Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського"
Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Звіт

з лабораторної роботи № 1 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження алгоритмів розгалуження»

Варіант 17

Виконав студент ІП-12, Коновалюк Іванна Леонідівна

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота 9
Дослідження алгоритмів обходу масивів

Мета – дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Варіант 17.
Задано матрицю дійсних чисел A[n,n], ініціалізувати матрицю обходом по рядках. На побічній діагоналі матриці знайти перший додатний і останній від'ємний елементи, та поміняти їх місцями з елементами головній діагоналі.

Постановка задачі. Заданий алгоритм повинен ініціалізувати квадратну матрицю 5x5, знайти перший додатний і останній від'ємний елементи на побічній діагоналі матриці та поміняти їх місцями з елементами головної діагоналі.

Математична модель.

Змінна	Тип	Ім'я	Призначення
Розмірність матриці	Натуральний	n	Початкові дані
Рядок елемента в масиві	Натуральний	a	Проміжні дані
Стовпець елемента в масиві	Натуральний	b	Проміжні дані
Ітератор	Натуральний	i	Проміжні дані
Ітератор	Натуральний	j	Проміжні дані
Останнє від'ємне число побічної діагоналі	Дійсний	negative	Проміжні дані
Перше додатнє число побічної діагоналі	Дійсний	positive	Проміжні дані
Тимчасова змінна	Дійсний	t	Проміжні дані
Масив	Дійсний	matrix	Кінцеві дані

rand()%n – повертає, випатковим чином згенероване, ціле число.

Власні функції

inputMatrix(двовимірний дійсний масив) – створює двовимірний масив
firstChange(двовимірний дійсний масив) – знаходить перший додатній елемент побічної діагоналі та міняє місцями з елементом головної діагоналі
secondChange(двовимірний дійсний масив) знаходить останній від'ємний елемент побічної діагоналі та міняє місцями з елементом головної діагоналі
outputMatrix(двовимірний дійсний масив) – виводить двовимірний масив

Розв’язання.

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію визначення двовимірного масиву за допомогою функції.

Крок 3. Деталізуємо дію виведення двовимірного масиву за допомогою функції.

Крок 4. Деталізуємо дію знаходження першого додатнього елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі за допомогою функції.

Крок 5. Деталізуємо дію знаходження останнього від’ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі за допомогою функції.

Крок 6. Деталізуємо дію виведення відредагованої матриці за допомогою функції.

Псевдокод.

Крок 1.

Початок

n := 5

matrix[n][n]

Визначення двовимірного масиву

Виведення двовимірного масиву

Знаходження першого додатнього елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі

Знаходження останнього від’ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі

Виведення відредагованої матриці

Кінець

Крок 2.

Початок

n := 5

matrix[n][n]

inputMatrix(matrix)

Виведення двовимірного масиву

Знаходження першого додатнього елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі

Знаходження останнього від’ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі

Виведення відредагованої матриці

Кінець

Крок 3.

Початок

n := 5

matrix[n][n]

inputMatrix(matrix)

outputMatrix(matrix)

Знаходження першого додатнього елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі

Знаходження останнього від’ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі

Виведення відредагованої матриці

Кінець

Крок 4.

Початок

n := 5

matrix[n][n]

inputMatrix(matrix)

outputMatrix(matrix)

firstChange(matrix)

Знаходження останнього від’ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі

Виведення відредагованої матриці

Кінець

Крок 5.

Початок

n := 5

matrix[n][n]

inputMatrix(matrix)

outputMatrix(matrix)
firstChange(matrix)
secondChange(matrix)
Виведення відредагованої матриці

Кінець

Крок 6.

Початок

n := 5
matrix[n][n]
inputMatrix(matrix)
outputMatrix(matrix)
firstChange(matrix)
secondChange(matrix)
outputMatrix(matrix)

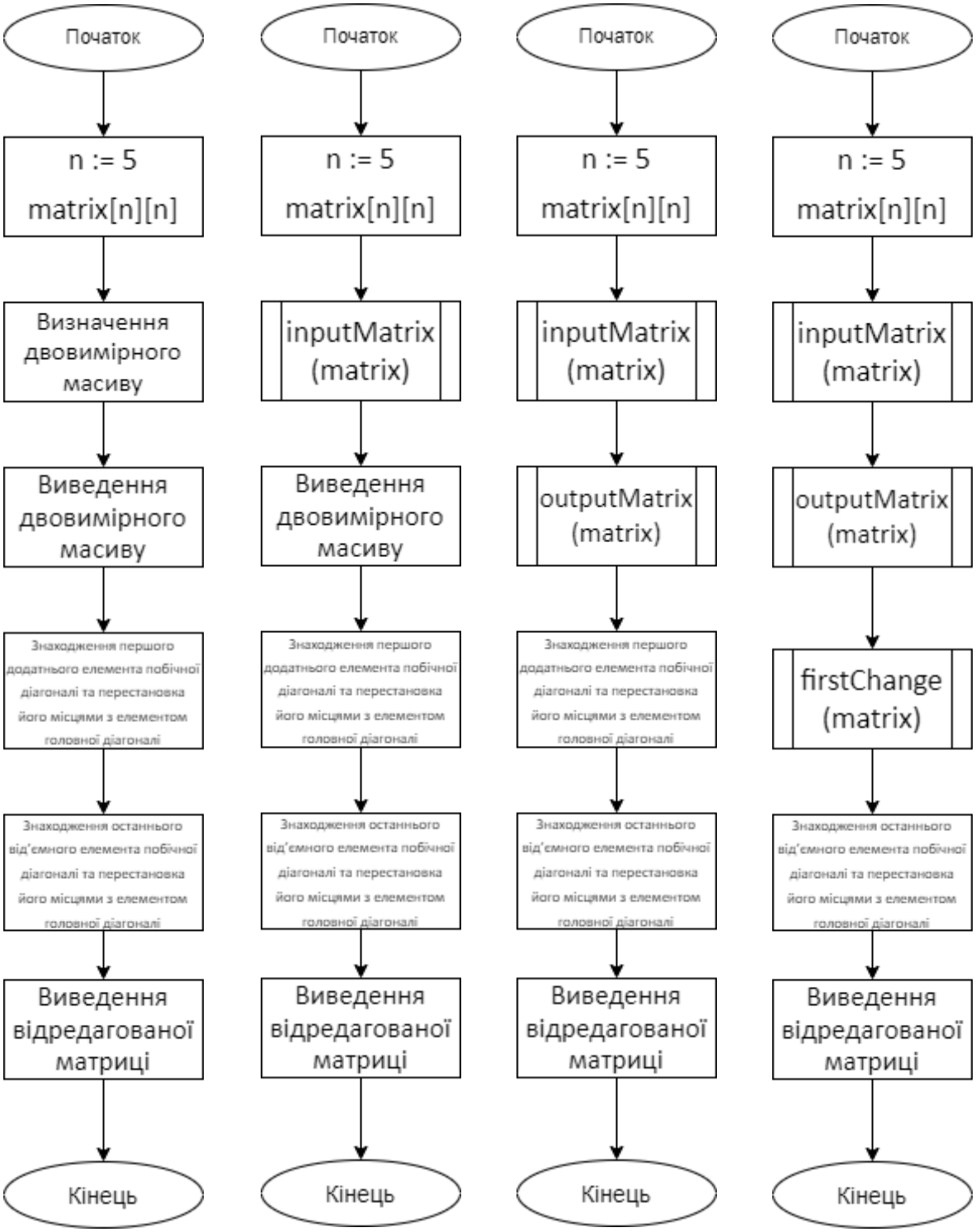
Кінець

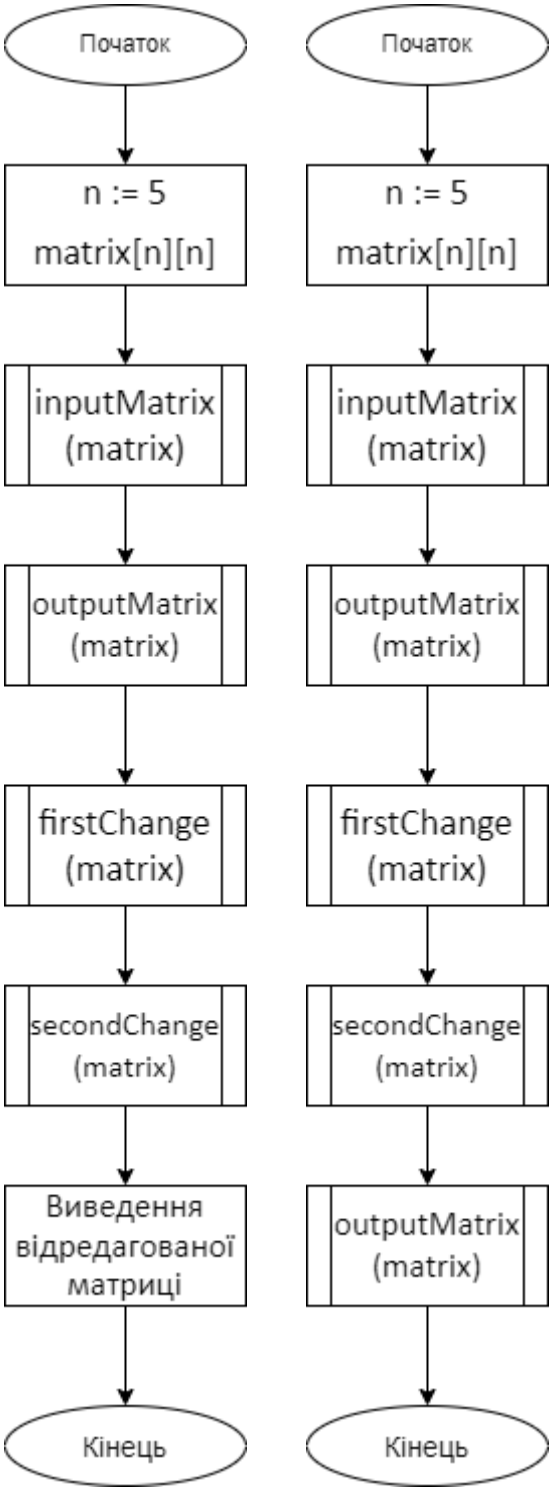
Підпрограма inputMatrix(matrix[n][n])
повторити
для і від 1 до n
повторити
для j від 1 до n
matrix[i][j] := (rand()%101-50)/10
все повторити
все повторити
Все підпрограма

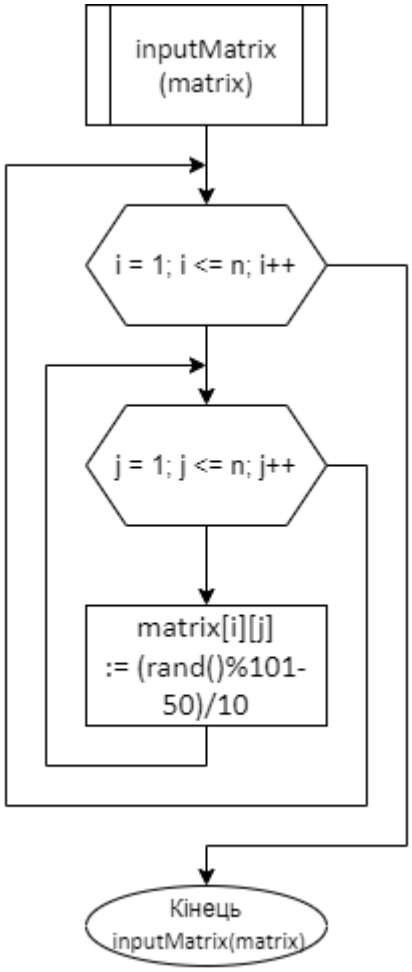
Підпрограма firstChange(matrix[n][n])
a := -1
b := -1
positive := -1
повторити
для і від 1 до n
якщо matrix[n – i + 1][i] > 0 **та** positive < 0
то
positive := matrix[n – i + 1][i]
a := n – i + 1
b := i
все якщо

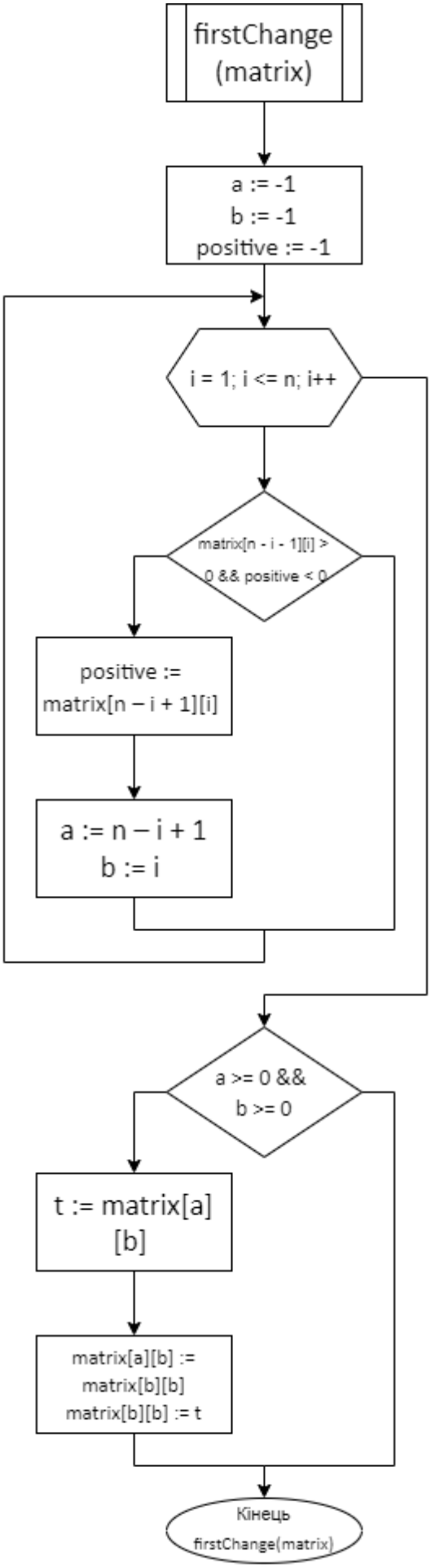
```
    все повторити
    якщо a >= 0 та b >=0
        то
            t := matrix[a][b]
            matrix[a][b] := matrix[b][b]
            matrix[b][b] := t
     все якщо
Все підпрограма

Підпрограма secondChange(matrix[n][n])
    a := -1
    b := -1
    negetive := 1
    повторити
    для і від 1 до n
        якщо matrix[i][n – i + 1] < 0 та negetive > 0
            то
                negetive := matrix[i][n – i + 1]
                a := i
                b := n – i + 1
        все якщо
     все повторити
     якщо a >= 0 та b >=0
         то
            t := matrix[a][b]
            matrix[a][b] := matrix[b][b]
            matrix[b][b] := t
     все якщо
Все підпрограма
```

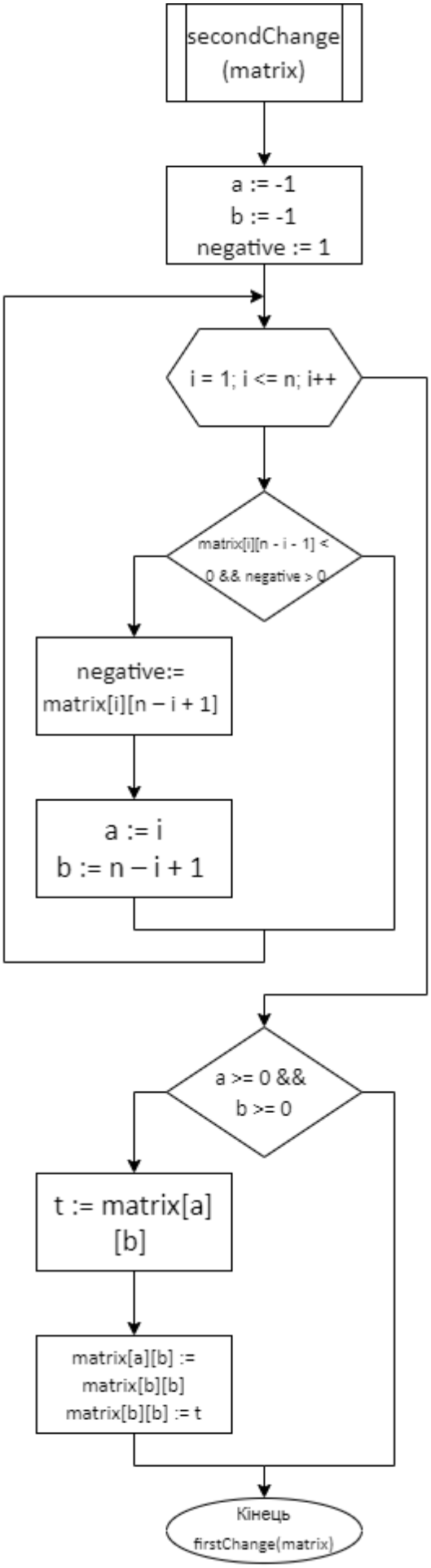
Блок-схеми.


Початок
n := 5
matrix[n][n]
Визначення двовимірного масиву
Виведення двовимірного масиву
Знаходження першого додатнього елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі
Знаходження останнього від'ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі
Виведення відредагованої матриці
Кінець
Початок
n := 5
matrix[n][n]
inputMatrix (matrix)
Виведення двовимірного масиву
Знаходження першого додатнього елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі
Знаходження останнього від'ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі
Виведення відредагованої матриці
Кінець
Початок
n := 5
matrix[n][n]
inputMatrix (matrix)
outputMatrix (matrix)
Знаходження першого додатнього елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі
Знаходження останнього від'ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі
Виведення відредагованої матриці
Кінець
Початок
n := 5
matrix[n][n]
inputMatrix (matrix)
outputMatrix (matrix)
firstChange (matrix)
Знаходження останнього від'ємного елемента побічної діагоналі та перестановка його місцями з елементом головної діагоналі
Виведення відредагованої матриці
Кінець


Початок
n := 5
matrix[n][n]
inputMatrix (matrix)
outputMatrix (matrix)
firstChange (matrix)
secondChange (matrix)
Виведення відредагованої матриці
Кінець
Початок
n := 5
matrix[n][n]
inputMatrix (matrix)
outputMatrix (matrix)
firstChange (matrix)
secondChange (matrix)
outputMatrix (matrix)
Кінець


inputMatrix (matrix)
i = 1; i <= n; i++
j = 1; j <= n; j++
matrix[i][j] := (rand()%101-50)/10
Кінець inputMatrix(matrix)


firstChange
(matrix)
a := -1
b := -1
positive := -1
i = 1; i <= n; i++
matrix[n - i - 1][i] >
0 && positive < 0
positive :=
matrix[n – i + 1][i]
a := n – i + 1
b := i
a >= 0 &&
b >= 0
t := matrix[a]
[b]
matrix[a][b] :=
matrix[b][b]
matrix[b][b] := t
Кінець
firstChange(matrix)


secondChange
(matrix)
a := -1
b := -1
negative := 1
i = 1; i <= n; i++
matrix[i][n - i - 1] <
0 && negative > 0
negative:=
matrix[i][n – i + 1]
a := i
b := n – i + 1
a >= 0 &&
b >= 0
t := matrix[a]
[b]
matrix[a][b] :=
matrix[b][b]
matrix[b][b] := t
Кінець
firstChange(matrix)

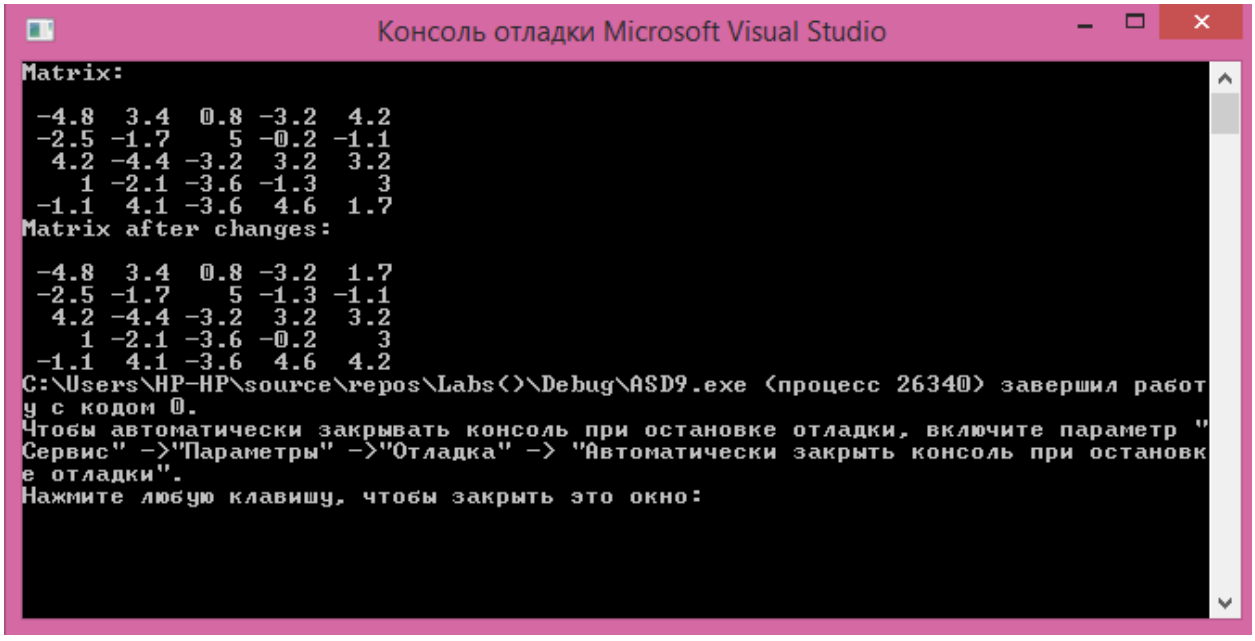
Код програми

```cpp
#include <iostream>
#include <iomanip>
using namespace std;

const int n = 5;

void inputMatrix(float[n][n]);
void firstChange(float[n][n]);
void secondChange(float[n][n]);
void outputMatrix(float[n][n]);

int main()
{
    srand(time(NULL));
    float matrix[n][n];
    inputMatrix(matrix);
    cout << "Matrix: " << "\n";
    outputMatrix(matrix);
    firstChange(matrix);
    secondChange(matrix);
    cout << "\n" << "Matrix after changes: " << "\n";
    outputMatrix(matrix);
}
```

```cpp
void inputMatrix(float matrix[n][n])
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            matrix[i][j] = (rand() % 101 - 50) / 10.0;
        }
    }
}

void firstChange(float matrix[n][n])
{
    int a = -1;
    int b = -1;
    float positive = -1;
    for (int i = 0; i < n; i++)
    {
        if (matrix[n - i - 1][i] > 0 && positive < 0)
        {
            positive = matrix[n - i - 1][i];
            a = n - i - 1;
            b = i;
        }
    }
    if (a >= 0 && b >= 0)
    {
        float t = matrix[a][b];
        matrix[a][b] = matrix[b][b];
        matrix[b][b] = t;
    }
}
```

```
void secondChange(float matrix[n][n])
{
    int a = -1;
    int b = -1;
    float negative = 1;
    for (int i = 0; i < n; i++)
    {
        if (matrix[i][n - i - 1] < 0 && negative > 0)
        {
            negative = matrix[i][n - i - 1];
            a = i;
            b = n - i - 1;
        }
    }
    if (a >= 0 && b >= 0)
    {
        float t = matrix[a][b];
        matrix[a][b] = matrix[b][b];
        matrix[b][b] = t;
    }
}

void outputMatrix(float matrix[n][n])
{
    for (int i = 0; i < n; i++)
    {
        cout << "\n";
        for (int j = 0; j < n; j++)
        {
            cout << setw(5) << matrix[i][j];
        }
    }
}
```

Випробуваня коду



Висновок. У результаті лабораторної роботи було дослідити алгоритми обходу масивів, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Було поставлено задачу, побудовано математичну модель, розроблено алгоритм її вирішення у вигляді псевдокоду, який було переведено на блок-схему. Алгоритм усмішно генерує двовимірний масив, знаходить перший додатний і останній від’ємний елементи на побічній діагоналі матриці та міняє їх місцями з елементами головної діагоналі.