**Лабораторна робота №2.1**

З дисципліни «Структури даних та алгоритми»

Алгоритми дівйкового пошуку

Виконав: Землянський Едуард

Група: КВ-22

**Мета лабораторної роботи**

Метою лабораторної роботи №2.1. «Алгоритми двійкового пошуку» є засвоєння теоретичного матеріалу та набуття практикних навичок рішення задачі пошуку заданої категорії елементів за допомогою різних алгоритмів методу двійкового пошуку у двовимірних масивах.

**Постановка задачі**

1. Написати програму розв’язання задачі пошуку (за варіантом) у двовимірному масиві (матриці) методом двійкового пошуку. Алгоритм двійкового пошуку задається варіантом завдання.
2. Розміри матриці m та n взяти самостійно у межах від 7 до 10.
3. 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 завдання**

Варіант № 12

Задано квадратну матрицю дійсних чисел A[n,n]. Визначити присутність у побічній діагоналі матриці заданого дійсного числа X і його місцезнаходження (координати) методом двійкового пошуку (Алгоритм №2), якщо елементи цієї діагоналі впорядковані за незменшенням

**Текст програми**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

#include <time.h>

void display(int n, float matrix[][n]){

    printf("\n");

    for(int i = 0; i < n; i++){

        for(int j = 0; j < n; j++){

            printf("%0.2f\t", matrix[i][j]);

        }

        printf("\n");

    }

}

void fill\_matrix(int n, int minval, int maxval, float matrix[][n]){

    for (int i = 0; i < n; i++){

        for (int j = 0; j < n; j++)

        {

            matrix[i][j] = minval + rand() % (maxval - minval + 1);

        }

    }

}

void get\_side\_diagonal(int n, float diagonal[n], float matrix[][n]){

    for (int i = 0; i < n; i++){

        diagonal[i] = matrix[i][n-i-1];

        printf("%0.2f\t", diagonal[i]);

    }

}

void swap(float\* xp, float\* yp)

{

    float temp = \*xp;

    \*xp = \*yp;

    \*yp = temp;

}

void bubbleSort(int n, float arr[n])

{

    for (int i = 0; i < n - 1; i++)

        for (int j = 0; j < n - i - 1; j++)

            if (arr[j] > arr[j + 1])

                swap(&arr[j], &arr[j + 1]);

}

int binSearch(int n, float arr[n], float value){

    int left = -1;

    int right = n - 1;

    while (left < right - 1){

        int mid = (left + right) / 2;

        if (arr[mid] < value)

            left = mid;

        else

            right = mid;

    }

    return right;

}

// if (arr[mid] <= value)

// ...

// return left; - for last right index. Standartly - left.

int main(){

    srand(time(0));

    int n;

    int minval, maxval;

    printf("Enter matrix dimension: ");

    scanf("%d", &n);

    printf("Enter min value: ");

    scanf("%d", &minval);

    printf("Enter max value: ");

    scanf("%d", &maxval);

    float A[n][n];

    float diagonal[n];

    fill\_matrix(n, minval, maxval, A);

    display(n, A);

    printf("\nSide diagonal is: \n");

    get\_side\_diagonal(n, diagonal, A);

    bubbleSort(n, diagonal);

    printf("\n\nSorted side diagonal: \n");

    for(int i = 0; i < n; i++)

        printf("%0.2f\t", diagonal[i]);

    float value;

    printf("\nEnter value to search: ");

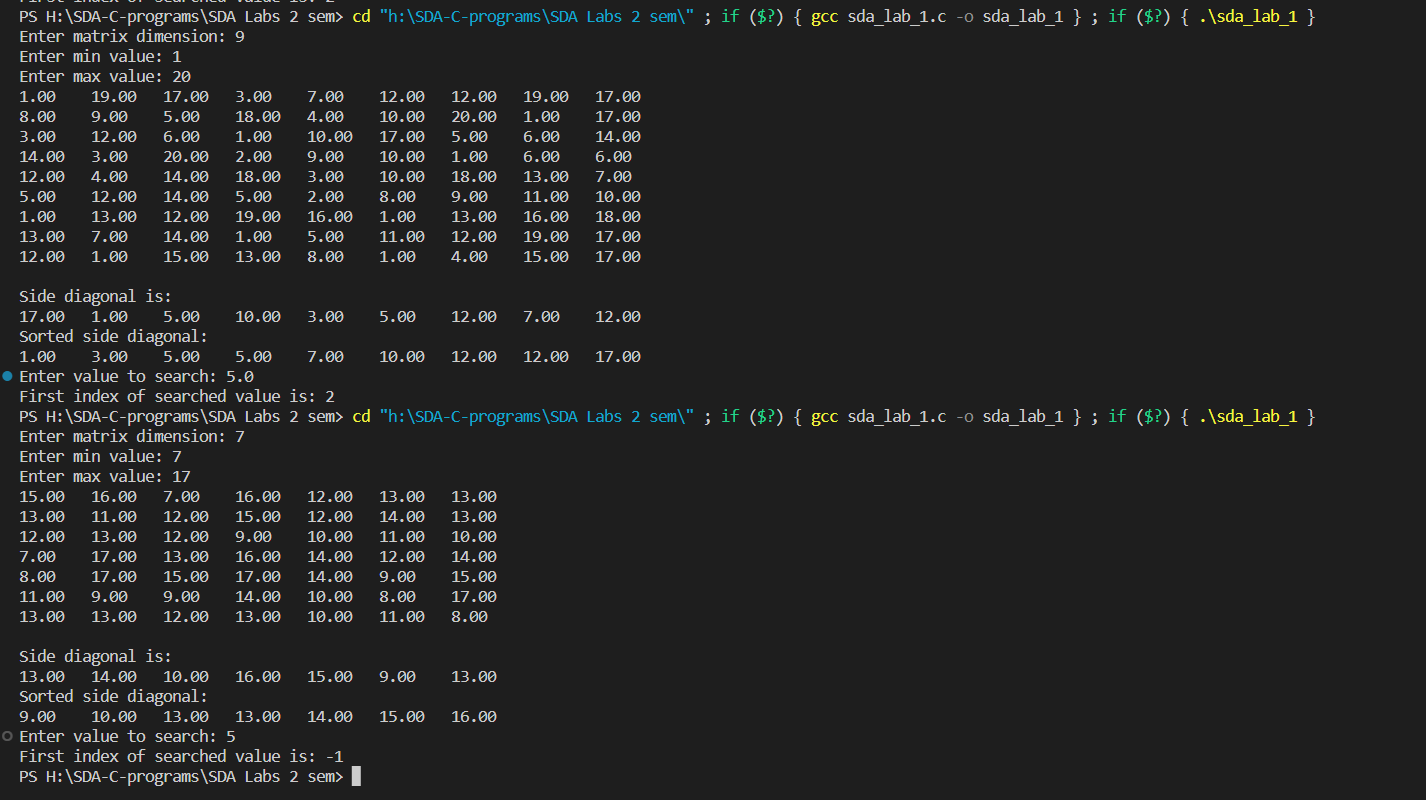
    scanf("%f", &value);

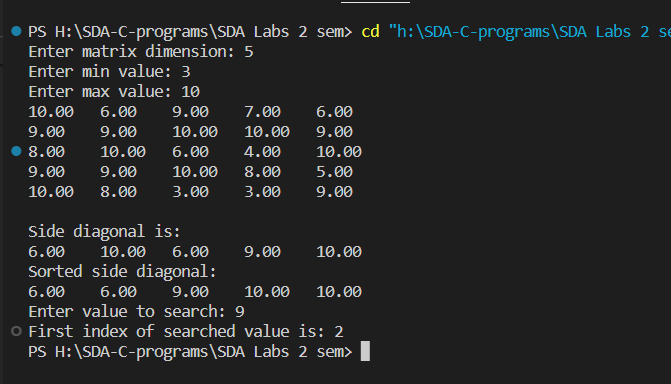
    printf("First index of searched value is: %d", binSearch(n, diagonal, value));

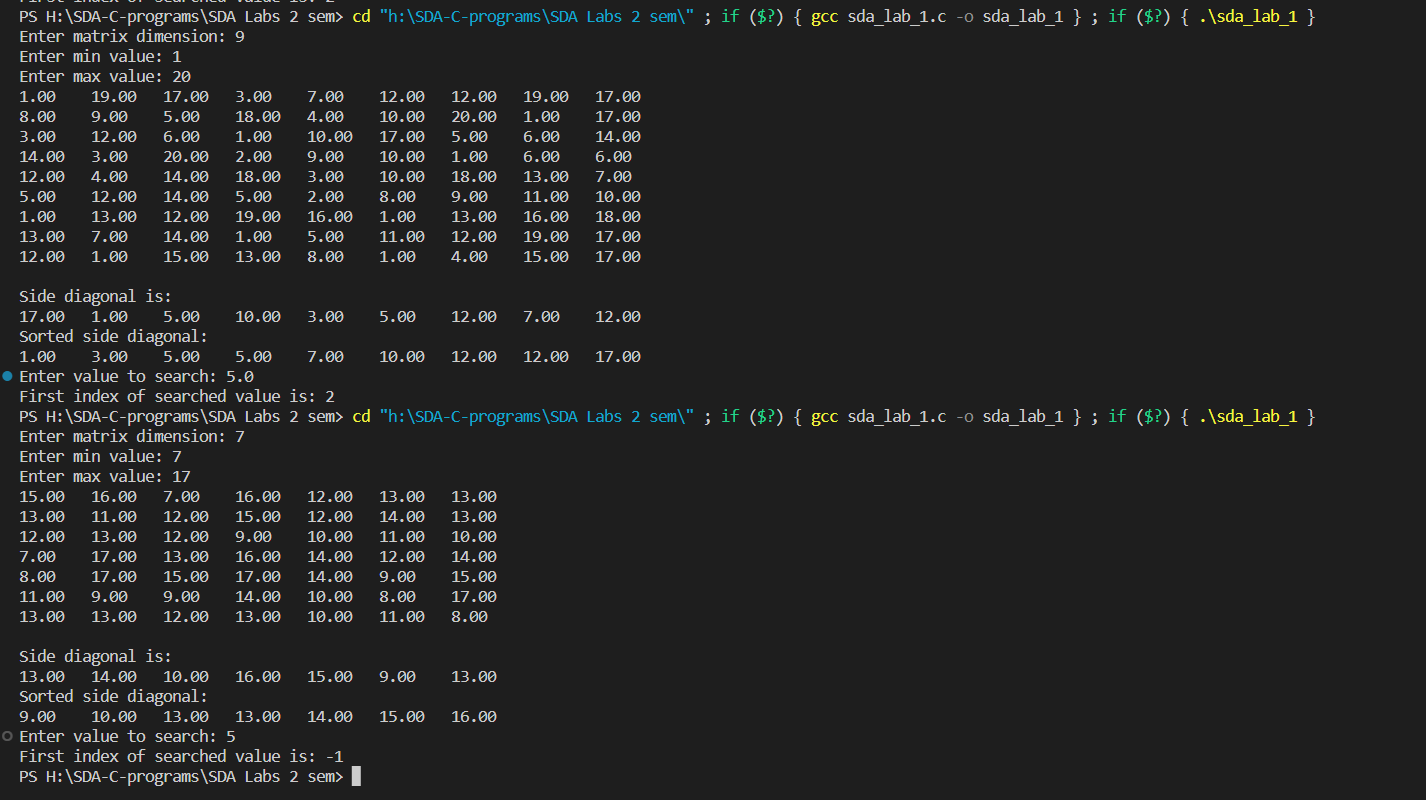
    return 0;

}

**Тестування, результати для різних вхідних даних:**

****

****

****