

Матриця вводиться зліва на право, зверху вниз.

i – номер рядка, j – номер стовпця.

*Формат вихідних даних:*

|  |  |  |  |
| --- | --- | --- | --- |
| Найменування даних | Умовне позначення | Вимоги до даних | Приклад |
| Елементи нової матриці | Matrix\_out[i][j] | Дійсне число. | 23414 |

Матриця виводиться зліва на право, зверху вниз.

i – номер рядка, j – номер стовпця.

# **Метод рішення завдання.**

1. Отримати M = 2N, перевірити на кратність двом(M%2=0).
2. Отримати матрицю MxM(2Nx2N).
3. Зробити чотири матриці NxN(чверті матриці MxM).
4. Записати у нову матрицю MxM чотири матриці NxN, помінявши місцями на 90 градусів за годинниковою стрілкою.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | → | 4 | 1 |
| 4 | 3 | → | 3 | 2 |

1. Вивести нову матрицю на єкран.

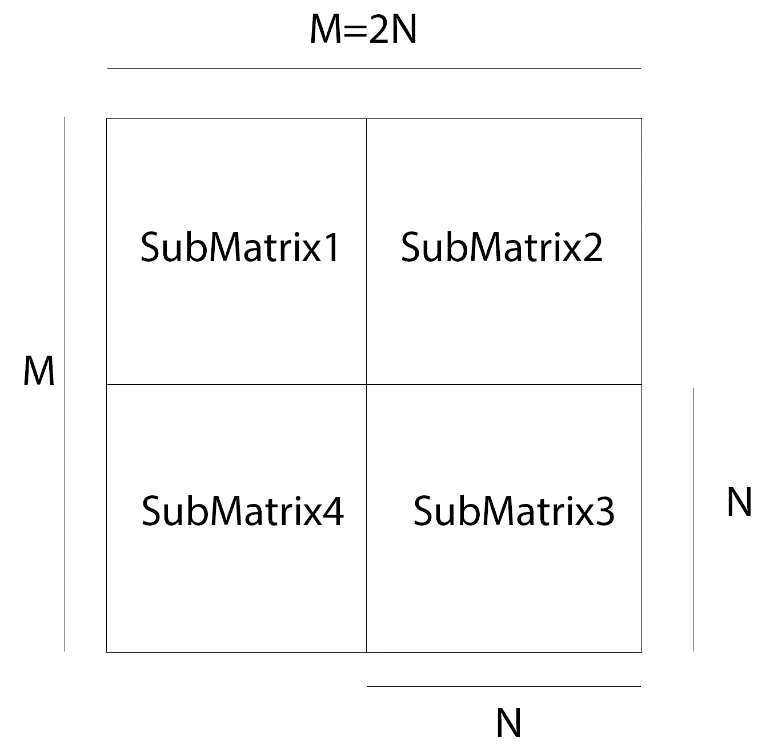
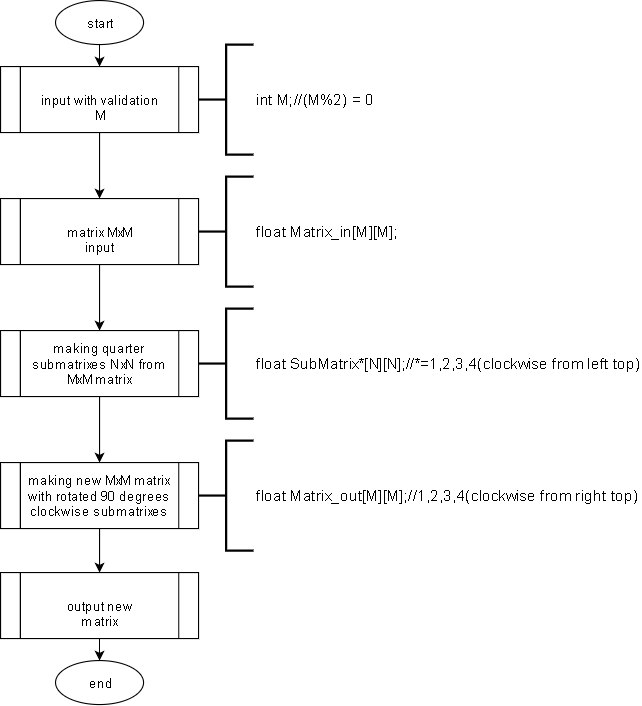
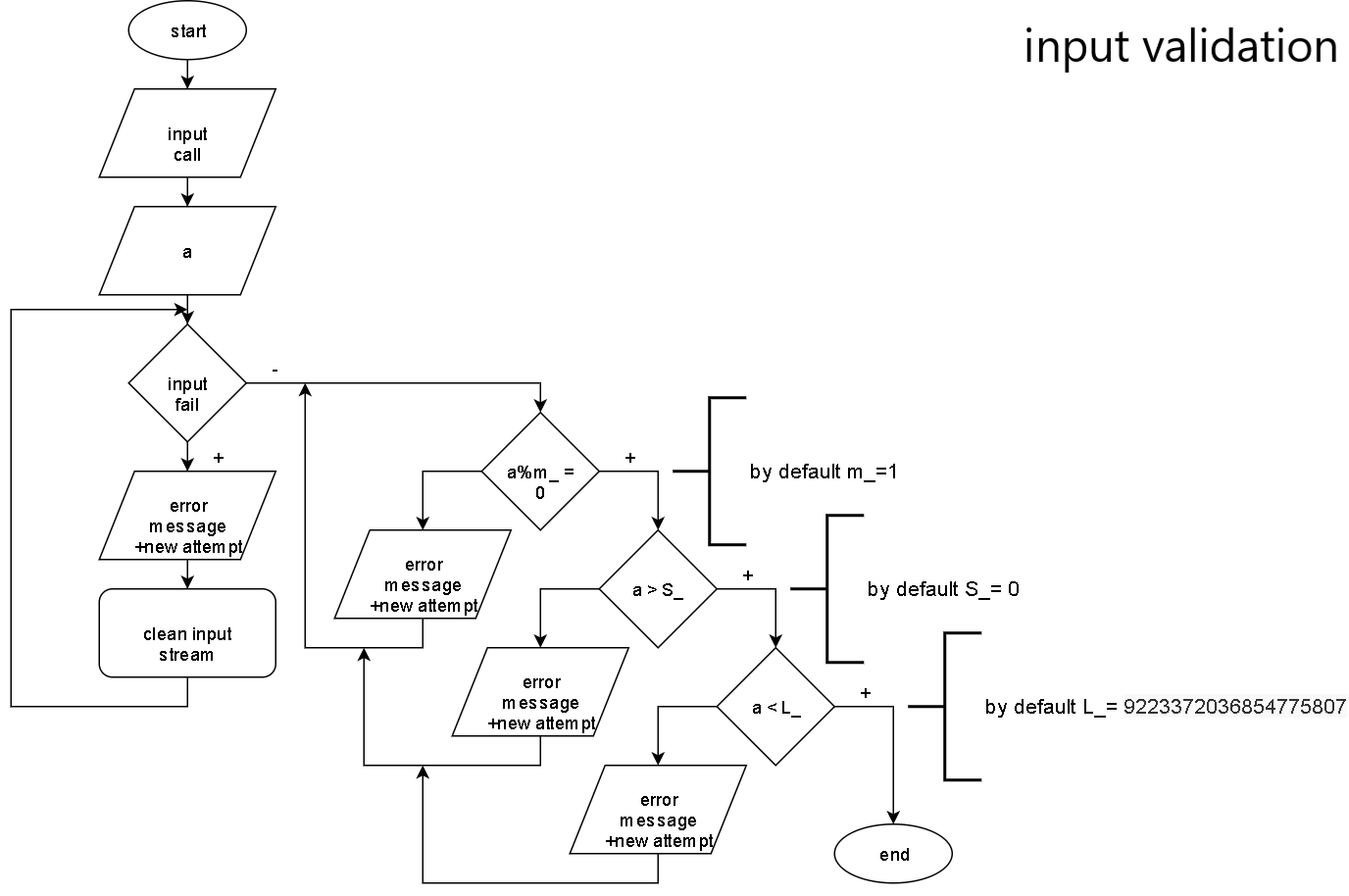


Рисунок 1

# **Алгоритм розв’язання задачі (блок-схема).**

Рисунок 2

Рисунок 3

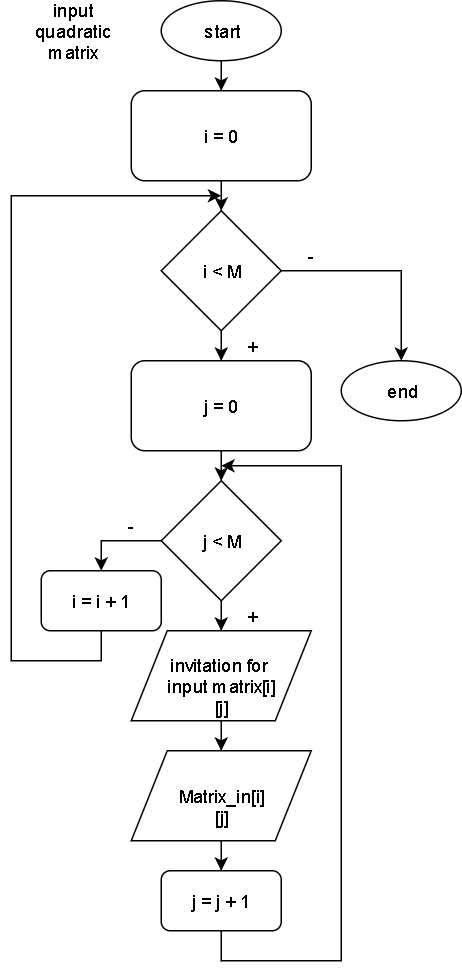


Рисунок 4

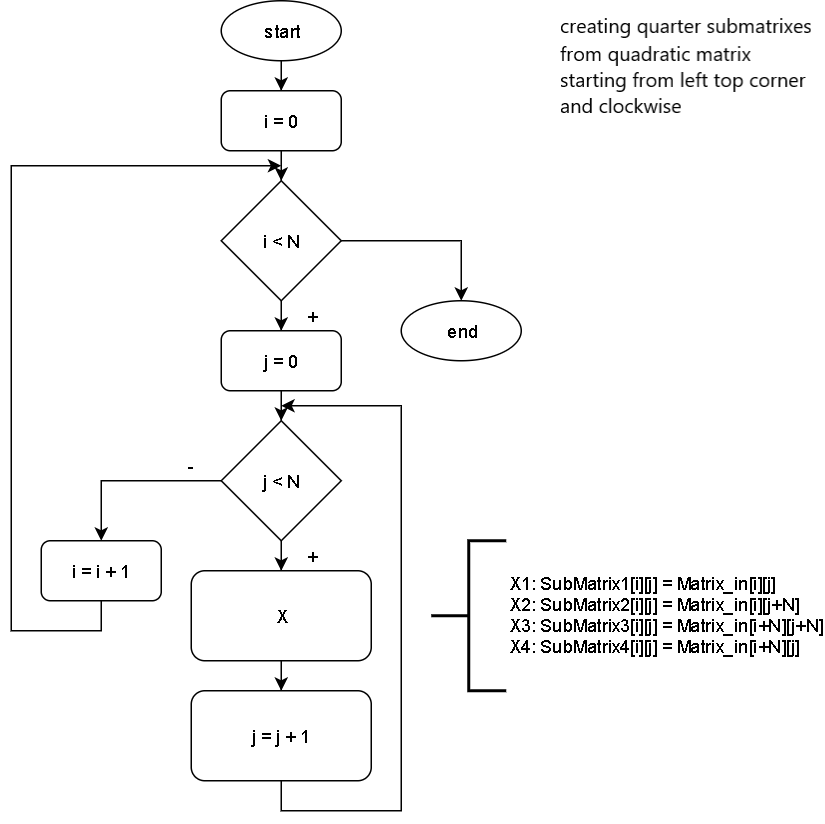


Рисунок 5

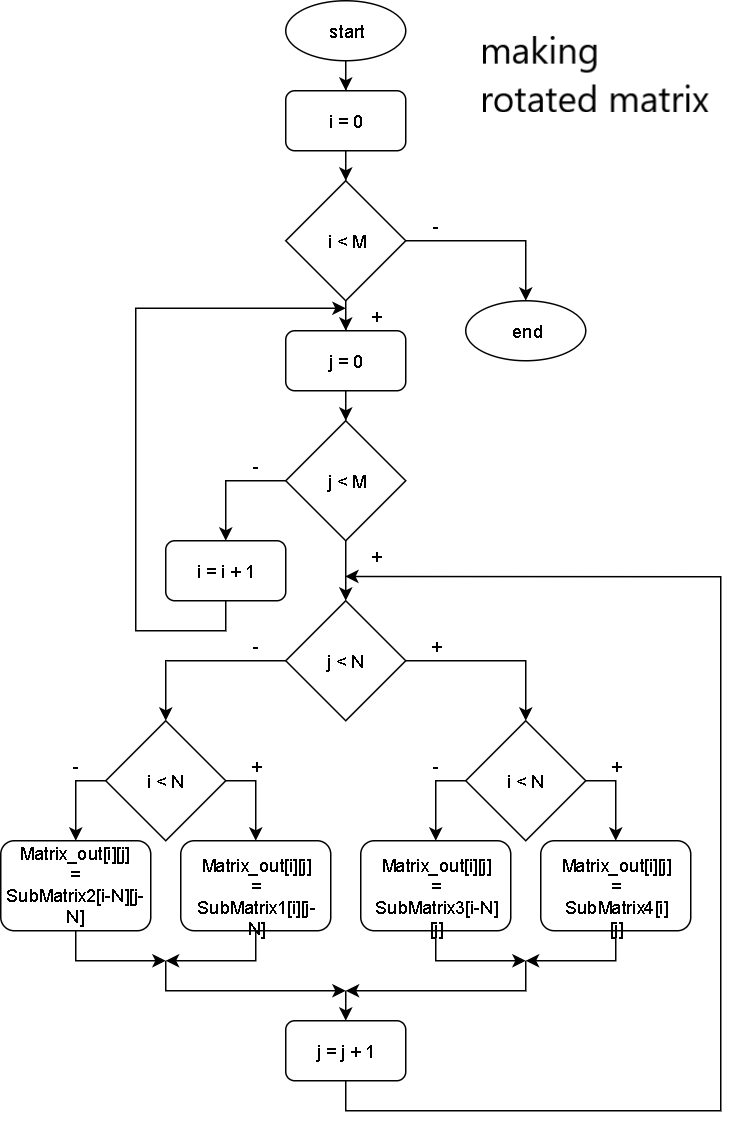


Рисунок 6

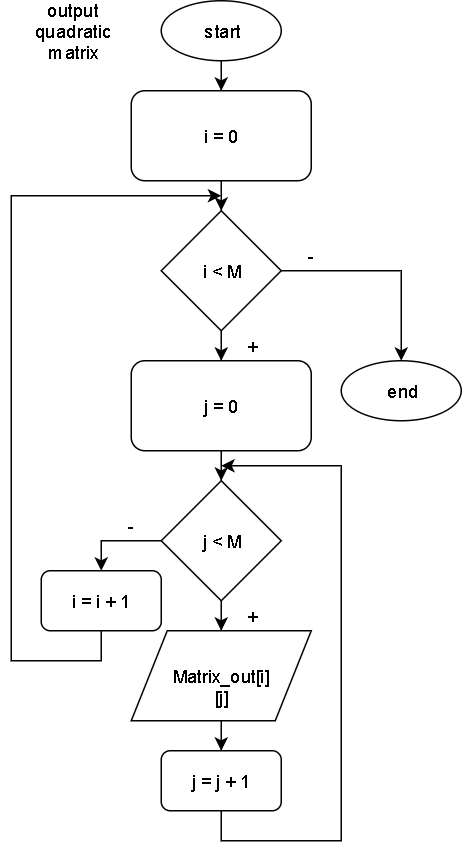
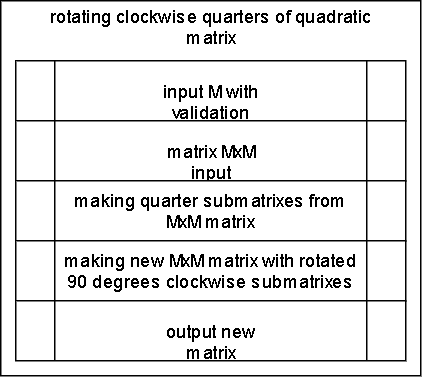
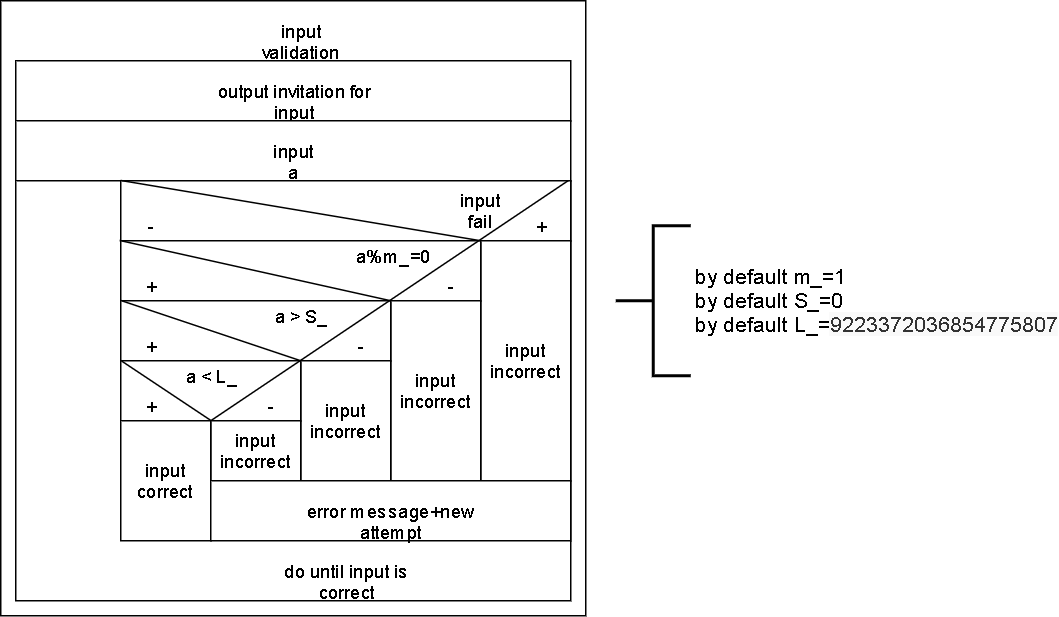


Рисунок 7

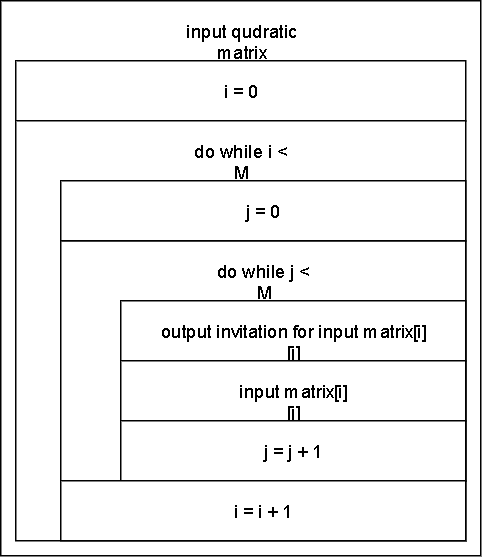
# **Алгоритм розв’язання задачі (діаграма Н-Ш).**



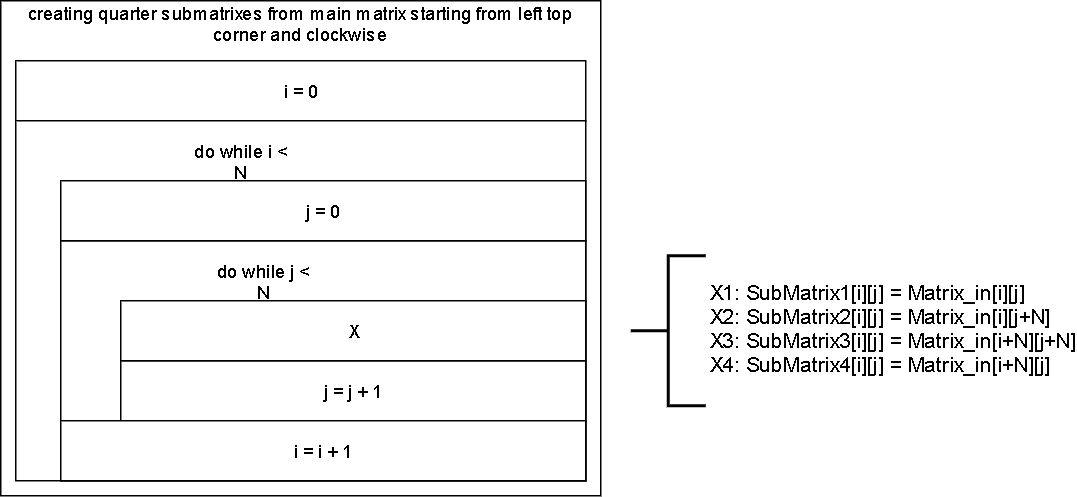
Рисунок



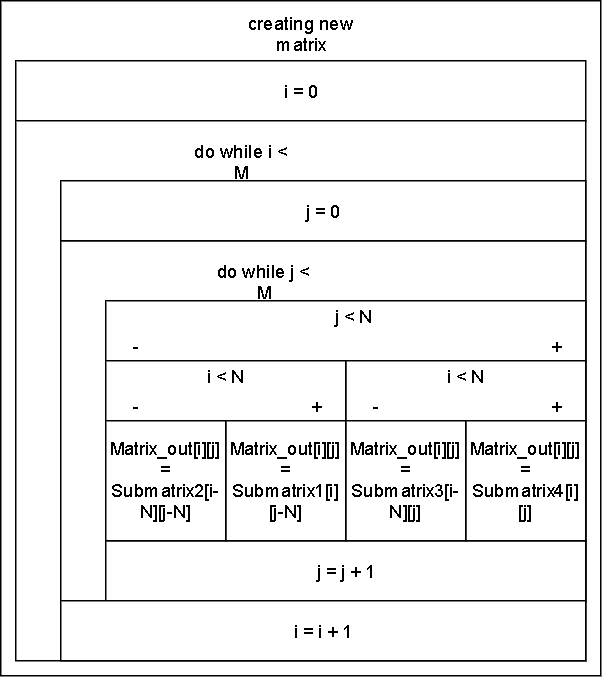
Рисунок



Рисунок



Рисунок



Рисунок

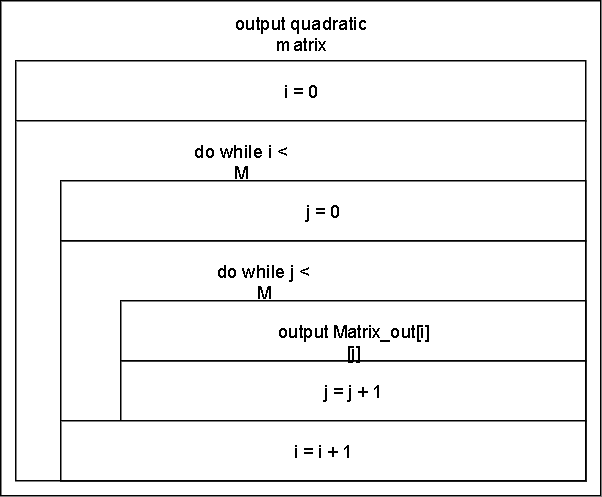


Рисунок 13

# **Набір тестів для перевірки правильності виконання завдання.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва тесту | Вхідні дані | | Очікувані результати |
|  |  | M | Matrix\_in[i][j] |  |
| 1 | Перевірка M%2 != 0 | 3 |  | Повідомлення про не коректні дані |
| 2 | M <= 1 | -10 |  | Повідомлення про не коректні дані |
| 3 | M – float | 6.32905 |  | Повідомлення про не коректні дані |
| 4 | Елемент матриці - символ | 2 | f | Повідомлення про не коректні дані |
| 5 | Перевірка матриці | 4 | {1} {2} {-1} {7}  {6} {12} {3} {34}  {-7} {0} {12} {1}  {0} {0} {2} {5} | {-7} {0} {1} {2}  {0} {0} {6} {12}  {12} {1} {-1} {7}  {2} {5} {3} {34} |

# **Текст програми.**

//application to rotate 90 degrees clockwise quarter submatrix of given qudratic matrix 2Nx2N

//in:

//number M(matrix order)

//Elements of Matrix MxM

//out:

//new matrix MxM with rotated 90 degrees clockwise quarter submatrixes

#include <iostream>

#include "inp\_val2.h"

#include <iomanip>

#include <stdlib.h>

int main()

{

std::cout << "Application takes input matrix MxM(M = 2N), and rotates its quarter submatrixes 90 degrees clockwise" << endl;

#pragma region M

int M;

char msg\_inviteM[8] = "input M";

char msg\_error[23] = "wrong input, try again";

M = cinum(msg\_inviteM, msg\_error, 2, 1, 2);

#pragma endregion input M with validation

#pragma region Matrix\_in\_0

double\*\* Matrix\_in = new double\* [M];

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

Matrix\_in[i] = new double[M];

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

Matrix\_in[i][j] = 0;

}

}

#pragma endregion setting 0 to starting matrix for future input

#pragma region Matrix\_in

char msg\_inviteij[21] = "input matrix element";

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

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

Matrix\_in[i][j] = cinum(msg\_inviteij, msg\_error, 1, 0, -214783648);

system("cls");

for (int i1 = 0; i1 < M; i1++) {

for (int j1 = 0; j1 < M; j1++) {

std::cout << std::setw(M) << Matrix\_in[i1][j1] << " ";

}

std::cout << std::endl;

}

}

}

#pragma endregion input matrix

#pragma region SubMatrix1

int N;

N = M / 2;

double\*\* SubMatrix1 = new double\* [N];

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

SubMatrix1[i] = new double[N];

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

SubMatrix1[i][j] = Matrix\_in[i][j];

}

}

#pragma endregion left top submatrix

#pragma region SubMatrix2

double\*\* SubMatrix2 = new double\* [N];

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

SubMatrix2[i] = new double[N];

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

SubMatrix2[i][j] = Matrix\_in[i][j + N];

}

}

#pragma endregion right top submatrix

#pragma region SubMatrix3

double\*\* SubMatrix3 = new double\* [N];

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

SubMatrix3[i] = new double[N];

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

SubMatrix3[i][j] = Matrix\_in[i + N][j + N];

}

}

#pragma endregion right bottom submatrix

#pragma region SubMatrix4

double\*\* SubMatrix4 = new double\* [N];

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

SubMatrix4[i] = new double[N];

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

SubMatrix4[i][j] = Matrix\_in[i + N][j];

}

}

#pragma endregion left bottom submatrix

#pragma region Matrix\_out

double\*\* Matrix\_out = new double\* [M];

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

Matrix\_out[i] = new double[M];

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

if (j < N) {

if (i < N) {

Matrix\_out[i][j] = SubMatrix4[i][j];

}

else {

Matrix\_out[i][j] = SubMatrix3[i - N][j];

}

}

else {

if (i < N) {

Matrix\_out[i][j] = SubMatrix1[i][j - N];

}

else {

Matrix\_out[i][j] = SubMatrix2[i - N][j - N];

}

}

}

}

#pragma endregion processed matrix assembly

#pragma region deleting SubMatrixes

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

delete[]SubMatrix1[i];

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

delete[]SubMatrix2[i];

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

delete[]SubMatrix3[i];

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

delete[]SubMatrix4[i];

#pragma endregion

#pragma region print Matrix\_out

system("cls");

std::cout << "given matrix:" << endl;

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

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

std::cout << std::setw(M) << Matrix\_in[i][j] << " ";

}

std::cout << std::endl;

}

std::cout << "rotated matrix:" << endl;

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

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

std::cout << std::setw(M) << Matrix\_out[i][j] << " ";

}

std::cout << std::endl;

}

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

delete[]Matrix\_in[i];

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

delete[]Matrix\_out[i];

#pragma endregion

system("pause");

return 0;

}

# **Inp\_val2.h**

//cinum - numerical only (R)

//char msg\_invite - your message for inviting user to input their number

//char msg\_error - if error input occures

//int m\_ - check if input is multiple of this number

//bool intFlag - =1(only integer input); =0(any input)

//double left - leftmost point in diapasone for input

//double right - rightmost point in diapasone for input

#include <iostream>

#include <windows.h>

#undef max

#include <cmath>

#include <climits>

using namespace std;

int cinum(char\* msg\_invite, char\* msg\_error, int m\_ = 1, bool intFlag = 0, float left = -2147483648.0, float right = 2147483647.0)

{

float aaa;

cout << msg\_invite << endl;

cin >> aaa;

int iaaa;

for (bool i = 0; i != 1;)

{

iaaa = (int)aaa;

if (cin.fail() || (aaa < left) || (aaa > right) || ((intFlag == 1) && (ceil(aaa) != floor(aaa))) || ((m\_ != 1) && ((iaaa % m\_) != 0))) {

cout << msg\_error << endl;

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin >> aaa;

}

else {

i = 1;

}

}

return (int)aaa;

}

# **Результати тестування програми та їх аналіз.**

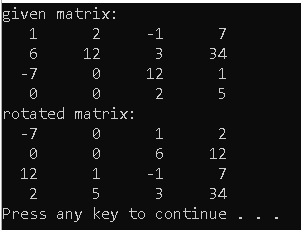


Рисунок 14

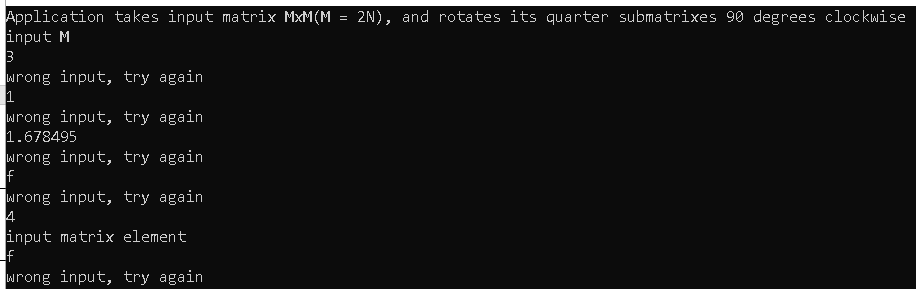


Рисунок 15

Результати відповідають очікуваним.

# **Висновки щодо способів завдання багатомірних масивів та особливостей розв’язання задач їх обробки.**

Багатомірні масиви описуються за допомогою вкладених циклів, які перебирають усі елементи. Так можна не тільки задавати стартові значення елементів, а й виводити масив на екран, маніпулювати цими значеннями, тощо.

У деякому сенсі багатомірні масиви та звичайні можна взаємно заміняти, алє перевага багатовимірних масивів у тому, що вони мають інтегровані багатовимірні «координати».

Найзручніше, для обробки багатомірних масивів, використовувати цикл з лічильником. Скільки вимірів, стільки й вкладених один в другий циклів, стільки ж лічильників, кожен з яких відповідає за свій вимір. Таким чином за допомогою багатомірних масивів можна працювати не тільки з матрицями(хоча й з ними можна багато зробити), а наприклад з багатовимірними просторами.