

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 Даниїл Романович
(шифр, прізвище, ім'я, по батькові)

Перевірів Вечерковська Анастасія Сергіївна
(прізвище, ім'я, по батькові)

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 $m \times n$ 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 |
| 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 завданні ми будемо йти по стовбцям, тому зовнішній 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 менше за кількість елементів стовбця, тому потрібно буде робити перевірку, чи індекс побічної діагоналі не менше за нуль. Останнім кроком буде виведення масиву, у якому вже змінені значення.

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 пошуку змінної `x` та заміни її з побічною діагоноюлю.

Крок 4. Деталізуємо вивід масиву.

Псевдокод.

Основна програма.

Початок

Введення `cSize, rSize, x`

`generate_arr(arr, rSize, cSize)`

`change_arr(arr, rSize, cSize, x)`

`out_arr(arr, rSize, 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

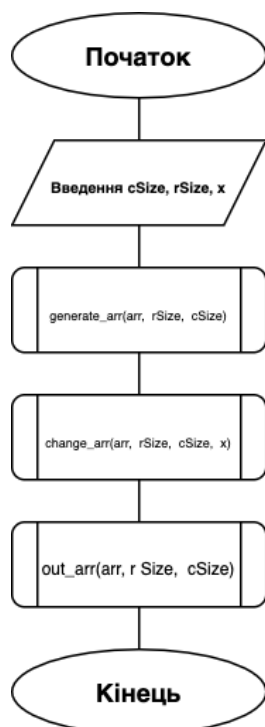
все повторити

все повторити

Кінець

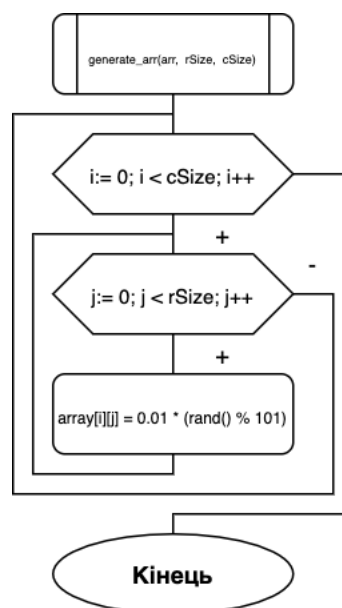
Блоксхема

Основна програма.

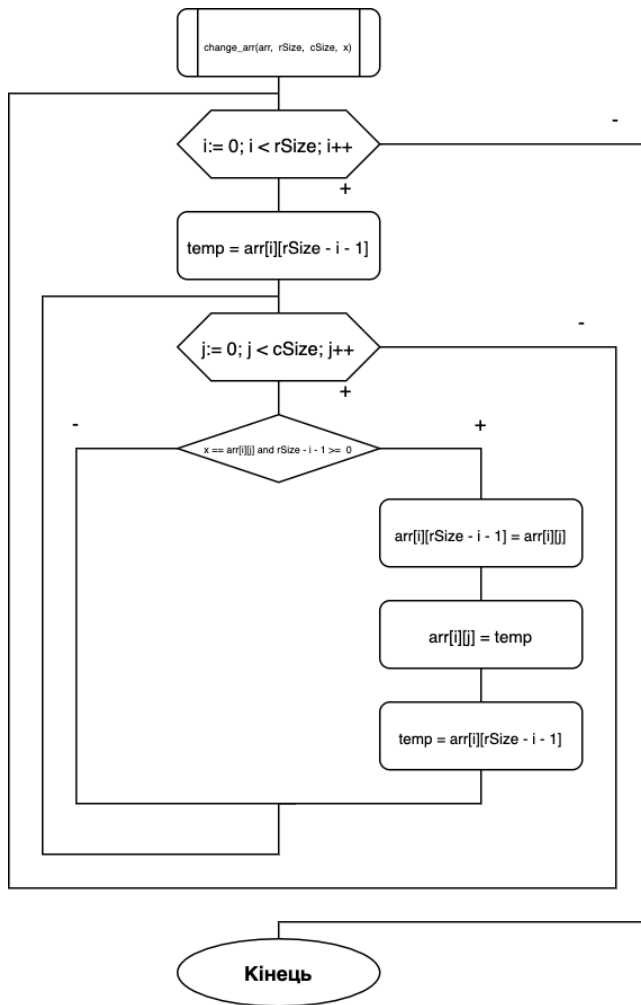


Підпрограми.

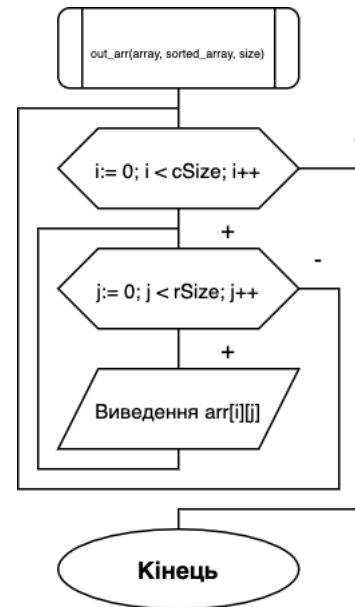
generate_arr(arr, rSize, cSize)



change_arr(arr, rSize, cSize, x)



out_arr(array, sorted_array, size)



Випробування

```

Enter raw of matrix: 4
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
0.69 0.37 0.23
    
```

```

0.31 0.65 0.41
0.19 0.41 0.15
0.72 0.11 0.78
0.69 0.37 0.23
    
```

Process finished with exit code 0

```

Enter raw of matrix: 2
Enter column of matrix: 2
Enter x: 0.410
0.41 0.65
0.31 0.41

0.65 0.41
0.41 0.31
    
```

Process finished with exit code 0

Код

```

1  #include <iostream>
2  #include <iomanip>
3  #include <cmath>
4  using namespace std;
5
6  void generate_arr(float** arr, int rSize, int cSize); //Генерація масива
7  void change_arr(float** arr, int rSize, int cSize, float x); //Смена масива
8  void out_arr(float** arr, int rSize, int cSize); //Вывод массива
9
10 int main(){
11     int cSize, rSize;
12     float x;
13     //Ввод значений и создание массива
14     cout << "Enter row of matrix: "; cin >> cSize;
15     cout << "Enter column of matrix: "; cin >> rSize;
16     cout << "Enter x: "; cin >> x;
17     float **arr = new float* [cSize];
18     for (int i = 0; i < cSize; i++)
19         arr[i] = new float [rSize];
20
21     //Вызов функции
22     generate_arr(arr, rSize, cSize);
23     out_arr(arr, rSize, cSize);
24     change_arr(arr, rSize, cSize, x);
25     out_arr(arr, rSize, cSize);
26
27     //Удаление массива
28     for (int i = 0; i < rSize; i++)
29         delete []arr[i];
30 }
31
32 void generate_arr(float** arr, int rSize, int cSize) {
33     srand(time(nullptr));
34     for (int i = 0; i < cSize; ++i) {
35         for (int j = 0; j < rSize; ++j) arr[i][j] = 0.01 * (rand() % 101);
36     }
37 }
38
39 void change_arr(float** arr, int rSize, int cSize, float x) {
40     float temp;
41     for (int i = 0; i < rSize; ++i) {
42         temp = arr[i][rSize - i - 1];
43         for (int j = 0; j < cSize; ++j) {
44             if (x == arr[i][j] && rSize - i - 1 >= 0) {
45                 arr[i][rSize - i - 1] = arr[i][j];
46                 arr[i][j] = temp;
47                 temp = arr[i][rSize - i - 1];
48             }
49         }
50     }
51 }
52
53 void out_arr(float** arr, int rSize, int cSize){
54     for (int i = 0; i < cSize; ++i) {
55         for (int j = 0; j < rSize; ++j) {
56             cout << setw(5) << arr[i][j];
57         }
58         cout << endl;
59     }
60     cout << endl;
61 }

```

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.