

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

«Дослідження арифметичних циклічних алгоритмів»
Варіант 17

Виконав студент ІІ-13 Козак Антон Миколайович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська Анастасія Сергіївна
(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

№	Опис варіанту
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 головної діагоналі.

Постановка задачі

Заданий алгоритм повинен:

1. Створити двовимірний масив дійсних чисел розмірності $n \times n$.
2. Ініціювати його обходом по рядках.
3. Знайти перший додатний і останній від'ємний елементи масиву на побічній діагоналі.
4. Поміняти їх місцями з відповідними елементами головної діагоналі.

Побудова математичної моделі

Таблиця змінних

Змінна	Тип	Ім'я	Призначення
Розмірність масиву	Цілий	n	Початкові дані
Рядок елемента в масиві	Натуральний	row	Проміжні дані
Стовпець елемента в масиві	Натуральний	column	Проміжні дані
Перше додатне число побічної діагоналі	Дійсний	firstPositive	Проміжні дані
Останнє від'ємне число побічної діагоналі	Дійсний	lastNegative	Проміжні дані
Значення елемента масиву	Дійсний	temp	Проміжні дані
Масив дійсних чисел	Дійсний	matrix	Кінцеві дані

Використані функції

- `rand()%n` – повертає, випадковим чином згенероване, ціле число з діапазону $[0, n)$.
- `a%b` – повертає остачу від ділення числа a на число b .

Власні функції

- `generateMatrix`(двовимірний дійсний масив, ціле число) – повертає двовимірний масив дійсних чисел заданої розмірності ініційований обходом по рядках, кожен елемент якого згенерований випадковим чином.
 - `findFirstPositive`(двовимірний дійсний масив, ціле число) – повертає рядок та стовпець першого додатного елемента побічної діагоналі матриці.
 - `findLastNegative`(двовимірний дійсний масив, ціле число) – повертає рядок та стовпець останнього від'ємного елемента побічної діагоналі матриці.
 - `replaceWithDiagonal`(двовимірний дійсний масив, ціле число, ціле число, ціле число) – повертає двовимірний масив дійсних чисел заданої розмірності, у якому елемент з заданою позицією переставлений місцями з відповідним елементом головної діагоналі.
1. Згенеруємо елементи двовимірного масиву за допомогою функції `rand`: `matrix[i][j] = (rand()%201-100)/10`.
 2. Визначимо орієнтацію обходу рядка, перевіривши його на парність: `i%2!=0`.
 3. Знайдемо перший додатний елемент побічної діагоналі, починаючи пошук з лівого нижнього кута: `matrix[n-i+1][i]`
 4. Знайдемо останній від'ємний елемент побічної діагоналі, починаючи пошук з правого верхнього кута: `matrix[i][n-i+1]`
 5. Переставимо відшукані елементи з елементами головної діагоналі за допомогою тимчасової змінної: `temp=matrix[row][column]`.

Розв'язання

1. Визначимо основні дії.
2. Деталізуємо дію генерації масиву.
3. Деталізуємо дію пошуку першого додатного елемента побічної діагоналі.
4. Деталізуємо дію перестановки першого додатного елемента побічної діагоналі за допомогою умовної форми вибору.
5. Деталізуємо дію пошуку останнього від'ємного елемента побічної діагоналі.
6. Деталізуємо дію перестановки останнього від'ємного елемента побічної діагоналі за допомогою умовної форми вибору.
7. Деталізуємо дію генерації масива обходом по рядках за допомогою ітераційної форми повторення та альтернативної форми вибору.

8. Деталізуємо дію пошуку першого додатного елемента побічної діагоналі за допомогою ітераційної форми повторення та умовної форми вибору.
9. Деталізуємо дію пошуку останнього від'ємного елемента побічної діагоналі за допомогою ітераційної форми повторення та умовної форми вибору.
10. Деталізуємо дію перестановки елемента з відповідним елементом головної діагоналі.

Псевдокод алгоритму

Крок 1

Початок

Введення n

matrix[n][n]

Генерація масиву

Пошук першого додатного елемента побічної діагоналі

Перестановка першого додатного елемента побічної діагоналі

Пошук останнього від'ємного елемента побічної діагоналі

Перестановка останнього від'ємного елемента побічної діагоналі

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

Генерація масива обходом по рядках

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

Пошук першого додатного елемента побічної діагоналі

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

Пошук останнього від'ємного елемента побічної діагоналі

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 2

Початок

Введення n

`matrix[n][n]`

`generateMatrix(matrix, n)`

Пошук першого додатного елемента побічної діагоналі

Перестановка першого додатного елемента побічної діагоналі

Пошук останнього від'ємного елемента побічної діагоналі

Перестановка останнього від'ємного елемента побічної діагоналі

Виведення matrix

Кінець

Підпрограма

`generateMatrix(matrix, n)`

Генерація масива обходом по рядках

Все підпрограма

Підпрограма

`findFirstPositive (matrix, n)`

Пошук першого додатного елемента побічної діагоналі

Все підпрограма

Підпрограма

`findLastNegative (matrix, n)`

Пошук останнього від'ємного елемента побічної діагоналі

Все підпрограма

Підпрограма

`replaceWithDiagonal(matrix, row, column, n)`

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 3

Початок

Введення n

matrix[n][n]

generateMatrix(matrix, n)

row, column = findFirstPositive(matrix, n)

Перестановка першого додатного елемента побічної діагоналі

Пошук останнього від'ємного елемента побічної діагоналі

Перестановка останнього від'ємного елемента побічної діагоналі

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

Генерація масива обходом по рядках

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

Пошук першого додатного елемента побічної діагоналі

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

Пошук останнього від'ємного елемента побічної діагоналі

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 4

Початок

Введення n

matrix[n][n]

generateMatrix(matrix, n)

row, column = findFirstPositive(matrix, n)

якщо row>0 та column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

Пошук останнього від'ємного елемента побічної діагоналі

Перестановка останнього від'ємного елемента побічної діагоналі

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

Генерація масива обходом по рядках

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

Пошук першого додатного елемента побічної діагоналі

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

Пошук останнього від'ємного елемента побічної діагоналі

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 5

Початок

Введення n

matrix[n][n]

generateMatrix(matrix, n)

row, column = findFirstPositive(matrix, n)

якщо row>0 **та** column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

row, column = findLastPositive(matrix, n)

Перестановка останнього від'ємного елемента побічної діагоналі

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

Генерація масива обходом по рядках

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

Пошук першого додатного елемента побічної діагоналі

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

Пошук останнього від'ємного елемента побічної діагоналі

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 6

Початок

Введення n

matrix[n][n]

generateMatrix(matrix, n)

row, column = findFirstPositive(matrix, n)

якщо row>0 **та** column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

row, column = findLastPositive(matrix, n)

якщо row>0 **та** column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

Генерація масива обходом по рядках

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

Пошук першого додатного елемента побічної діагоналі

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

Пошук останнього від'ємного елемента побічної діагоналі

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 7

Початок

Введення n

matrix[n][n]

generateMatrix(matrix, n)

row, column = findFirstPositive(matrix, n)

якщо row>0 **та** column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

row, column = findLastPositive(matrix, n)

якщо row>0 **та** column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

повторити

для i від 1 до n

якщо i%2!=0

то

повторити

для j від 1 до n

matrix[i][j]=(rand()%201-100)/10

все повторити

інакше

повторити

для j від n до 1

matrix[i][j]=(rand()%201-100)/10

все повторити

все якщо

все повторити

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

Пошук першого додатного елемента побічної діагоналі

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

Пошук останнього від'ємного елемента побічної діагоналі

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 8

Початок

Введення n

matrix[n][n]

generateMatrix(matrix, n)

row, column = findFirstPositive(matrix, n)

якщо row>0 **та** column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

row, column = findLastPositive(matrix, n)

якщо row>0 **та** column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

повторити

для i **від** 1 **до** n

якщо i%2!=0

то

повторити

для j **від** 1 **до** n

matrix[i][j]=(rand()%201-100)/10

все повторити

інакше

повторити

для j **від** n **до** 1

matrix[i][j]=(rand()%201-100)/10

все повторити

все якщо

все повторити

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

row=-1

column=-1

firstPositive=-1

повторити

для i від 1 до n

якщо matrix[n-i+1][i]>0 та firstPositive<0

то

firstPositive=matrix[n-i+1][i]

row=n-i+1

column=i

все якщо

все повторити

повернути row, column

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

Пошук останнього від'ємного елемента побічної діагоналі

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 9

Початок

Введення n

matrix[n][n]

generateMatrix(matrix, n)

row, column = findFirstPositive(matrix, n)

якщо row>0 та column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

row, column = findLastPositive(matrix, n)

якщо row>0 та column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

повторити

для i від 1 до n

якщо i%2!=0

то

повторити

для j від 1 до n

matrix[i][j]=(rand()%201-100)/10

все повторити

інакше

повторити

для j від n до 1

matrix[i][j]=(rand()%201-100)/10

все повторити

все якщо

все повторити

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

row=-1

column=-1

firstPositive=-1

повторити

для i від 1 до n

якщо matrix[n-i+1][i]>0 та firstPositive<0

то

firstPositive=matrix[n-i+1][i]

row=n-i+1

column=i

все якщо

все повторити

повернути row, column

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

row=-1

column=-1

firstPositive=1

повторити

для i від 1 до n

якщо matrix[i][n-i+1]<0 **та** firstPositive>0

то

firstPositive= matrix[i][n-i+1]<

row=i

column=n-i+1

все якщо

все повторити

повернути row, column

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

Перестановка елемента з відповідним елементом головної діагоналі

Все підпрограма

Крок 10

Початок

Введення n

matrix[n][n]

generateMatrix(matrix, n)

row, column = findFirstPositive(matrix, n)

якщо row>0 та column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

row, column = findLastPositive(matrix, n)

якщо row>0 та column>0

то

replaceWithDiagonal(matrix, row, column, n)

все якщо

Виведення matrix

Кінець

Підпрограма

generateMatrix(matrix, n)

повторити

для i від 1 до n

якщо i%2!=0

то

повторити

для j від 1 до n

matrix[i][j]=(rand()%201-100)/10

все повторити

інакше

повторити

для j від n до 1

matrix[i][j]=(rand()%201-100)/10

все повторити

все якщо

все повторити

Все підпрограма

Підпрограма

findFirstPositive (matrix, n)

row=-1

column=-1

firstPositive=-1

повторити

для i від 1 до n

якщо matrix[n-i+1][i]>0 **та** firstPositive<0

то

firstPositive=matrix[n-i+1][i]

row=n-i+1

column=i

все якщо

все повторити

повернути row, column

Все підпрограма

Підпрограма

findLastNegative (matrix, n)

row=-1

column=-1

firstPositive=1

повторити

для i від 1 до n

якщо matrix[i][n-i+1]<0 **та** firstPositive>0

то

firstPositive= matrix[i][n-i+1]

row=i

column=n-i+1

все якщо

все повторити

повернути row, column

Все підпрограма

Підпрограма

replaceWithDiagonal(matrix, row, column, n)

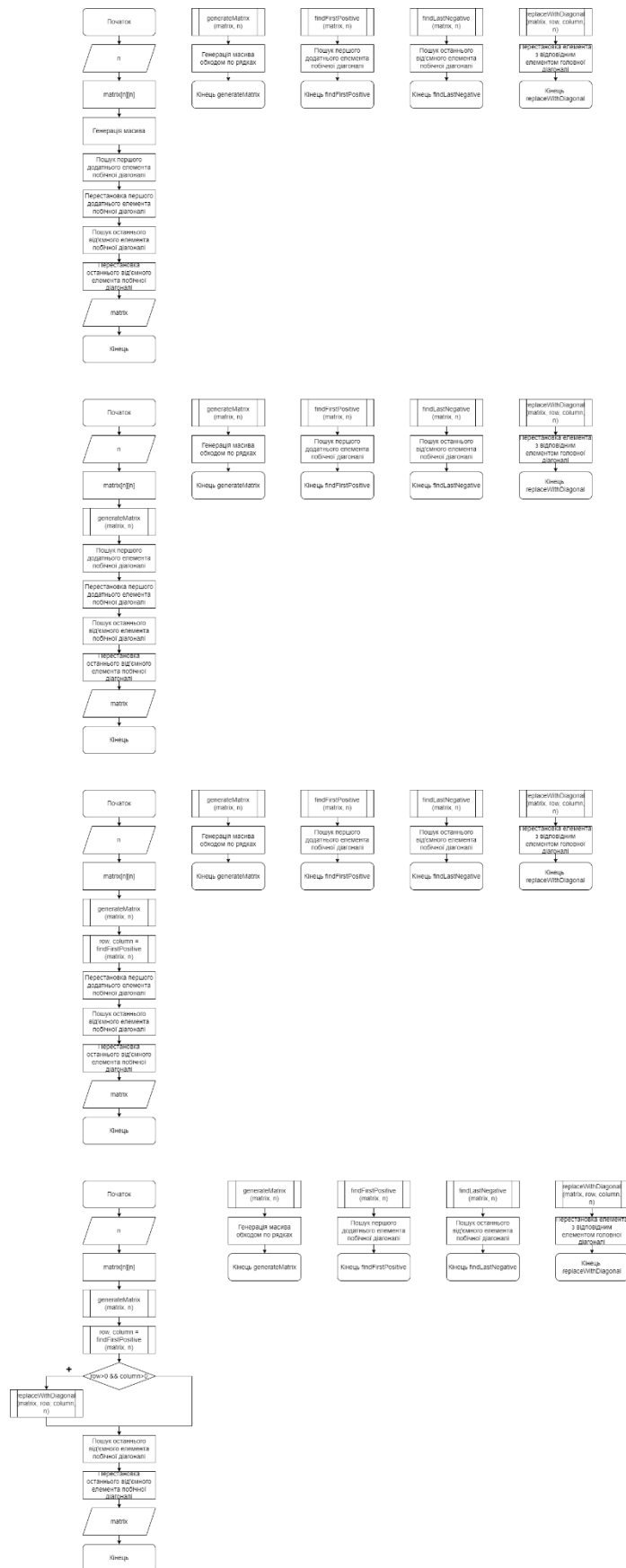
temp=matrix[row][column]

matrix[row][column]=matrix[column][column]

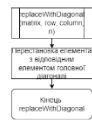
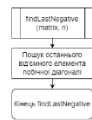
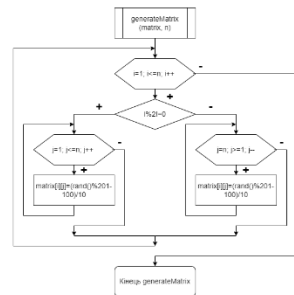
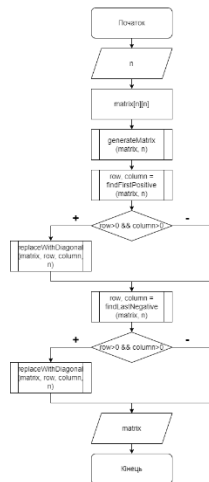
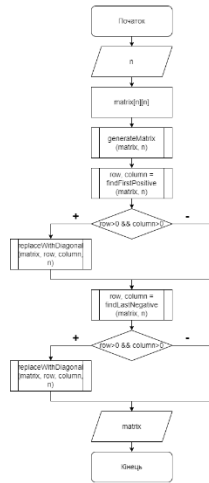
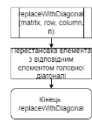
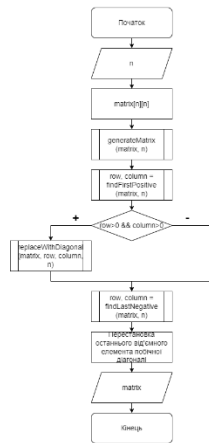
matrix[column][column]=temp

Все підпрограма

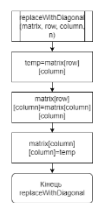
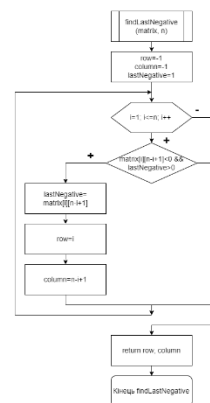
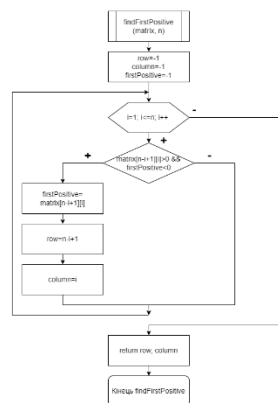
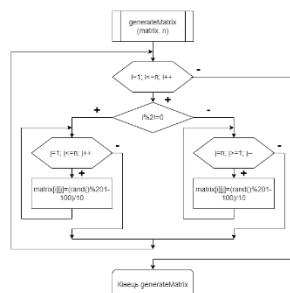
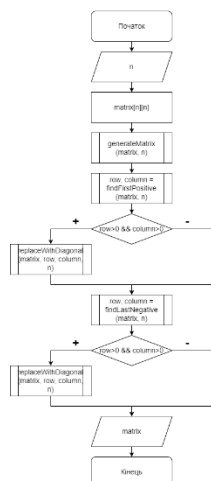
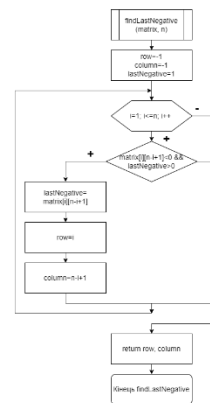
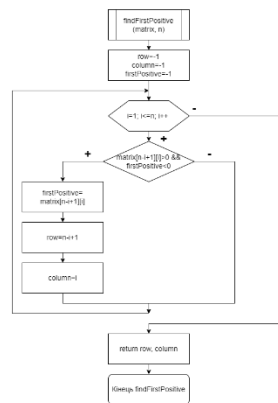
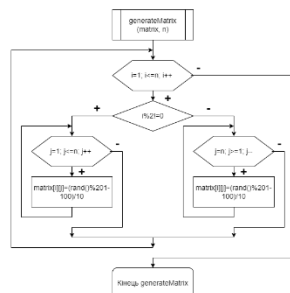
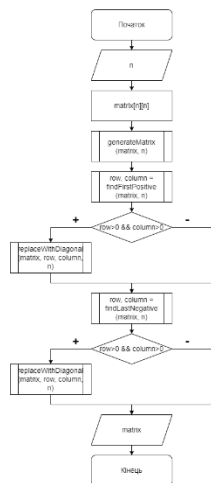
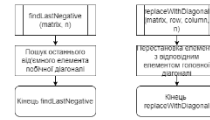
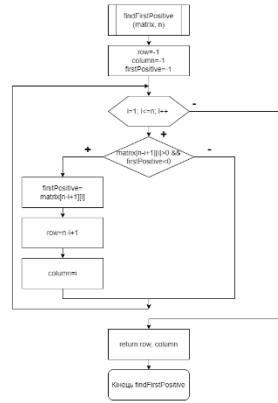
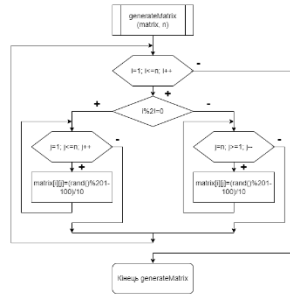
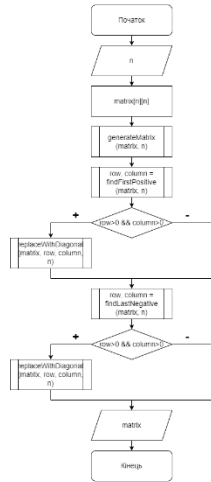
Блок-схема алгоритму



Основи програмування – 1. Алгоритми та структури



Основи програмування – 1. Алгоритми та структури



Код програми

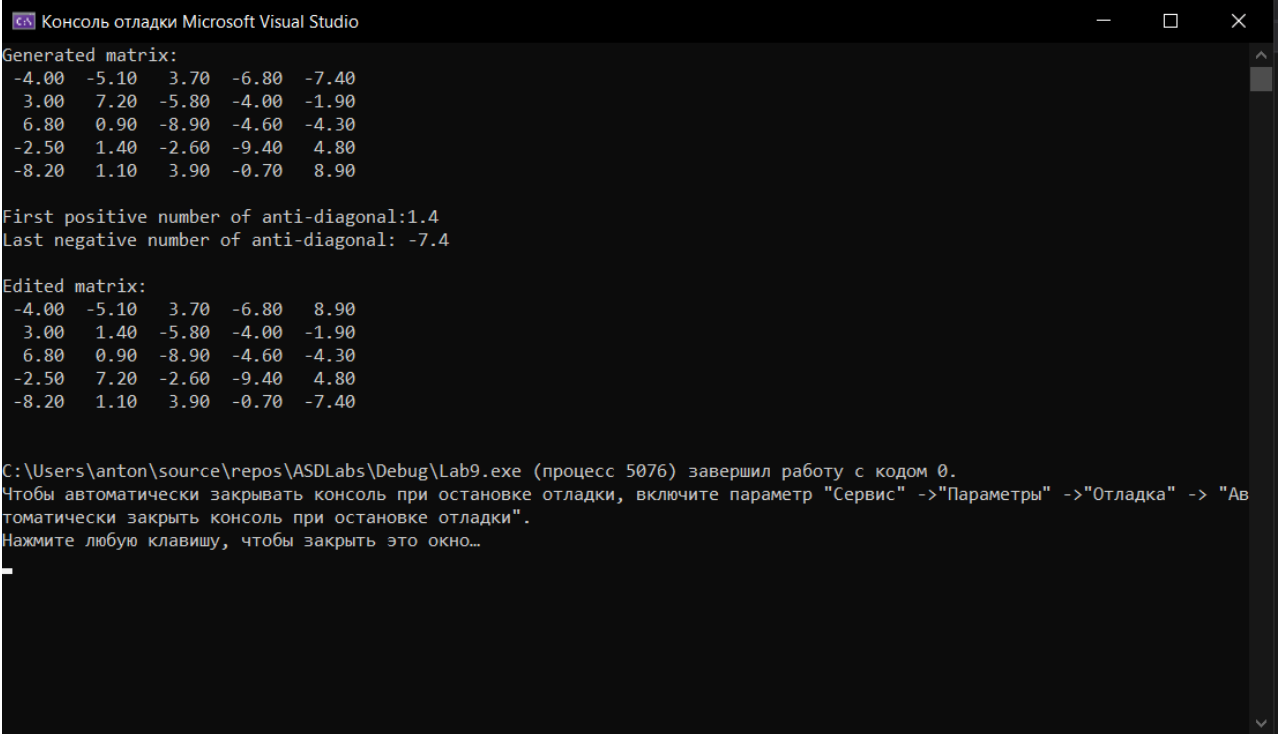
```

1  /* Козак Антон ІП-13
2  Варіант 17
3  Задано матрицю дійсних чисел A[n,n], ініціалізувати матрицю обходом порядках.
4  На побічній діагоналі матриці знайти перший додатний і останній від'ємний елементи, та поміняти їх місцями з елементами головної діагоналі.*/
5
6  #include <iostream>
7  using namespace std;
8
9  const int n = 5; //розмірність двовимірного масиву
10
11 void generateMatrix(float matrix[n][n]); //генерує двовимірний масив
12 void findFirstPositive(float matrix[n][n], int& row, int& column); //знаходиться позицію першого додатного елемента на побічній діагоналі
13 void findLastNegative(float matrix[n][n], int& row, int& column); //знаходиться позицію останнього від'ємного елемента на побічній діагоналі
14 void replaceWithDiagonal(float matrix[n][n], int row, int column); //переставляє елемент з відповідним елементом головної діагоналі
15 void outputMatrix(float matrix[n][n]); //виводить двовимірний масив
16
17 int main()
18 {
19     srand(time(NULL));
20     float matrix[n][n]; //двовимірний масив
21     int row, column; //позиція елемента в матриці
22
23     //генерація та виведення масиву
24     generateMatrix(matrix);
25     cout << "Generated matrix: " << "\n";
26     outputMatrix(matrix);
27
28     //знаходження та перестановка першого додатного елемента
29     findFirstPositive(matrix, row, column);
30     if (row >= 0 && column >= 0)
31         replaceWithDiagonal(matrix, row, column);
32
33     //знаходження та перестановка останнього від'ємного елемента
34     findLastNegative(matrix, row, column);
35     if (row >= 0 && column >= 0)
36         replaceWithDiagonal(matrix, row, column);
37
38     //виведення відредагованого масиву
39     cout << "\n" << "Edited matrix: " << "\n";
40     outputMatrix(matrix);
41 }
42
43 void generateMatrix(float matrix[n][n])
44 {
45     for (int i = 0; i < n; i++) {
46         if (i % 2 == 0) {
47             for (int j = 0; j < n; j++) {
48                 matrix[i][j] = (-100 + rand() % 201) / 10.0;
49             }
50         }
51         else {
52             for (int j = n-1; j >= 0; j--) {
53                 matrix[i][j] = (-100 + rand() % 201) / 10.0;
54             }
55         }
56     }
57 }
58
59 void outputMatrix(float matrix[n][n])
60 {
61     for (int i = 0; i < n; i++) {
62         for (int j = 0; j < n; j++) {
63             printf("%.2f ", matrix[i][j]);
64         }
65         cout << "\n";
66     }
67     cout << "\n";
68 }
69
70 void findFirstPositive(float matrix[n][n], int& row, int& column)
71 {
72     row = -1;
73     column = -1;
74     float firstPositive = -1;
75
76     for (int i = 0; i < n; i++) {
77         if (matrix[n - i - 1][i] > 0 && firstPositive < 0) {
78             firstPositive = matrix[n - i - 1][i];
79             row = n - i - 1;
80             column = i;
81         }
82     }
83
84     if (firstPositive > 0)
85         cout << "First positive number of anti-diagonal: " << firstPositive << "\n";
86     else
87         cout << "Anti-diagonal doesn't have any positive numbers" << "\n";
88 }
89
90

```

```
91 void findLastNegative(float matrix[n][n], int & row, int & column)
92 {
93     row = -1;
94     column = -1;
95     float lastNegative = 1;
96
97     for (int i = 0; i < n; i++) {
98         if (matrix[i][n - i - 1] < 0 && lastNegative > 0) {
99             lastNegative = matrix[i][n - i - 1];
100             row = i;
101             column = n - i - 1;
102         }
103     }
104
105     if (lastNegative < 0)
106         cout << "Last negative number of anti-diagonal: " << lastNegative << "\n";
107     else
108         cout << "Anti-diagonal doesn't have any negative numbers" << "\n";
109 }
110
111 void replaceWithDiagonal(float matrix[n][n], int row, int column)
112 {
113     float temp = matrix[row][column];
114     matrix[row][column] = matrix[column][column];
115     matrix[column][column] = temp;
116 }
```

Тестування програми

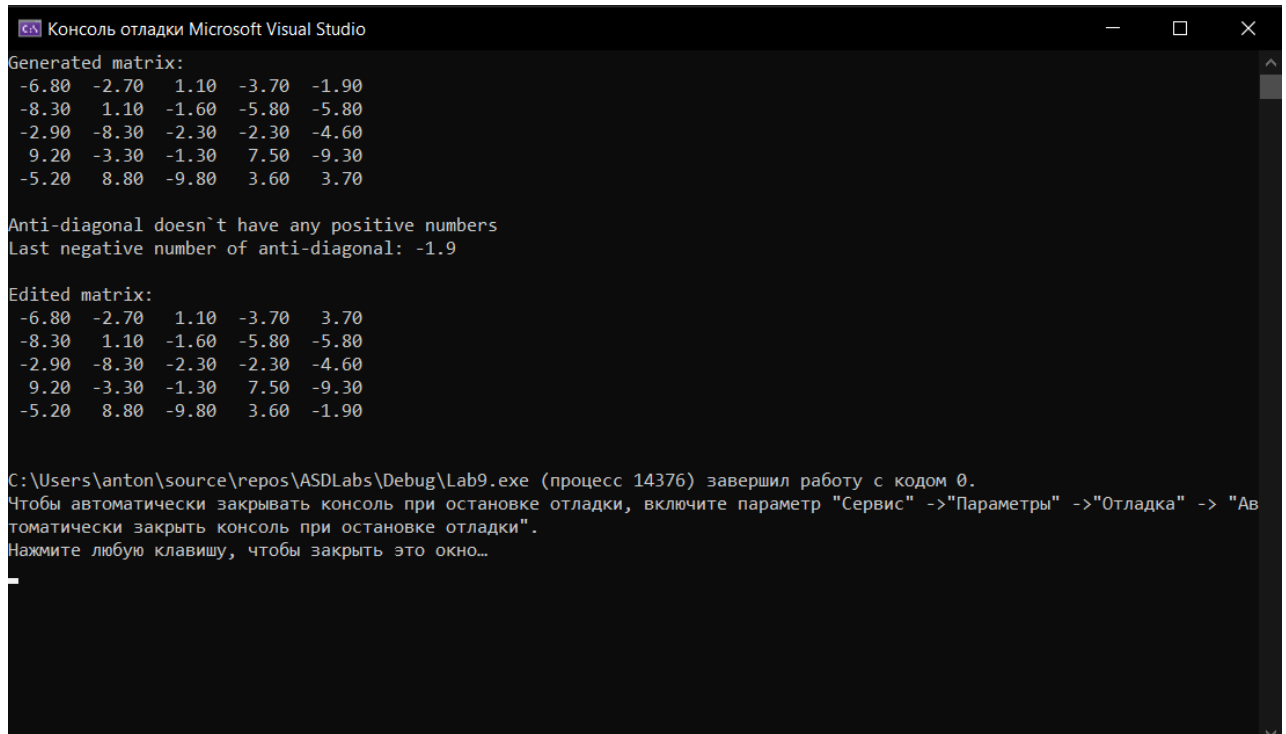


```
Консоль отладки Microsoft Visual Studio
Generated matrix:
-4.00 -5.10 3.70 -6.80 -7.40
3.00 7.20 -5.80 -4.00 -1.90
6.80 0.90 -8.90 -4.60 -4.30
-2.50 1.40 -2.60 -9.40 4.80
-8.20 1.10 3.90 -0.70 8.90

First positive number of anti-diagonal:1.4
Last negative number of anti-diagonal: -7.4

Edited matrix:
-4.00 -5.10 3.70 -6.80 8.90
3.00 1.40 -5.80 -4.00 -1.90
6.80 0.90 -8.90 -4.60 -4.30
-2.50 7.20 -2.60 -9.40 4.80
-8.20 1.10 3.90 -0.70 -7.40

C:\Users\anton\source\repos\ASDLabs\Debug\Lab9.exe (процесс 5076) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```



```
Консоль отладки Microsoft Visual Studio

Generated matrix:
-6.80 -2.70 1.10 -3.70 -1.90
-8.30 1.10 -1.60 -5.80 -5.80
-2.90 -8.30 -2.30 -2.30 -4.60
9.20 -3.30 -1.30 7.50 -9.30
-5.20 8.80 -9.80 3.60 3.70

Anti-diagonal doesn't have any positive numbers
Last negative number of anti-diagonal: -1.9

Edited matrix:
-6.80 -2.70 1.10 -3.70 3.70
-8.30 1.10 -1.60 -5.80 -5.80
-2.90 -8.30 -2.30 -2.30 -4.60
9.20 -3.30 -1.30 7.50 -9.30
-5.20 8.80 -9.80 3.60 -1.90

C:\Users\anton\source\repos\ASDLabs\Debug\Lab9.exe (процесс 14376) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" ->"Параметры" ->"Отладка" -> "Автоматически закрыть консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Висновки

Протягом виконання цієї лабораторної роботи я набув навичок використання алгоритмів обходу масивів під час складання програмних специфікацій. Маючи розмірність та тип масиву, я склав алгоритм, який успішно генерує заданий масив обходом по рядках, знаходить перший додатний та останній від'ємний елементи його побічної діагоналі та переставляє їх місцями з відповідними елементами головної діагоналі.