Національний технічний університет України
“Київський політехнічний інститут”

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп’ютерних систем

ЛАБОРАТОРНА РОБОТА №1.4.

з дисципліни “Структури даних і алгоритми”

ТЕМА: “АЛГОРИТМИ ЛІНІЙНОГО ПОШУКУ ”

Група: КВ-34
Виконав: Фесенко Д.О.
Оцінка:

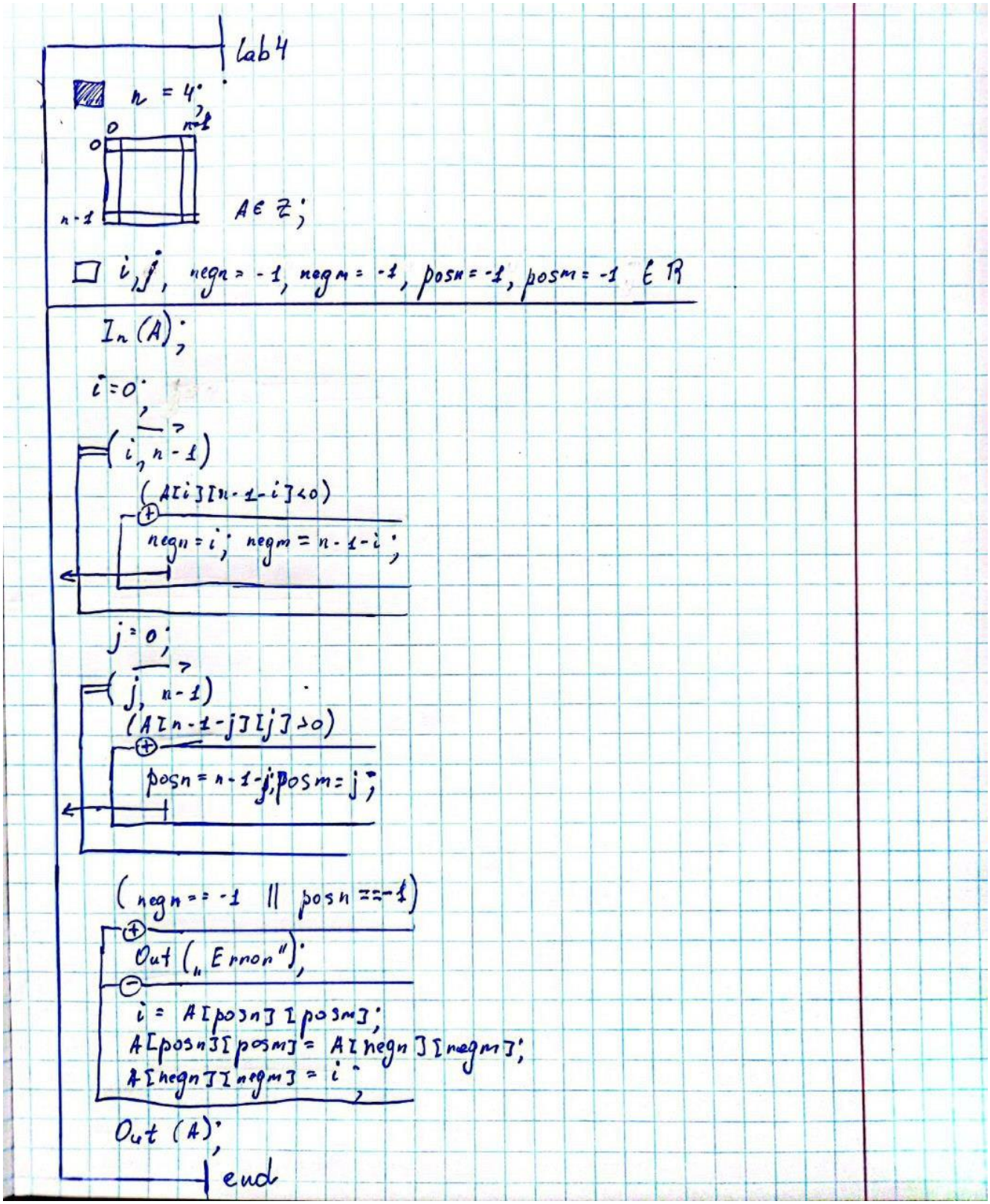
Постановка задачі

1. Написати програму розв’язання задачі пошуку (за варіантом) у двовимірному масиві (матриці) A[m][n] або A[n][n], в залежності від варіанту, одним з алгоритмів методу лінійного пошуку.
2. В усіх варіантах завдань, в яких не зазначено явним чином, який по порядку елемент потрібно знайти, перший чи останній, задача лінійного пошуку трактується в її класичному визначенні, тобто пошук продовжується до першого співпадаючого елемента по ходу у напрямку пошуку.
3. Виконати тестування та налагодження програми згідно вказівок, описаних у розділі «Тестування».

Завдання за варіантом №23

Задано матрицю цілих чисел A[n][n]. У побічній діагоналі матриці знайти перший від’ємний і останній додатний елементи, а також поміняти їх місцями.

Діаграма дій



Код програми

```c
#include <stdio.h>
#define n 8

int main() {
    int A[n][n], i, j, negn = -1, negm = -1, posn = -1, posm = -1;

    for (i = 0; i <= n - 1; i++) {
       for (j = 0; j <= n - 1; j++) {
           scanf("%d", &A[i][j]);
       }
    }
    for (i = 0; i <= n-1; i++) {
        if (A[i][n - 1 - i]< 0) {
            negn = i; negm = n -1 - i; break;
        }
    }
    for (j = 0; j <= n - 1; j++) {
        if (A[n - 1 - j][j] > 0) {
            posn = n - 1 - j; posm = j; break;
        }
    }

    printf("Matrix before transposition:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("[%6d] ", A[i][j]);
        }
        printf("\n");
    }

    if (negn == -1 || posn == -1) printf("Error\n");
    else {
        i = A[posn][posm];
        A[posn][posm] = A[negn][negm];
        A[negn][negm] = i;
    }

    printf("Matrix after transposition:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("[%6d] ", A[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Тестування

1) m = n = 1 (вироджений випадок);

```
Matrix before transposition:
[    -5]
Error
Matrix after transposition:
[    -5]
```

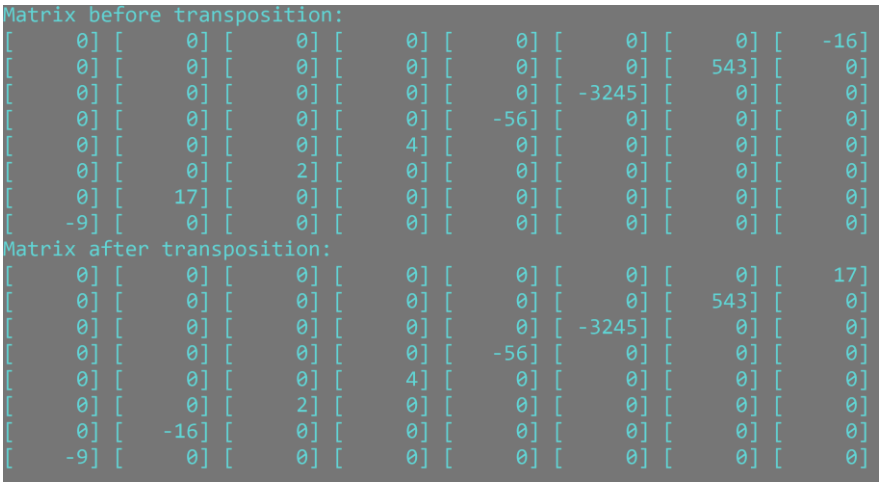
2) m = n = 3 (майже вироджений випадок);

```
Matrix before transposition:
[     0] [     1] [   -16]
[     3] [     4] [     5]
[     0] [     7] [     8]
Matrix after transposition:
[     0] [     1] [     4]
[     3] [   -16] [     5]
[     0] [     7] [     8]
```

3) Шуканий елемент відсутній у заданій за варіантом області;

```
Matrix before transposition:
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
Error
Matrix after transposition:
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
[     0] [     0] [     0] [     0] [     0] [     0]
```

4) У заданій за варіантом області знаходиться не менше трьох елементів, співпадаючих за значенням з шуканим;



5) Шуканий елемент присутній у заданій за варіантом області один раз і знаходиться на останній позиції цієї області;