**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 №5**

з дисципліни  
«Алгоритми і структури даних»

Виконав: Перевірила:

студент групи IM-43 Молчанова А. А.

Олексійчук Станіслав Юрійович

номер у списку групи: 23

Київ 2024

***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 від- стежити коректність виконання пошуку і ця коректність була б протестована для всіх можливих випадків. З метою тестування дозволяється використовувати матриці меншого розміру.

***Варіант № 23:***

Задано матрицю дійсних чисел **A[m,n]**. Окремо у останньому рядку і першому стовпчику визначити присутність будь-якого з чисел діапазону **[0,5]** і його місцезнаходження (координати) методом двійкового пошуку (Алгоритм №1), якщо елементи цього рядка і стовпчика впорядковані за незбільшенням.

***Текст програми***

Для виконування цієї лабораторної було використано програму для матриці 8х10. Останній рядок та перший стовпець автоматично є відсортованими за незбільшенням. Ось приклад програми:

#include <stdio.h>

#include <stdlib.h>

#define LENGTH 10

#define SIZE 8

void print\_array(double matrix[SIZE][LENGTH]){

for (int row = 0; row < SIZE; row++) {

for (int column = 0; column < LENGTH; column++) {

printf("%.1lf\t", matrix[row][column]);

}

printf("\n");

}

printf("\n-----------------------------------------------------------------\n\n");

}

void binary\_search(double matrix[SIZE][LENGTH], int left\_border, int right\_border, int index, int min, int max, int position) {

int check = 0;

char \*text = (position) ? "last row" : "first column";

while (left\_border <= right\_border) {

int center = (right\_border + left\_border) / 2;

double element = (position) ? matrix[index][center] : matrix[center][index];

if (element < min) {

right\_border = center - 1;

}

else if (element > max) {

left\_border = center + 1;

}

else {

printf("The wanted element in the %s is %.1lf.", text, element);

printf(" It is on index: %d.\n", center);

check = 1;

break;

}

}

if (!check) {

printf("There is no such element in the %s.\n", text);

}

};

int main()

{

double matrix[SIZE][LENGTH] = {

{5.0, 3.1, -5.2, 2.3, 4.7, 0.9, 3.8, 15.4, 15.4, -1.6},

{4.5, 1.2, 0.4, -10.5, 3.2, 1.8, 4.6, 10.9, 8.9, 7.1},

{4.2, 0.5, -20.7, 3.3, 1.9, -5.4, 2.8, 5.3, 6.6, 1,7},

{4.2, 2.8, 0.2, -15.6, 4.5, 1.0, 0.7, 4.2, 9.2, 0.1},

{4.1, 3.7, 4.4, 0.1, 0.5, 2.6, -2.3, 3.9, 1.0, 17.5},

{3.3, -10.9, -5.7, -20.4, 1.3, 0.3, 3.9, 1.2, 3.4, -5.6},

{3.3, 4.3, 1.5, 0.7, 3.0, 4.2, -5.9, -3.7, 2.1, -7.2},

{3.3, 3.2, 3.1, 3.0, 3.0, 2.1, 2.0, 2.0, 1.0, 0.9}

};

print\_array(matrix);

int min = 0;

int max = 5;

int L = 0;

int R = LENGTH - 1;

int row = SIZE - 1;

int position = 1;

binary\_search(matrix, L, R, row, min, max, position);

L = 0;

R = SIZE - 1;

int column = 0;

position = 0;

binary\_search(matrix, L, R, column, min, max, position);

return 0;

}

***Результати тестування програми***

Для наглядності тестування матриці відбуватиметься 6 разів, а саме з допомогою зміни елементів в останньому рядку та першому стовпчику.

1) Усі елементи рядка й стовпця входять у діапазон.

{

{5.0, 3.1, -5.2, 2.3, 4.7, 0.9, 3.8, 15.4, 15.4, -1.6},

{4.5, 1.2, 0.4, -10.5, 3.2, 1.8, 4.6, 10.9, 8.9, 7.1},

{4.2, 0.5, -20.7, 3.3, 1.9, -5.4, 2.8, 5.3, 6.6, 1,7},

{4.2, 2.8, 0.2, -15.6, 4.5, 1.0, 0.7, 4.2, 9.2, 0.1},

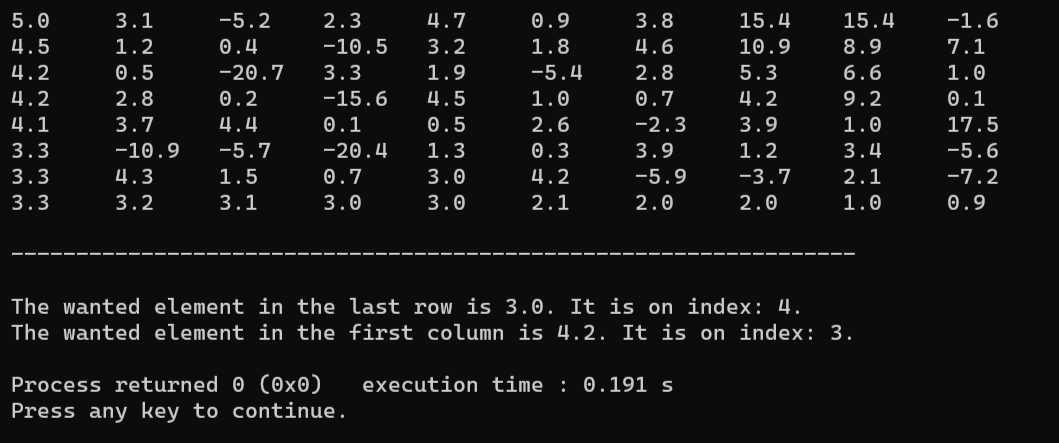
{4.1, 3.7, 4.4, 0.1, 0.5, 2.6, -2.3, 3.9, 1.0, 17.5},

{3.3, -10.9, -5.7, -20.4, 1.3, 0.3, 3.9, 1.2, 3.4, -5.6},

{3.3, 4.3, 1.5, 0.7, 3.0, 4.2, -5.9, -3.7, 2.1, -7.2},

{3.3, 3.2, 3.1, 3.0, 3.0, 2.1, 2.0, 2.0, 1.0, 0.9}

};



2) В останньому рядку та першому стовпці є тільки один (“спільний”) елемент, що входить у діапазон.

{

{15.0, 3.1, -5.2, 2.3, 4.7, 0.9, 3.8, 15.4, 15.4, -1.6},

{14.5, 1.2, 0.4, -10.5, 3.2, 1.8, 4.6, 10.9, 8.9, 7.1},

{14.3, 0.5, -20.7, 3.3, 1.9, -5.4, 2.8, 5.3, 6.6, 1,7},

{14.2, 2.8, 0.2, -15.6, 4.5, 1.0, 0.7, 4.2, 9.2, 0.1},

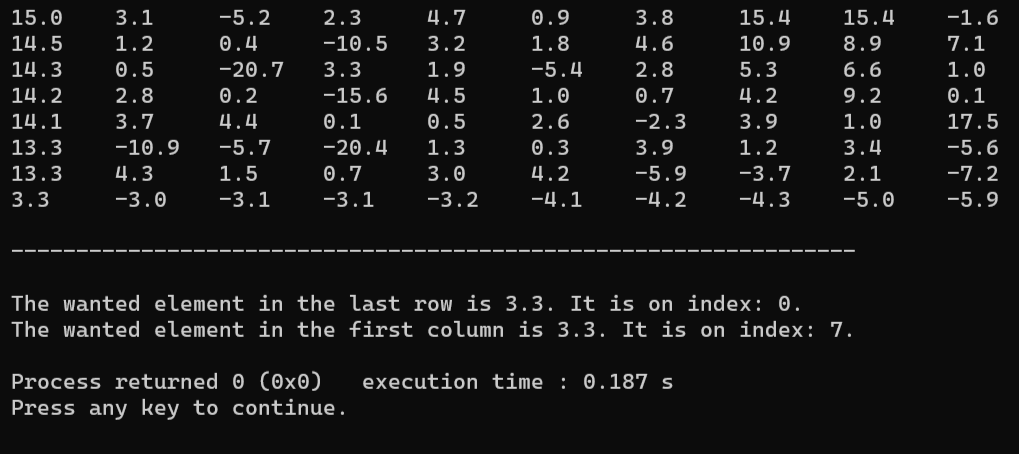
{14.1, 3.7, 4.4, 0.1, 0.5, 2.6, -2.3, 3.9, 1.0, 17.5},

{13.3, -10.9, -5.7, -20.4, 1.3, 0.3, 3.9, 1.2, 3.4, -5.6},

{13.3, 4.3, 1.5, 0.7, 3.0, 4.2, -5.9, -3.7, 2.1, -7.2},

{3.3, -3.0, -3.1, -3.1, -3.2, -4.1, -4.2, -4.3, -5.0, -5.9}

};



3) У першому стовпці немає елементів, що належать діапазону, а в останньому рядку – є (один елемент).

{

{15.0, 3.1, -5.2, 2.3, 4.7, 0.9, 3.8, 15.4, 15.4, -1.6},

{14.5, 1.2, 0.4, -10.5, 3.2, 1.8, 4.6, 10.9, 8.9, 7.1},

{14.3, 0.5, -20.7, 3.3, 1.9, -5.4, 2.8, 5.3, 6.6, 1,7},

{14.2, 2.8, 0.2, -15.6, 4.5, 1.0, 0.7, 4.2, 9.2, 0.1},

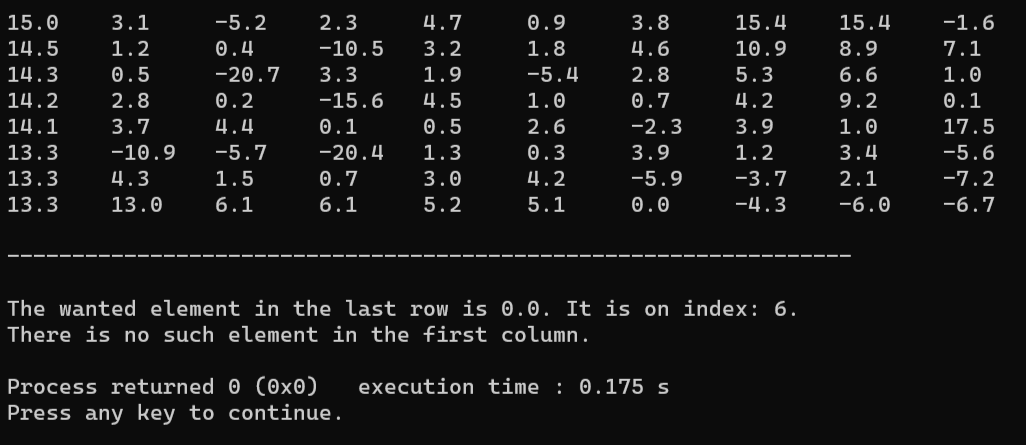
{14.1, 3.7, 4.4, 0.1, 0.5, 2.6, -2.3, 3.9, 1.0, 17.5},

{13.3, -10.9, -5.7, -20.4, 1.3, 0.3, 3.9, 1.2, 3.4, -5.6},

{13.3, 4.3, 1.5, 0.7, 3.0, 4.2, -5.9, -3.7, 2.1, -7.2},

{13.3, 13.0, 6.1, 6.1, 5.2, 5.1, 0.0, -4.3, -6.0, -6.7}

};



4) В останньому рядку немає елементів, що належать діапазону, а в першому стовпці – є.

{

{15.0, 3.1, -5.2, 2.3, 4.7, 0.9, 3.8, 15.4, 15.4, -1.6},

{4.5, 1.2, 0.4, -10.5, 3.2, 1.8, 4.6, 10.9, 8.9, 7.1},

{4.3, 0.5, -20.7, 3.3, 1.9, -5.4, 2.8, 5.3, 6.6, 1,7},

{-4.2, 2.8, 0.2, -15.6, 4.5, 1.0, 0.7, 4.2, 9.2, 0.1},

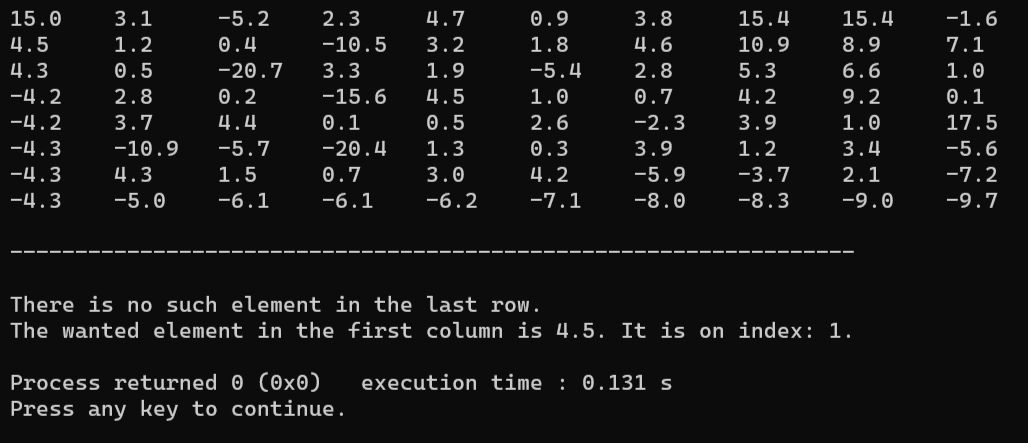
{-4.2, 3.7, 4.4, 0.1, 0.5, 2.6, -2.3, 3.9, 1.0, 17.5},

{-4.3, -10.9, -5.7, -20.4, 1.3, 0.3, 3.9, 1.2, 3.4, -5.6},

{-4.3, 4.3, 1.5, 0.7, 3.0, 4.2, -5.9, -3.7, 2.1, -7.2},

{-4.3, -5.0, -6.1, -6.1, -6.2, -7.1, -8.0, -8.3, -9.0, -9.7}

};



5) У першому стовпці й останньому рядку є декілька елементів, що входять у діапазон.

{

{15.0, 3.1, -5.2, 2.3, 4.7, 0.9, 3.8, 15.4, 15.4, -1.6},

{14.5, 1.2, 0.4, -10.5, 3.2, 1.8, 4.6, 10.9, 8.9, 7.1},

{14.3, 0.5, -20.7, 3.3, 1.9, -5.4, 2.8, 5.3, 6.6, 1,7},

{14.2, 2.8, 0.2, -15.6, 4.5, 1.0, 0.7, 4.2, 9.2, 0.1},

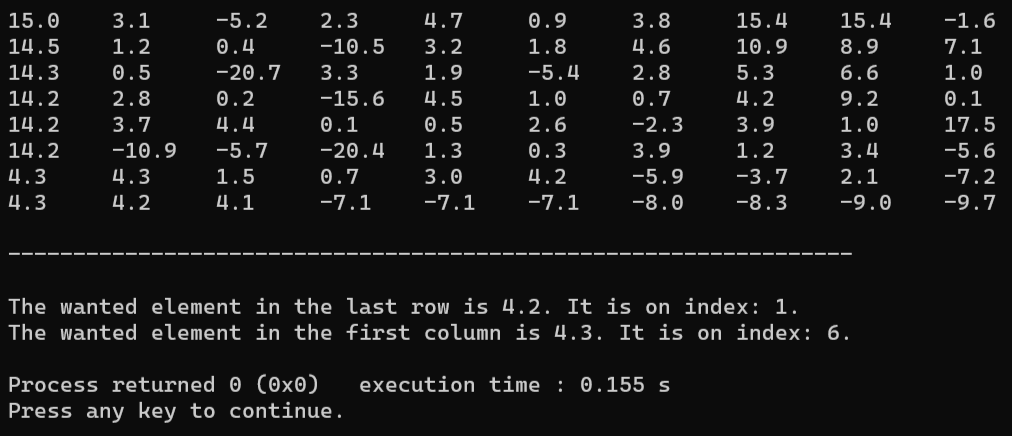
{14.2, 3.7, 4.4, 0.1, 0.5, 2.6, -2.3, 3.9, 1.0, 17.5},

{14.2, -10.9, -5.7, -20.4, 1.3, 0.3, 3.9, 1.2, 3.4, -5.6},

{4.3, 4.3, 1.5, 0.7, 3.0, 4.2, -5.9, -3.7, 2.1, -7.2},

{4.3, 4.2, 4.1, -7.1, -7.1, -7.1, -8.0, -8.3, -9.0, -9.7}

};



6)

{

{15.0, 3.1, -5.2, 2.3, 4.7, 0.9, 3.8, 15.4, 15.4, -1.6},

{14.5, 1.2, 0.4, -10.5, 3.2, 1.8, 4.6, 10.9, 8.9, 7.1},

{14.3, 0.5, -20.7, 3.3, 1.9, -5.4, 2.8, 5.3, 6.6, 1,7},

{14.2, 2.8, 0.2, -15.6, 4.5, 1.0, 0.7, 4.2, 9.2, 0.1},

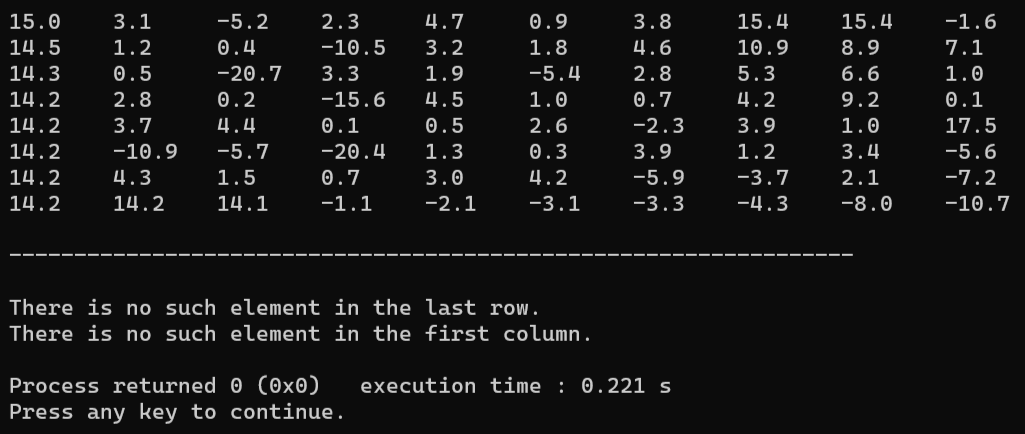
{14.2, 3.7, 4.4, 0.1, 0.5, 2.6, -2.3, 3.9, 1.0, 17.5},

{14.2, -10.9, -5.7, -20.4, 1.3, 0.3, 3.9, 1.2, 3.4, -5.6},

{14.2, 4.3, 1.5, 0.7, 3.0, 4.2, -5.9, -3.7, 2.1, -7.2},

{14.2, 14.2, 14.1, -1.1, -2.1, -3.1, -3.3, -4.3, -8.0, -10.7}

};



***Висновок:*** я зрозумів різницю лінійного та бінарного пошуків у двовимірному масиві; навчився використовувати Алгоритм №1 для знаходження елемента в певній частині масиву та його місцезнаходження; зрозумів, що таке незбільшення та незменшення з точки зору сортування для статичної матриці (двовимірного масиву); зрозумів перевагу бінарного (двійкового) пошуку, на відміну від лінійного: знайти елемент масиву можливо за меншу кількість операцій, що відповідно впливає на оптимізацію коду; тому використання двійкового пошуку більш рекомендоване, аніж лінійного (якщо, звичайно, масив/рядок (стовпець) масиву відсортовано).