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

Лабораторна робота №1.5

з дисципліни
«Алгоритми і структури даних»

Виконав

Студент групи ІМ-34
Никифоров Артем Михайлович
номер у списку групи: 17

Перевірила:

Молчанова А. А.

Київ 2023

Постановка задачі

1. Написати програму розв’язання задачі пошуку (за варіантом) у двовимірному масиві (матриці) одним з алгоритмів методу лінійного пошуку.

2. Розміри матриці m та n взяти самостійно у межах від 7 до 10.

3. Виконати тестування та налагодження програми на комп’ютері. При тестуванні програми необхідно підбирати такі вхідні набори початкових значеннь матриці, щоб можна було легко відстежити коректність виконання пошуку і ця коректність була б протестована для всіх можливих випадків. З метою тестування дозволяється використовувати матриці меншого розміру.

Варіант 17

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

Текст програми:

```
#include <stdio.h>


int main() {
    int n;
    int i, j;
    printf("matrix size:");
    scanf("%d", &n);
    int A[n][n];
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("A[%d][%d] =", i, j);
            scanf("%d", &A[i][j]);
        }
    }
    printf("matrix A:\n");
    for (i = 0; i < n; i++) {
        for (j = 0; j < n; j++) {
            printf("%d\t", A[i][j]);
        }
        printf("\n");
    }
    int firstPositive = -1;
    int lastNegative = -1;
    for (i = 0; i < n; i++) {
        if (A[n - 1 - i][i] > 0 && firstPositive == -1) {
            firstPositive = i;
        }
        if (A[n - 1 - i][i] < 0) {
            lastNegative = i;
        }
    }
    if (firstPositive != -1) {
        printf("pershiy dodatniy: A[%d][%d]\n", n - 1 - firstPositive, firstPositive);
```

```
    } else {
        printf("nema dodatnih\n");
    }
    if (lastNegative != -1) {
        printf("ostanniy vid'emniy -: A[%d][%d]\n", n - 1 - lastNegative, lastNegative);
    } else {
        printf("nema vid'emnih\n");
    }
    if (firstPositive != -1 && lastNegative != -1) {
        int temp = A[n - 1 - firstPositive][firstPositive];
        A[n - 1 - firstPositive][firstPositive] = A[n - 1 - lastNegative][lastNegative];
        A[n - 1 - lastNegative][lastNegative] = temp;
        printf("new matrix:\n");
        for (i = 0; i < n; i++) {
            for (j = 0; j < n; j++) {
                printf("%d\t", A[i][j]);
            }
            printf("\n");
        }
    }else{
            printf("nemae dodatnih ta/abo vid'emnih elementiv");
    }
    return 0;
}
```

Результати тестування програми

```
matrix A:
1       2       3
1       -2      3
-1      2       3
pershiy dodatniy: A[0][2]
ostanniy vid'emniy -: A[1][1]
new matrix:
1       2       -2
1       3       3
-1      2       3
```

```
matrix A:
1       2
-1      -2
pershiy dodatniy: A[0][1]
ostanniy vid'emniy -: A[1][0]
new matrix:
1       -1
2       -2
```

```
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
nema dodatnih
nema vid'emnih
nemae dodatnih ta/abo vid'emnih elementiv
Process finished with exit code 0
```

```
matrix A:
1       -2      3       -4      5       -6      -7      -8      9       10
-1      2       -3      4       -5      6       7       8       -9      10
-1      -2      -3      -4      -5      -6      -7      -8      9       0
-10     1       2       3       4       5       6       7       8       9
10      1       2       3       4       5       6       1       1       1
1       1       1       1       1       1       1       1       1       1
1       1       1       1       1       1       1       1       1       1
1       1       1       1       1       1       1       1       1       -1
-1      -1      -1      -1      -1      -1      -1      -1      -1      -1
-1      -1      -1      -1      1       2       3       4       5       6
pershiy dodatniy: A[7][2]
ostanniy vid'emniy -: A[1][8]
new matrix:
1       -2      3       -4      5       -6      -7      -8      9       10
-1      2       -3      4       -5      6       7       8       1       10
-1      -2      -3      -4      -5      -6      -7      -8      9       0
-10     1       2       3       4       5       6       7       8       9
10      1       2       3       4       5       6       1       1       1
1       1       1       1       1       1       1       1       1       1
1       1       1       1       1       1       1       1       1       1
1       1       -9      1       1       1       1       1       1       -1
-1      -1      -1      -1      -1      -1      -1      -1      -1      -1
-1      -1      -1      -1      1       2       3       4       5       6

Process finished with exit code 0
```

```
0       0       0       0       -1
0       0       0       2       0
0       0       -3      0       0
0       4       0       0       0
-5      0       0       0       0
pershiy dodatniy: A[3][1]
ostanniy vid'emniy -: A[0][4]
new matrix:
0       0       0       0       4
0       0       0       2       0
0       0       -3      0       0
0       -1      0       0       0
-5      0       0       0       0

Process finished with exit code 0
```