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 алгоритмів обходу масивів»

Варіант 22

Виконав студент ІП-13, Музичук Віталій Андрійович
(шифр, прізвище, ім'я, по батькові)

Перевірила Вєчерковська 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 22

Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису змінної індексованого типу (двовимірного масиву) згідно з варіантом.

2. Ініціювання змінної, що описана в п.1 даного завдання.

3. Обчислення змінної, що описана в п.1, згідно з варіантом.

22	Задано матрицю дійсних чисел A[m,n]. У кожному стовпчику матриці знайти останній додатний елемент X і його місцезнаходження. Обміняти знайдене значення X з елементом середнього рядка.

Постановка задачі

Спочатку ми створюємо двовимірний масив із заданою кількістю рядків і стовпців. Ініціалізація цієї змінної відбувається випадково числами з проміжку [-9, 9] за допомогою підпрограми initMatrix(), при цьому обходимо матрицю «змійкою» по стовпцях. Далі за допомогою підпрограми outMatrix() виводимо матрицю в консоль для наглядності. Після цього за допомогою підпрограми operarion() виводимо в консоль останні додатні елементи і їх положення в матриці. Далі ще раз виводимо всю матрицю щоб побачити зміни в положенні додатних елементів та середніх елементів рядка.

Побудова математичної моделі

Складемо таблицю змінних

Змінна	Тип	Ім’я	Призначення
Двовимірний масив	Цілий	*matrix*	Проміжне значення
Кількість рядків	Цілий	*rows*	Початкове значення

Кількість стовпців	Цілий	*columns*	Початкове значення
Підпрограма для ініціалізації матриці	void	*initMatrix*	Ініціалізація матриці
Підпрограма для виведення матриці	void	*outMatrix*	Виведення матриці в консоль
Підпрограма для обробки матриці за варіантом	Дійсний	*operation()*	Обробка матриці за варіантом

initMatrix – підпрограма яка заповнює матрицю за допомогою проходу «змійкою» по стовпцях випадковими числами з проміжку [-9; 9].

outMatrix – підпрограма, яка поелементно виводить в консоль дані з матриці за допомогою 2 ітераційних циклів.

operation() - підпрограма яка виводить в консоль значення та позицію останнього додатного елемента кожного стовпця та міняє їх з середнім елементом рядків.

Rand(a, b) – функція яка генерує випадкове число на проміжку [a, b]

Розв’язання

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначаємо основні дії
Крок 2. Введення даних
Крок 3. Створення й ініціалізація матриці
Крок 4. Виведення матриці
Крок 5. Обробка матриці

Псевдокод

крок 1
початок

введення даних
створення й ініціалізація матриці
виведення матриці
обробка матриці

кінець

крок 2

початок

введення rows, columns
створення й ініціалізація матриці
виведення матриці
обробка матриці

кінець

крок 3

початок

введення rows, columns
matrix[rows, columns]
initMatrix(matrix, rows, columns)
виведення матриці
обробка матриці

кінець

крок 4

початок

введення rows, columns
matrix[rows, columns]
initMatrix(matrix, rows, columns)
outMatrix(matrix, rows, columns
обробка матриці

кінець

крок 5

початок

введення rows, columns
matrix[rows, columns]
initMatrix(matrix, rows, columns)
outMatrix(matrix, rows, columns)
operation(matrix, rows, columns)
outMatrix(matrix, rows, columns)

кінець

Псевдокод підпрограм:

initMatrix(matrix, rows, columns**)**

```
k = 1
для j від 1 до columns повторити
    якщо k > 0
        то для i від 1 до rows повторити
            matrix[i, j] = rand(-9, 9)
        інакше для i від rows до 1 повторити з кроком -1
            matrix[i, j] = rand(-9, 9)
    k = k * -1
все повторити
```

outMatrix(matrix, rows, columns)

```
для i від 1 до rows повторити
    для j від 1 до columns повторити
        виведення matrix[i, j]
    все повторити
виведення '\n'
все повторити
```

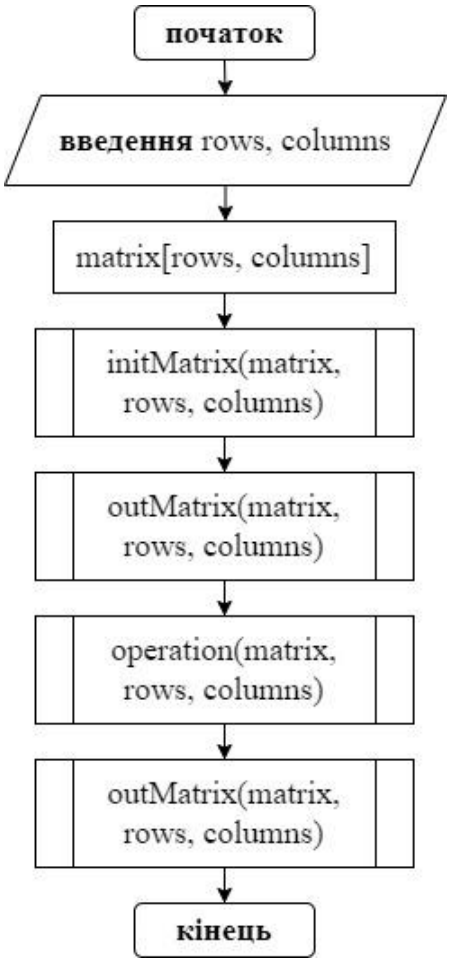
operation(matrix, rows, columns**)**

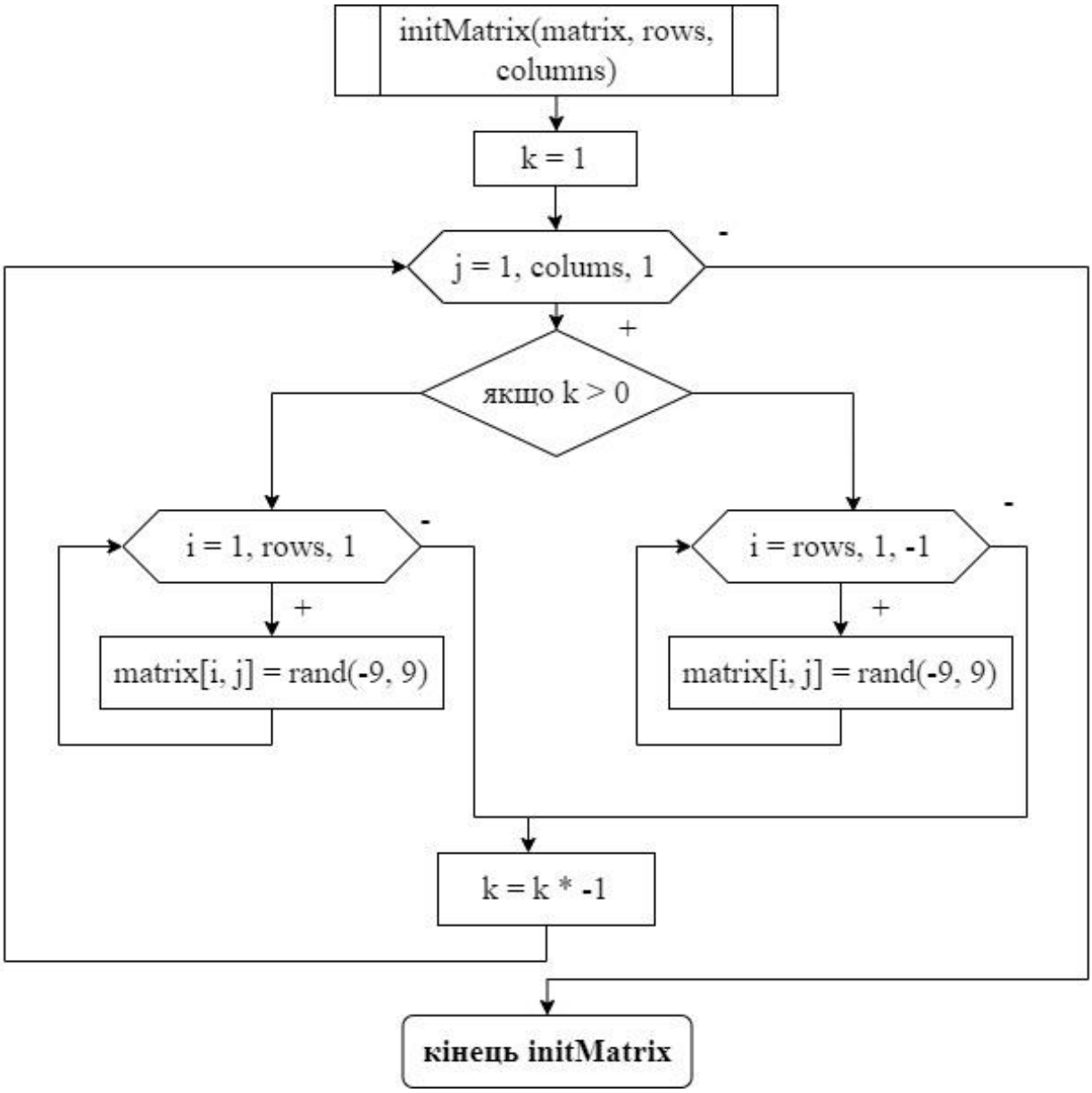
middleRow = (row / 2) + 1
для j **від** 1 **до** columns **повторити**

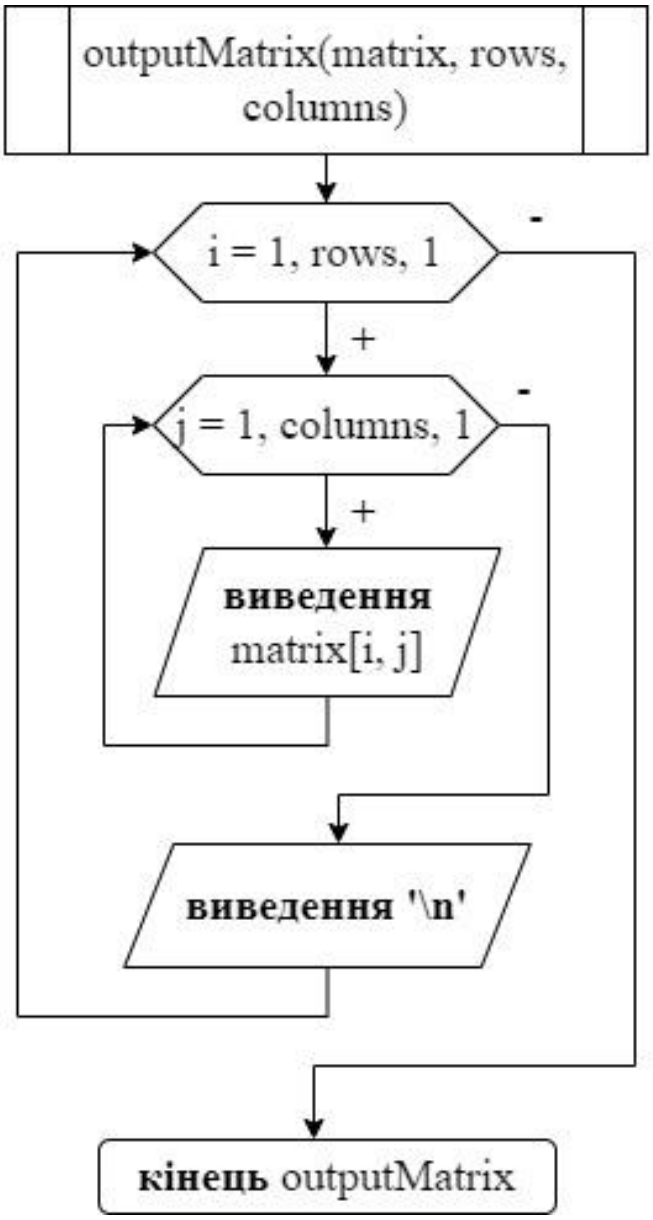
    element = -1
    для i **від** 1 **до** rows **повторити**
        якщо matrix[i, j] > 0
            element = matrix[i, j]
            index = i
        все якщо
    все повторити
    якщо element != -1
        tmp = matrix[middleRow, j]
        matrix[middleRow, j] = element
        matrix[index, j] = tmp
    все якщо

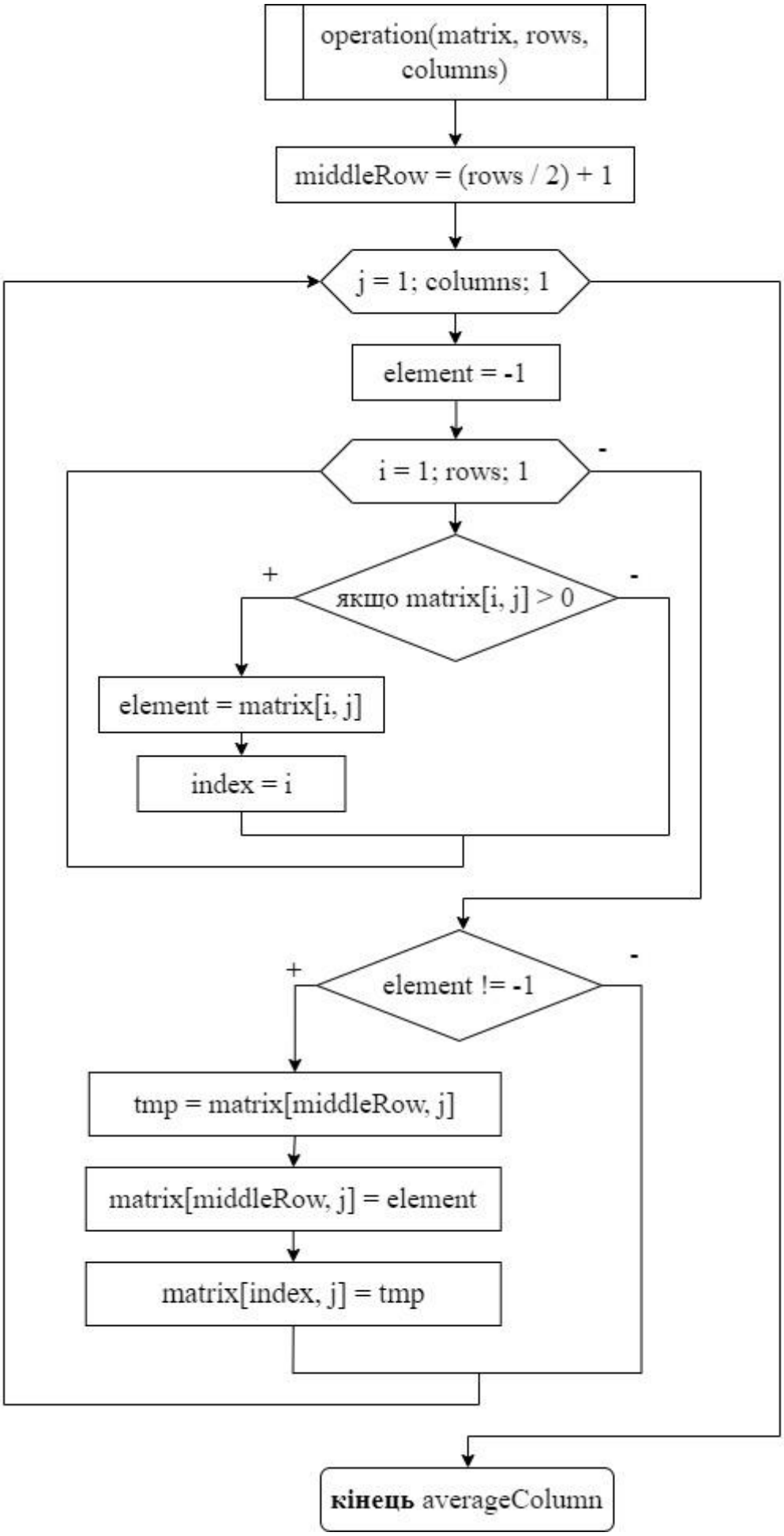
все повторити

Блок-схема




initMatrix(matrix, rows, columns)
k = 1
j = 1, colums, 1
-
+
якщо k > 0
i = 1, rows, 1
-
+
matrix[i, j] = rand(-9, 9)
i = rows, 1, -1
-
+
matrix[i, j] = rand(-9, 9)
k = k * -1
кінець initMatrix


outputMatrix(matrix, rows, columns)
i = 1, rows, 1
-
+
j = 1, columns, 1
-
+
виведення matrix[i, j]
виведення '\n'
кінець outputMatrix


operation(matrix, rows, columns)
middleRow = (rows / 2) + 1
j = 1; columns; 1
element = -1
i = 1; rows; 1
-
+
якщо matrix[i, j] > 0
-
element = matrix[i, j]
index = i
+
element != -1
-
tmp = matrix[middleRow, j]
matrix[middleRow, j] = element
matrix[index, j] = tmp
кінець averageColumn

Код програми:

ACD

Lab_9 (ACD).cpp

Lab_9 (ACD)

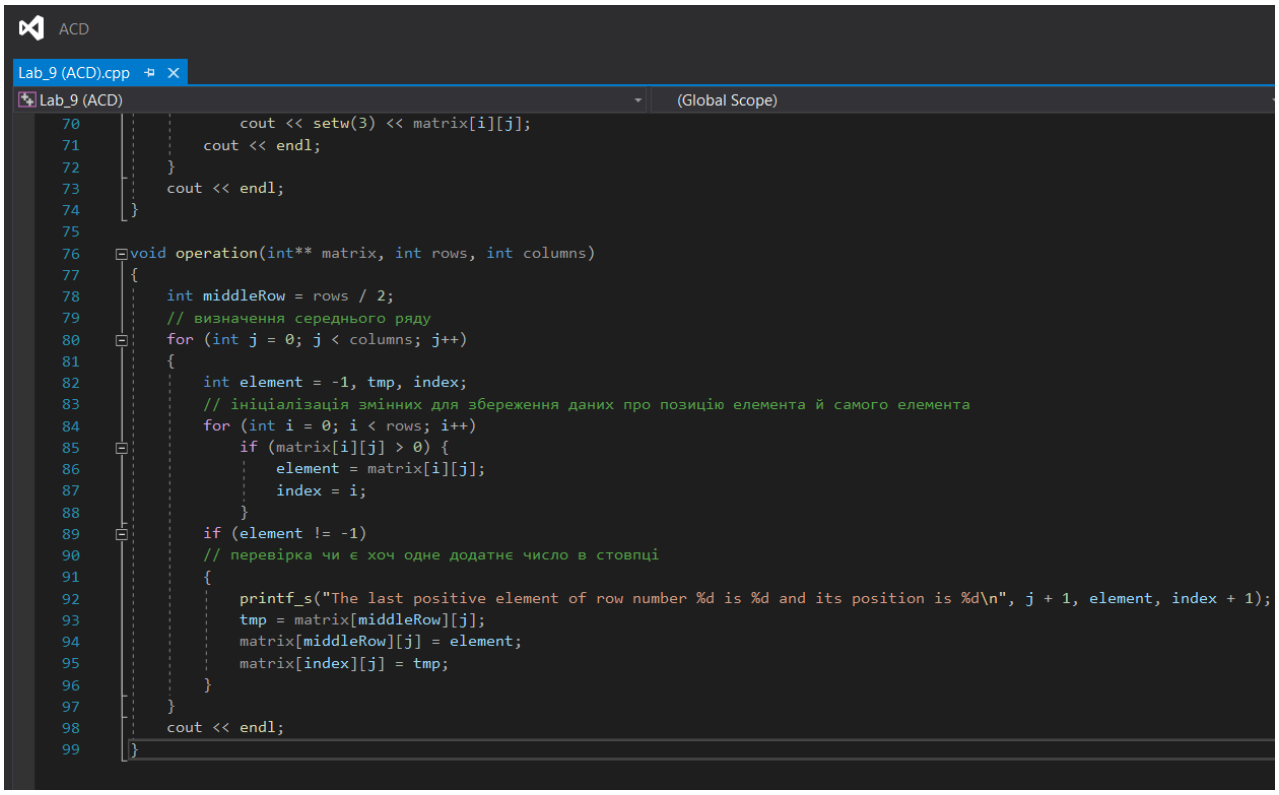
```cpp
#include <iostream>
#include <iomanip>
#include "stdlib.h"
#include "time.h"

using namespace std;

void initMatrix(int**, int, int);
void outMatrix(int**, int, int);
void operation(int**, int, int);


int main()
{
    int rows, columns;
    cout << "Enter a number of rows: "; cin >> rows;
    cout << "Enter a number of column: "; cin >> columns;
    // введенян кількості стовпців і рядків

    int** matrix = new int* [rows];
    for (int i = 0; i < rows; i++)
        matrix[i] = new int[columns];
    // створення динамічної матриці

    cout << "The matrix is\n";
    initMatrix(matrix, rows, columns);
    outMatrix(matrix, rows, columns);
    // ініціалізація матриці рандомно заданими числами

    operation(matrix, rows, columns);
    cout << "The processed matrix\n";
    outMatrix(matrix, rows, columns);
    // обробка матриці згідно з варіантом

    for (int i = 0; i < rows; i++)
        delete[] matrix[i];
    delete[] matrix;
    // видалення матриці

    return 0;
}
```

ACD

Lab_9 (ACD).cpp

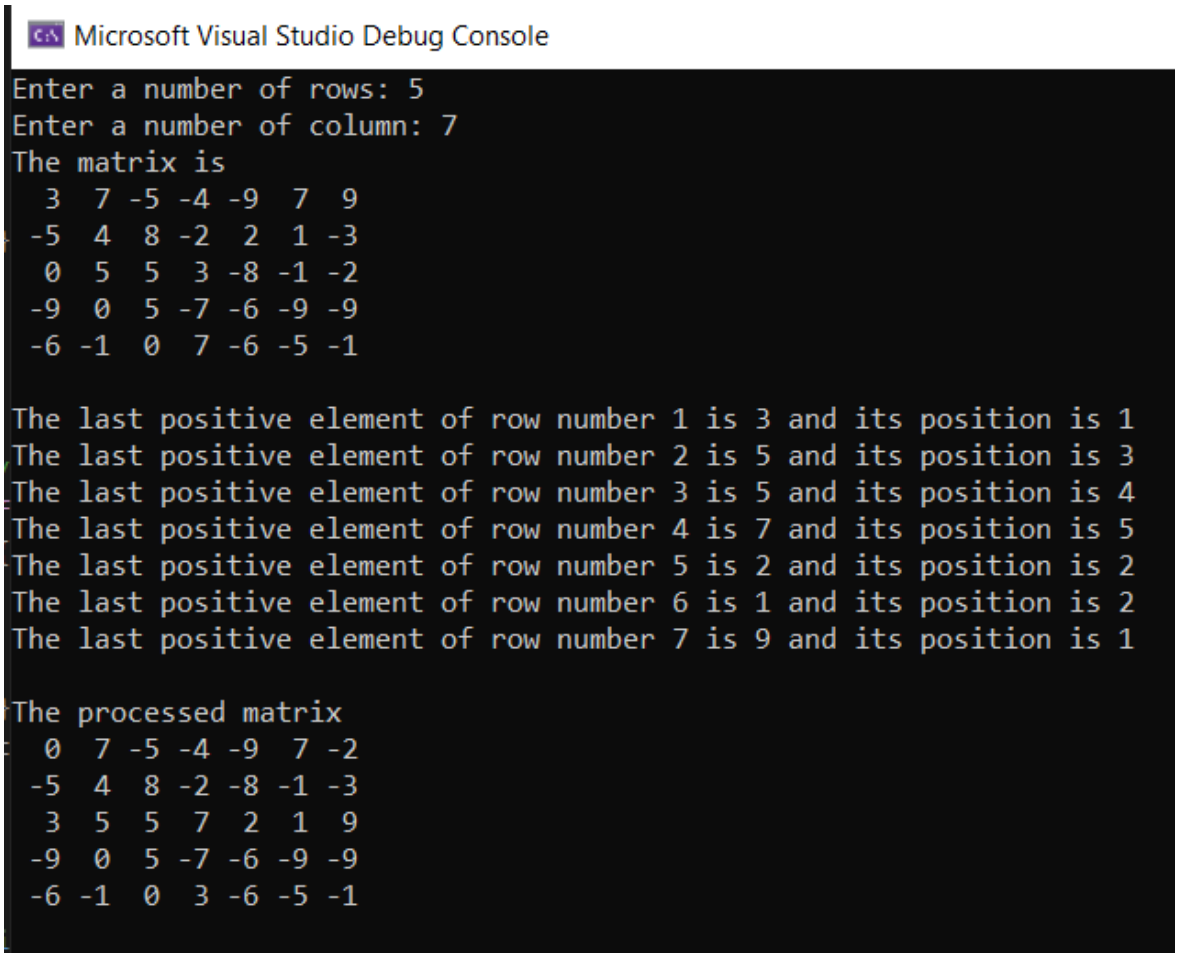
Lab_9 (ACD)

```
43
44  void initMatrix(int** matrix, int rows, int columns)
45  {
46      // ініціалізація матриці обходом змійкою по стовпцях
47      srand(time(NULL));
48      int k = 1;
49      for (int j = 0; j < columns; j++)
50      {
51          if (k > 0) {
52              for (int i = 0; i < rows; i++)
53                  matrix[i][j] = rand()%19 - 9;
54          }
55          else {
56              for (int i = rows - 1; i >= 0; i--)
57                  matrix[i][j] = rand() % 19 - 9;
58          }
59          k *= -1;
60      }
61
62  }
63
64  void outMatrix(int** matrix, int rows, int columns)
65  {
66      // виведення матриці
67      for (int i = 0; i < rows; i++)
68      {
69          for (int j = 0; j < columns; j++)
70              cout << setw(3) << matrix[i][j];
71          cout << endl;
72      }
73      cout << endl;
74  }
75
```

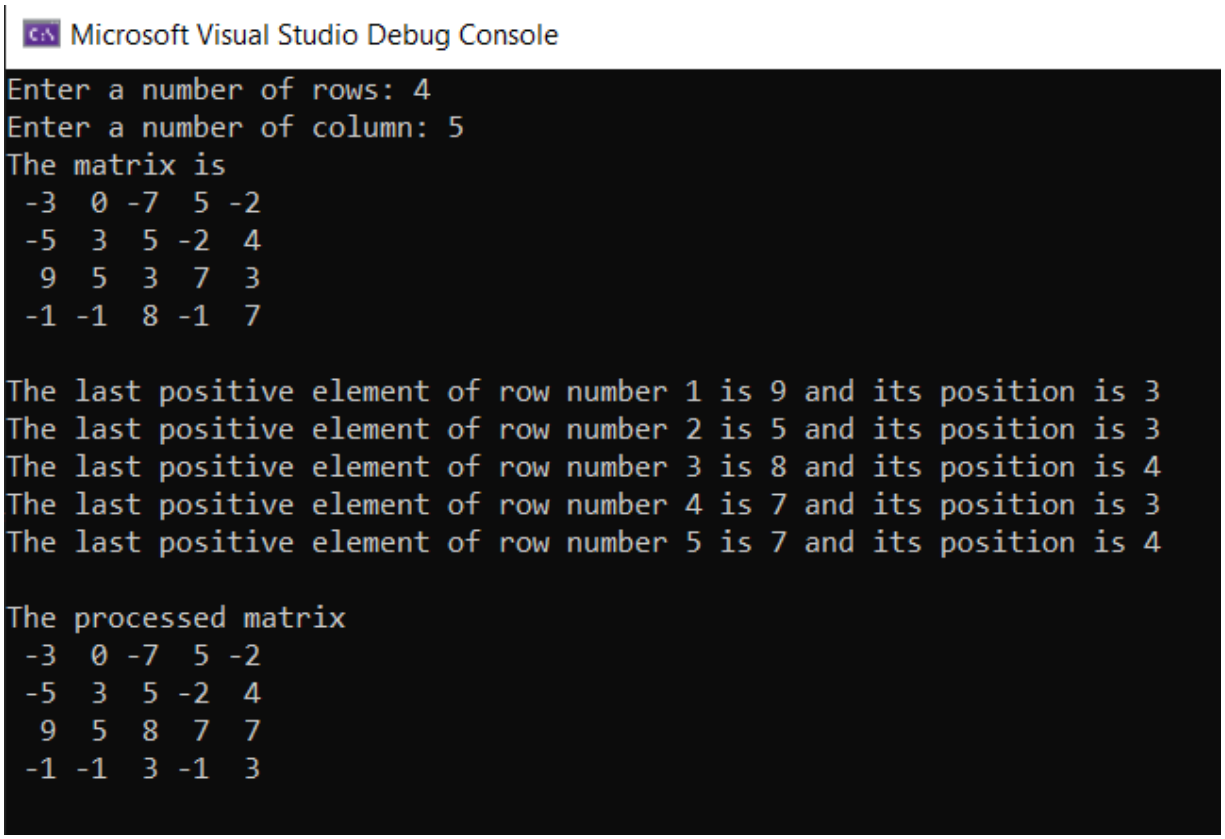


```
Lab_9 (ACD).cpp
Lab_9 (ACD)                                   (Global Scope)
70              cout << setw(3) << matrix[i][j];
71          cout << endl;
72      }
73      cout << endl;
74  }
75
76  void operation(int** matrix, int rows, int columns)
77  {
78      int middleRow = rows / 2;
79      // визначення середнього ряду
80      for (int j = 0; j < columns; j++)
81      {
82          int element = -1, tmp, index;
83          // ініціалізація змінних для збереження даних про позицію елемента й самого елемента
84          for (int i = 0; i < rows; i++)
85              if (matrix[i][j] > 0) {
86                  element = matrix[i][j];
87                  index = i;
88              }
89          if (element != -1)
90          // перевірка чи є хоч одне додатнє число в стовпці
91          {
92              printf_s("The last positive element of row number %d is %d and its position is %d\n", j + 1, element, index + 1);
93              tmp = matrix[middleRow][j];
94              matrix[middleRow][j] = element;
95              matrix[index][j] = tmp;
96          }
97      }
98      cout << endl;
99  }
```

Випробування алгоритму:



```
Microsoft Visual Studio Debug Console
Enter a number of rows: 5
Enter a number of column: 7
The matrix is
  3  7 -5 -4 -9  7  9
 -5  4  8 -2  2  1 -3
  0  5  5  3 -8 -1 -2
 -9  0  5 -7 -6 -9 -9
 -6 -1  0  7 -6 -5 -1

The last positive element of row number 1 is 3 and its position is 1
The last positive element of row number 2 is 5 and its position is 3
The last positive element of row number 3 is 5 and its position is 4
The last positive element of row number 4 is 7 and its position is 5
The last positive element of row number 5 is 2 and its position is 2
The last positive element of row number 6 is 1 and its position is 2
The last positive element of row number 7 is 9 and its position is 1

The processed matrix
  0  7 -5 -4 -9  7 -2
 -5  4  8 -2 -8 -1 -3
  3  5  5  7  2  1  9
 -9  0  5 -7 -6 -9 -9
 -6 -1  0  3 -6 -5 -1
```



Висновки:

На цій практичній ми дослідити алгоритми обходу матриць, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Також зробили постановку задачі склали матмодель написали псевдокод та намлювали блок-схему. Отримали очікуваний результат.