Додаток 1

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 лінійних алгоритмів»

Варіант 9

Виконав студент Григоренко Родіон Ярославович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська А.С.
(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.

Індивідуальне завдання **Варіант 9**

9	Задано матрицю дійсних чисел A[m,n]. При обході матриці по рядках знайти в ній останній нульовий елемент X і його місцезнаходження. Обміняти знайдене значення X з елементом першого стовбця.

Постановка задачі

Результатом є виведення координат останнього нульового елементу знайденого в матриці при обході по рядках та виведення самої матриці,де останній нульовий елемент і елемент першого стовпця змінені місцями.

Побудова математичної моделі

Складемо таблицю імен змінних та функцій

Змінна	Тип	Ім’я	Призначення
Двовимірний динамічний масив	Покажчик на масив покажчиків(float)	A	Початкове дане
Кількість рядків	Цілочисельний	row	Проміжні дані
Кількість стовпців	Цілочисельний	col	Проміжні дані
Параметри арифметичних циклів	Цілочисельний	i,j,k	Проміжні дані
Одновимірний динамічний масив	int*	index	Проміжні дані
Функція для ініціювання початкового масиву	float**	init	Ініціювання початкового масиву
Функція виведення масиву	void	output	Виведення масиву
Функція для зміни місцями елементів масиву	void	swap	Зміна місцями елементів масиву

Функція для знаходження координат нульових елементів	bool	check	Знаходження координат нульових елементів
Головна функція для пошуку останнього нульового елементу і обробки матриці	void	process	Пошук останнього нульового елементу і обробка матриці
Функція для заповнення масиву рандомними значеннями	Цілочисельний	rand	Заповнення масиву рандомними значеннями

init - функція для ініціювання початкового двовимірного динамічного масиву.

output - функція для виведення матриці

A- двовимірний масив, розмірність якого вводиться користувачем, заповнюється випадковими числами,за допомогою функції rand().

row, col - розмірності масивів.

Розв'язання

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми. Крок 1. Визначимо основні дії.

Крок 2.Деталізуємо функцію init.

Крок 3.Деталізуємо функцію output.

Крок 4.Деталізуємо функцію process.

Крок 5.Деталізуємо функцію check.

Крок 6.Деталізуємо функцію swap.

Псевдокод

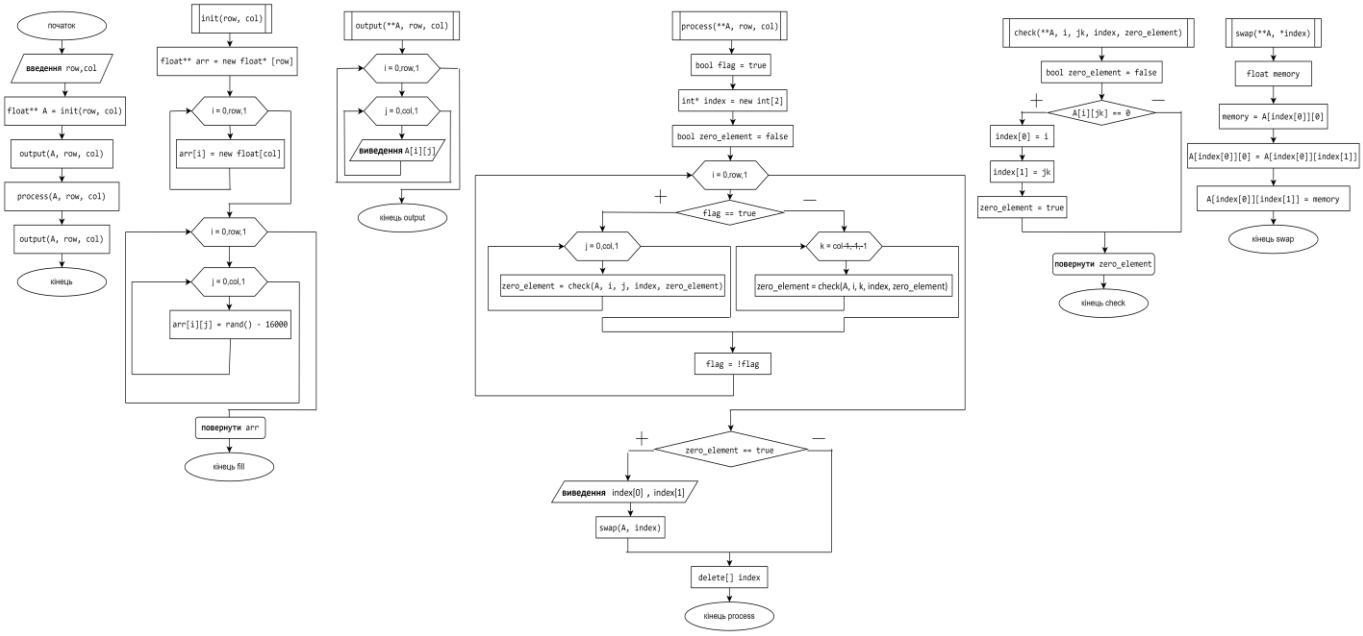
Крок 6

```
функція init(row, col)
    float** arr = new float* [row]
    для i від 0 до row, збільшувати на 1
        arr[i] = new float[col]
        для j від 0 до col,збільшувати на 1
            arr[i][j] = rand() - 16000
    повернути arr
все функція
```

```
функція output(**A, row, col)
    для i від 0 до row, збільшувати на 1
        для j від 0 до col, збільшувати на 1
            виведення A[i][j]
все функція

функція process(**A, row, col)
   bool flag = true
   int* index = new int[2]
   bool zero_element = false
   для i від 0 до row, збільшувати на 1
       якщо flag == true
          то
              для j від 0 до col, збільшувати на 1
                  zero_element = check(A, i, j, index, zero_element)
          інакше
              для k від col – 1 включно до 0, зменшувати на 1
                  zero_element = check(A, i, k, index, zero_element)
      flag = !flag
   якщо zero_element == true
      то
          виведення  index[0] , index[1]
          swap(A, index)
   delete[] index
все функція

функція check(**A, i, jk, index, zero_element)
   якщо A[i][jk] == 0
       то
           index[0] = i
           index[1] = jk
           zero_element = true
   повернути zero_element
все функція

функція swap(**A, *index)
    float memory
    memory = A[index[0]][0]
    A[index[0]][0] = A[index[0]][index[1]]
    A[index[0]][index[1]] = memory
все функція

початок
    int row, col
    введення row,col
    float** A = init(row, col)
    output(A, row, col)
    process(A, row, col)
    output(A, row, col)
кінець
```

Блок-схема

Крок 6



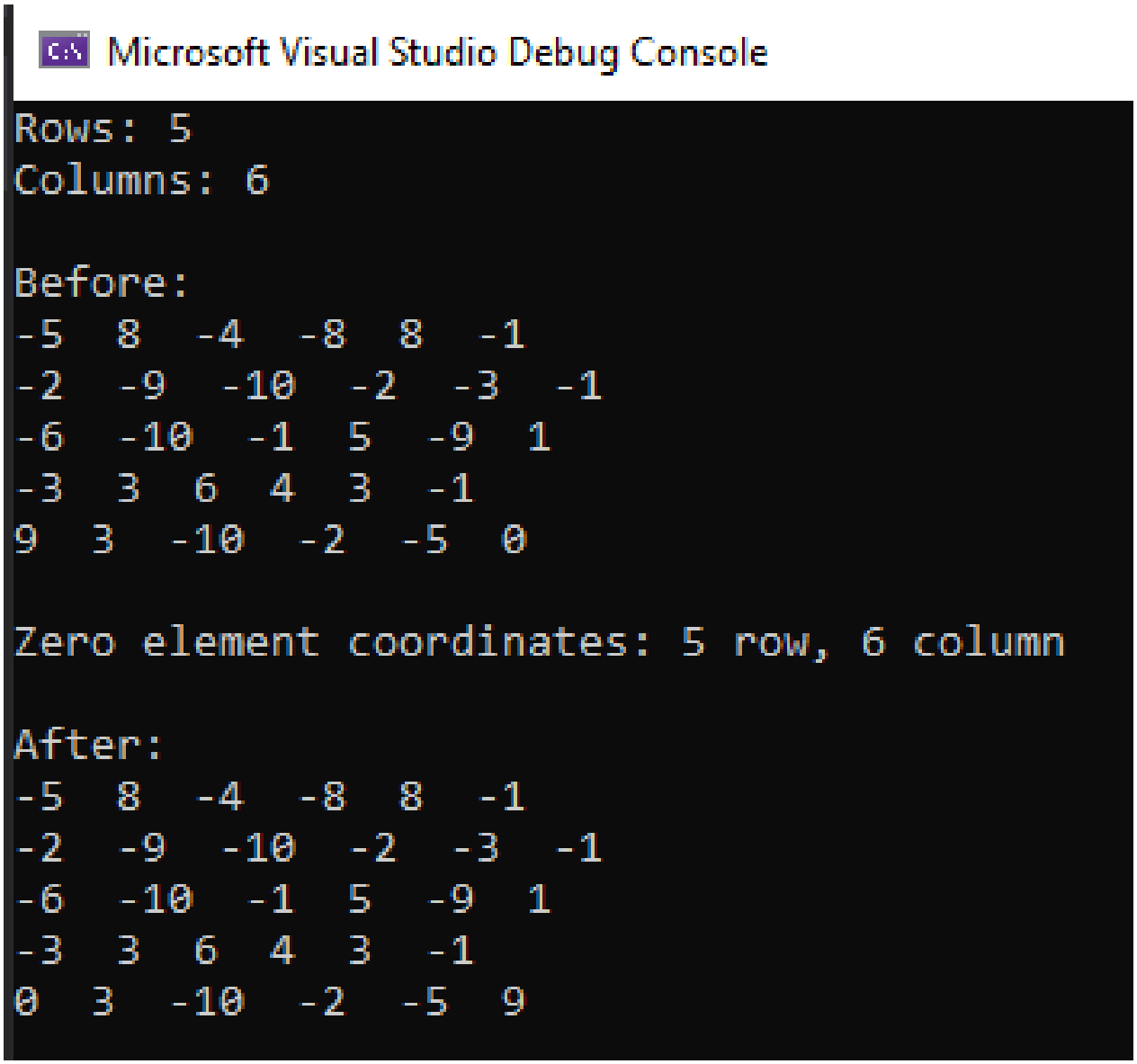
Код

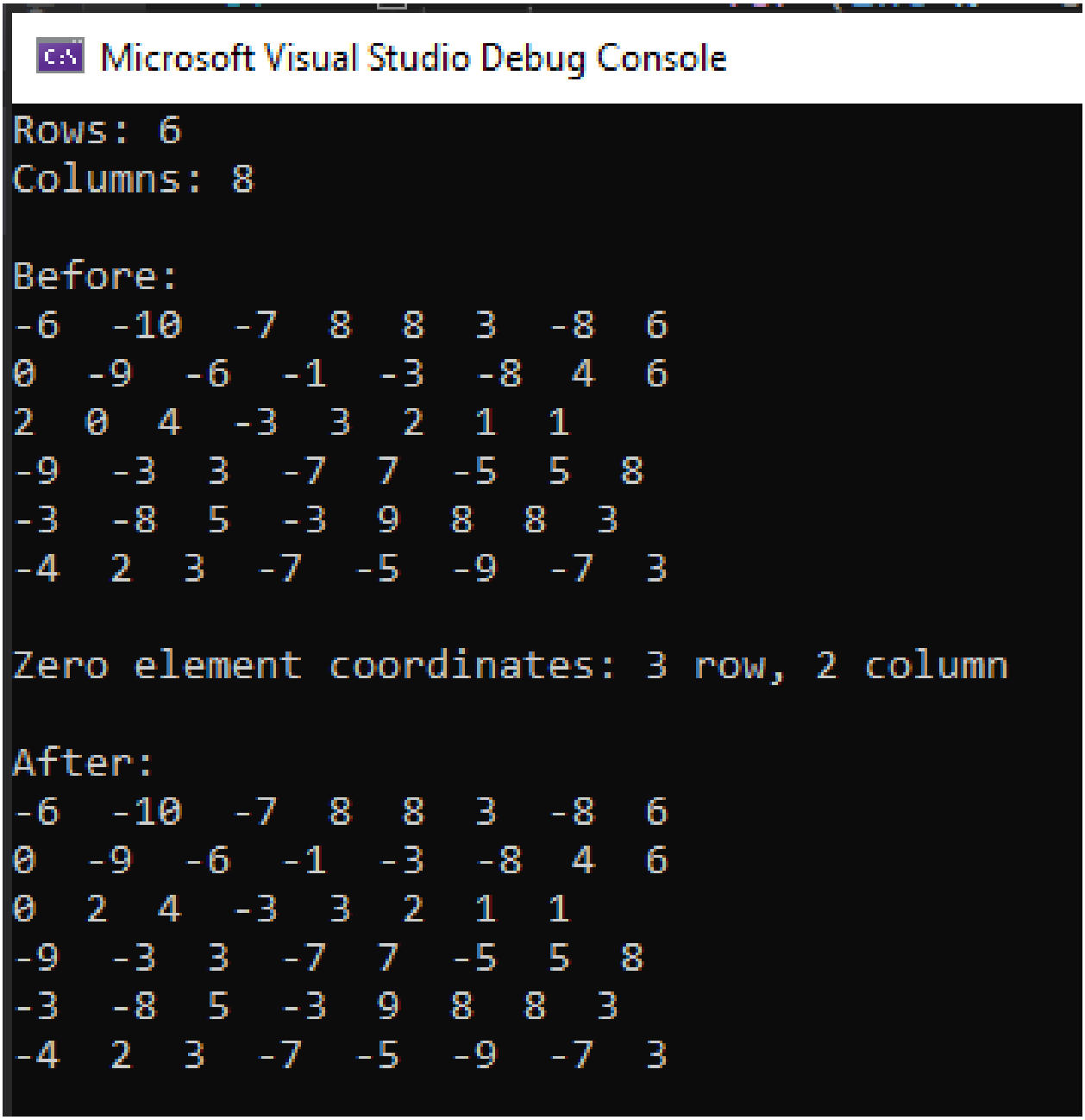
```cpp
#include <stdlib.h>
#include <iostream>
#include <time.h>
using namespace std;
float** init(int row, int col);
void output(float** A,int row,int col);
void process(float** A, int row, int col);
void swap(float** A, int* index);
bool check(float** A, int i, int jk, int* index, bool zero_element);
int main()
{
    int row, col;
    cout << "Rows: ";
    cin >> row;
    cout << "Columns: ";
    cin >> col;
    float** A = init(row, col);
    cout << "\n" << "Before:" << "\n";
    output(A, row, col);
    process(A, row, col);
    cout << "\n" << "After:" << "\n";
    output(A, row, col);
}
float** init(int row, int col) {
    srand(time(NULL));
    float** arr = new float* [row];
    for (int i = 0; i < row; i++) {
        arr[i] = new float[col];
        for (int j = 0; j < col; j++) {
            arr[i][j] = rand() % 20 - 10;
        }
    }
    return arr;
}
```

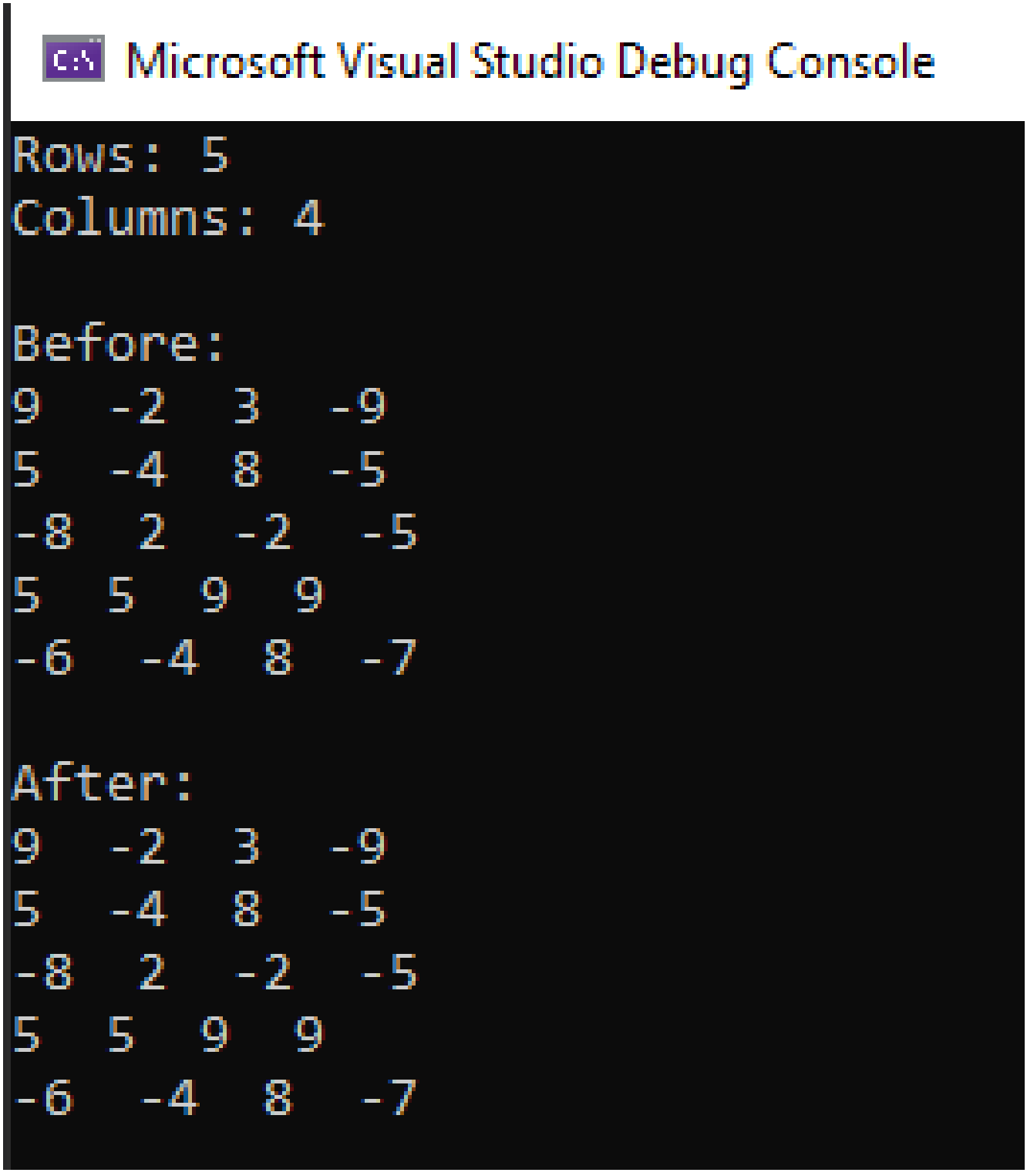
```cpp
void output(float** A,int row,int col) {
    for (int i = 0; i < row; i++)
    {
        for (int j = 0; j < col; j++) {
            cout << A[i][j] << "  ";
        }
        cout << '\n';
    }
}

void process(float** A, int row, int col) {
    bool flag = true;
    int* index = new int[2];
    bool zero_element = false;
    for (int i = 0; i < row; i++) {
        if (flag == true) {
            for (int j = 0; j < col; j++) {
                zero_element = check(A, i, j, index, zero_element);
            }
        }
        else {
            for (int k = col - 1; k >= 0; k--) {
                zero_element = check(A, i, k, index, zero_element);
            }
        }
        flag = !flag;
    }
    if (zero_element == true) {
        cout << "\n" << "Zero element coordinates: " << index[0] + 1 << " row, " << index[1] + 1 << " column" << "\n";
        swap(A, index);
    }
    delete[] index;
}
```

```
bool check(float** A, int i, int jk, int* index,bool zero_element) {
    if (A[i][jk] > -0.1 && A[i][jk] < 0.1) {
        index[0] = i;
        index[1] = jk;
        zero_element = true;
    }
    return zero_element;
}

void swap(float** A, int* index) {
    float memory;
    memory = A[index[0]][0];
    A[index[0]][0] = A[index[0]][index[1]];
    A[index[0]][index[1]] = memory;
}
```

Результат коду




Microsoft Visual Studio Debug Console
Rows: 6
Columns: 8
Before:
-6 -10 -7 8 8 3 -8 6
0 -9 -6 -1 -3 -8 4 6
2 0 4 -3 3 2 1 1
-9 -3 3 -7 7 -5 5 8
-3 -8 5 -3 9 8 8 3
-4 2 3 -7 -5 -9 -7 3
Zero element coordinates: 3 row, 2 column
After:
-6 -10 -7 8 8 3 -8 6
0 -9 -6 -1 -3 -8 4 6
0 2 4 -3 3 2 1 1
-9 -3 3 -7 7 -5 5 8
-3 -8 5 -3 9 8 8 3
-4 2 3 -7 -5 -9 -7 3



Висновки

Я дослідив алгоритми обходу масивів, набув практичних навичок використання цих алгоритмів під час складання програмних специфікацій.Побудував мат. модель, псевдокод та блок схему.Протестував алгоритм.