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 № 1 з дисципліни
«Алгоритми та структури даних-1.
Основи алгоритмізації»

«Дослідження алгоритмів обходу

масивів»

Варіант 4

Виконав студент Бутов Даниіл Романович
(шифр, прізвище, ім'я, по батькові)

Перевірив Вечерковська Анастасія Сергіївна
(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.

Варіант 4

Завдання. Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом.

2. Ініціювання змінної, що описана в п.1 даного завдання.

3. Обчислення змінної, що описана в п.1, згідно з варіантом

№	Опис варіанту
4	Задано матрицю дійсних чисел A[m,n]. В кожному стовпчику матриці визначити присутність заданого дійсного числа X і його місцезнаходження. Обміняти знайдене значення X з елементом побічної діагоналі.

Постановка задачі.

Нам потрібно згенерувати двовимірний масив mxn дійсного типу, де потрібно знайти задане значення X. Результатом буде змінений масив де знайдене значення x буде обмінене зі значення побічної дігоналі на цьому стовбці.

Побудова математичної моделі.

Змінна	**Тип**	**Ім'я**	**Призначення**
Двовимірний масив	Дійсний	arr	Початкове
Розмір стовбців	Цілий	cSize	Початкове
Розмір рядка	Цілий	rSize	Початкове
Значення x	Дійсний	x	Початкове
Елемент побічної діагоналі	Дійсний	temp	Проміжні дані
Генерування масиву	Процедура	generate_arr	Початкове
Пошук місця x	Процедура	find_positions	Проміжні дані
Зміна масиву	Процедура	change_arr	Проміжні дані
Вивід масивів	Процедура	our_arr	Результат

arr[m][n] згенеруємо випадковим чином. Ми будемо брати дійсний проміжок [0; 1]. Генерування масиву випадковим чином ми будемо робити за допомогою: 0.01 * (rand() % 101) Наступним кроком буде перевірка на те, щоб знайти x у матриці. Як вказано у завдані ми будемо йти по стовбцям, тому зовніший цикл буде рахувати рядки. Елемент побічної діагоналі ми будемо знаходити за (rSize - i - 1), де i - лічильник рядків. Якщо матриця не квадратна, то кількість елементів у побічної діагоналі буде меньше за кількість елементів стовбця, тому потрібно буде робити перевірку, чи індекс побічної діагоналі не меньше за нуль. Останнім кроком буде виведення масиву, у якому вже змінені значення. Також нам потрібно знайти місцезнаходження x у матриці, тому будемо виводити його індекси.

Розв'язання.

Програмні специфікації запишемо у псевдокоді та графічній формі у вигляді блок-схеми.

Крок 1. Визначемо основні дії.

Крок 2. Деталізуємо генерування масиву arr[m][n]

Крок 3. Деталізуємо місцезнаходження x.

Крок 4. Деталізуємо пошуку змінної x та заміни її з побічною діагональю.

Крок 5. Деталізуємо вивід масиву.

Псевдокод.

Основна программа.

Початок

Введення cSize, rSize, x

generate_arr(arr, rSize, cSize)

out_arr(arr, r Size, cSize)

find_positions (arr, rSize, cSize, x)

change_arr(arr, rSize, cSize, x)

out_arr(arr, r Size, cSize)

Кінець

generate_arr(arr, rSize, cSize)

Початок

для i **до** cSize **повторити**

для j **до** rSize **повторити**

array[i][j] = 0.01 * (rand() % 101)

все повторити

все повторити

Кінець

out_arr(array, sorted_array, size)

Початок

для i **до** cSize **повторити**

для j **до** rSize **повторити**

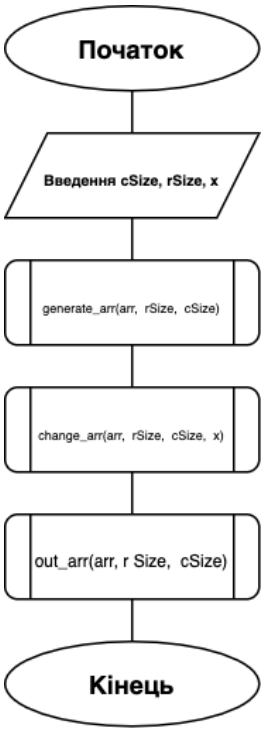
```
                виведення arr[i][j]
            все повторити
        все повторити
Кінець

change_arr(arr, rSize, cSize, x)
Початок
        для i до rSize повторити
            temp = arr[i][rSize - i - 1]
            для j до cSize повторити
                якщо x == arr[i][j] and rSize - i - 1 >= 0
                    то
                        arr[i][rSize - i - 1] = arr[i][j]
                        arr[i][j] = temp
                        temp = arr[i][rSize - i - 1]
                все якщо
            все повторити
        все повторити
Кінець

find_positions (arr, rSize, cSize, x)
Початок
        для i до cSize повторити
            для j до rSize повторити
                якщо x == arr[i][j]
                    то
                      виведення i та j
            все повторити
        все повторити
Кінець
```

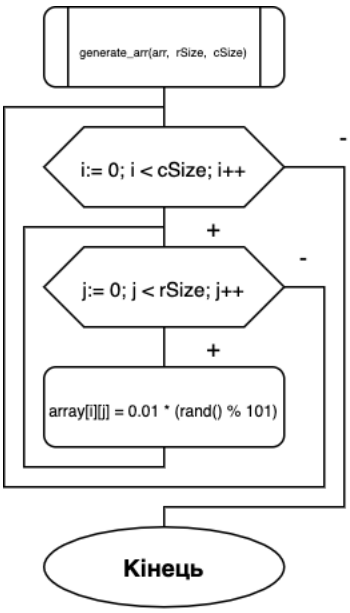
Блоксхема

Основна програма.

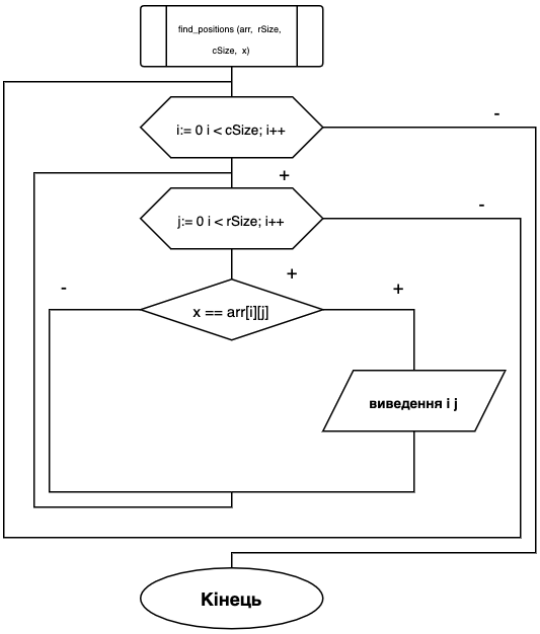


Підпрограми.

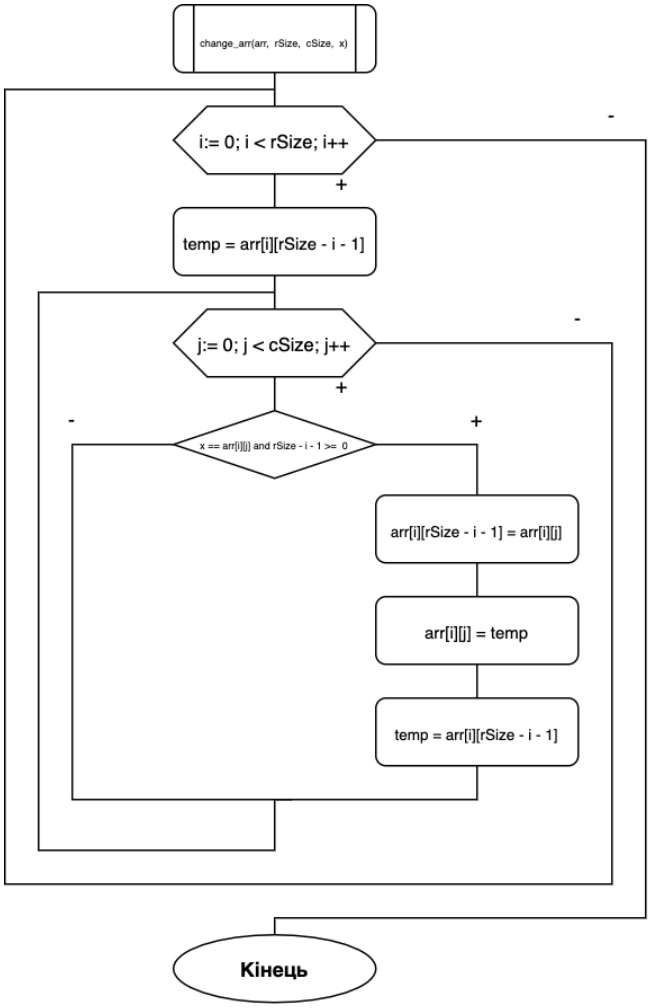
generate_arr(arr, rSize, cSize)



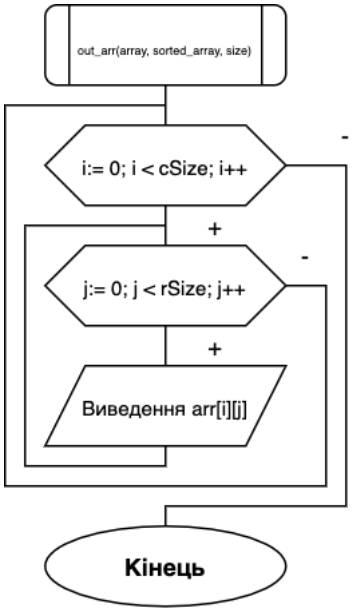
find_positions(arr, rSize, cSize, x)



change_arr(arr, rSize, cSize, x)



out_arr(array, sorted_array, size)



Випробування

```
Enter raw of matrix: 3
Enter column of matrix: 3
Enter x: 0.410
 0.41 0.65 0.31
 0.41 0.19 0.15
 0.72 0.11 0.78

Position of x: 0 0

Position of x: 1 0


 0.31 0.65 0.41
 0.19 0.41 0.15
 0.72 0.11 0.78


Process finished with exit code 0
```

Код

```cpp
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

void generate_arr(float** arr, int rSize, int cSize); //Генерация массива
void change_arr(float** arr, int rSize, int cSize, float x); //Смена массива
void out_arr(float** arr, int rSize, int cSize); //Вывод массива
void find_positions(float** arr, int rSize, int cSize, float x); //  Поиск иксов
int main(){
    int cSize, rSize;
    float x;
    //Ввод значений и создание массива
    cout << "Enter raw of matrix: "; cin >> cSize;
    cout << "Enter column of matrix: "; cin >> rSize;
    cout << "Enter x: "; cin >> x;
    float **arr = new float* [cSize];
    for (int i = 0; i < cSize; i++)
        arr[i] = new float [rSize];

    //Вызов функции
    generate_arr(arr, rSize, cSize);
    out_arr(arr, rSize, cSize);
    find_positions(arr, rSize, cSize, x);
    change_arr(arr, rSize, cSize, x);
    out_arr(arr, rSize, cSize);

    //Удаление массива
    for (int i = 0; i < rSize; i++)
        delete []arr[i];
}

void generate_arr(float** arr, int rSize, int cSize) {
    for (int i = 0; i < cSize; ++i) {
        for (int j = 0; j < rSize; ++j) arr[i][j] = 0.01 * (rand() % 101);
    }
}
void find_positions(float** arr, int rSize, int cSize, float x){
    for (int i = 0; i < rSize; ++i) {
        for (int j = 0; j < cSize; ++j) {
            if (x == arr[i][j]) cout << "Position of x: " << i << " " << j << endl;
        }
        cout << endl;
    }
}
void change_arr(float** arr, int rSize, int cSize, float x) {
    float temp;
    for (int i = 0; i < rSize; ++i) {
        temp = arr[i][rSize - i - 1];
        for (int j = 0; j < cSize; ++j) {
            if (x == arr[i][j] && rSize - i - 1 >= 0) {
                arr[i][rSize - i - 1] = arr[i][j];
                arr[i][j] = temp;
                temp = arr[i][rSize - i - 1];
            }
        }
    }
}

void out_arr(float** arr, int rSize, int cSize){
    for (int i = 0; i < cSize; ++i) {
        for (int j = 0; j < rSize; ++j) cout << setw( n: 5) << arr[i][j];
        cout << endl;
    }
    cout << endl;
}
```

Висновок

Ми дослідили алгоритми обходу масивів, набули практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Склали алгоритм раціонального знаходження заданого значення в матриці та заміною його з елементом побічної діагоналі. Засвоїли на практиці розробку алгоритму для завдань такого типу.