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 № 8 з
дисципліни «Алгоритми та структури даних-1. Основи алгоритмізації»
«Дослідження алгоритмів пошуку та сортування»
Варіант 2

Виконав студент ІП-12, Басараб Олег Андрійович
(шифр, прізвище, ім'я, по батькові)

Перевірив _____
(прізвище, ім'я, по батькові)

Київ 2021

Лабораторна робота № 8 “ Дослідження алгоритмів пошуку та сортування ”

Варіант 2

Мета – дослідити алгоритми пошуку та сортування, набути практичних навичок використання цих алгоритмів під час складання програмних специфікацій.

Задача 2. Розробити алгоритм та написати програму, яка складається з наступних дій:

1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом (масив дійсних чисел розмірністю 6 x 5).
2. Ініціювання змінної, що описана в п. 1 даного завдання.
3. Створення нової змінної індексованого типу (одновимірний масив) та її ініціювання значеннями, що обчислюються згідно з варіантом (елементами масиву є середні арифметичні значення елементів рядків двовимірного масиву, відсортовані обміном за спаданням).

Розв'язування.

Постановка задачі. Результатом розв’язку є дійсний одновимірний масив newArr1D[] розмірністю ROWS_NUM, заповнений відповідно до умови. Для його знаходження вхідні дані не потрібні. Початковими даними є дві константі цілі змінні ROWS_NUM = 6, COLUMNS_NUM = 5, дійсний двовимірний масив arr2D[][] розмірністю ROWS_NUM x COLUMNS_NUM, дійсний одновимірний масив arr1D[] розмірністю ROWS_NUM, дійсний одновимірний масив newArr1D[] розмірністю ROWS_NUM. Під час розв’язування буде використано функцію rand(), що повертає випадкове дійсне число з проміжку [-100.0; 100.0]. Вважаємо, що масиви передаються до підпрограм за посиланням, запис float** arr2D означає посилання на дійсний двовимірний масив, float* arr1D – посилання на дійсний одновимірний масив. Також вважаємо, що заголовок арифметичного циклу виду «для і від 1 до n» означає, що і – лічильник циклу з початковим значенням 1, ($i \le n$) – умова невиходу, а крок = 1.

Побудова математичної моделі. Складемо таблицю імен змінних основної програми.

Змінна	Тип	Ім’я	Призначення
Кількість рядків двовимірного масиву	Цілий (константа)	ROWS_NUM	Початкове дане
Кількість стовбців двовимірного масиву	Цілий (константа)	COLUMNS_NUM	Початкове дане
Двовимірний масив	Дійсний двовимірний масив	arr2D [] []	Початкове дане
Перший одновимірний масив	Дійсний одновимірний масив	arr1D []	Початкове дане
Другий одновимірний масив	Дійсний одновимірний масив	newArr1D []	Результат

Складемо таблицю імен змінних підпрограми getConditional1DArray для задання елементів одновимірного масиву arr1D.

Змінна	Тип	Ім’я	Призначення
Кількість рядків двовимірного масиву	Цілий (константа)	ROWS_NUM	Вхідне дане
Кількість стовбців двовимірного масиву	Цілий (константа)	COLUMNS_NUM	Вхідне дане
Двовимірний масив, на основі елементів якого задаватимуться елементи одновимірного масиву	Дійсний двовимірний масив	arr2D [] []	Вхідне дане

Лічильник арифметичного циклу	Цілий	i	Поточне дане
Лічильник арифметичного циклу	Цілий	j	Поточне дане
Середнє арифметичне елементів рядків двовимірного масиву	Дійсний	rowAverage	Поточне дане
Результуючий одновимірний масив	Дійсний одновимірний масив	arr1D []	Результат

Складемо таблицю імен змінних підпрограми getRandomReal2DArray для задання елементів двовимірного масиву arr2D.

Змінна	Тип	Ім’я	Призначення
Кількість рядків двовимірного масиву	Цілий (константа)	ROWS_NUM	Вхідне дане
Кількість стовбців двовимірного масиву	Цілий (константа)	COLUMNS_NUM	Вхідне дане
Лічильник арифметичного циклу	Цілий	i	Поточне дане
Лічильник арифметичного циклу	Цілий	j	Поточне дане
Результуючий двовимірний масив, елементами якого є випадкові дійсні числа	Дійсний двовимірний масив	arr2D [] []	Результат

Складемо таблицю імен змінних підпрограми copyArray1D для копіювання елементів одного одновимірного масиву в другий одновимірний масив.

Змінна	Тип	Ім’я	Призначення
Масив, елементи якого копіюються	Дійсний одновимірний масив	arr1 []	Вхідне дане
Масив, елементи якого є скопійованими	Дійсний одновимірний масив	arr2 []	Вхідне дане
Розмір одновимірних масивів arr1 і arr2	Цілий (константа)	ARR_SIZE	Вхідне дане
Лічильник арифметичного циклу	Цілий	i	Поточне дане

Складемо таблицю імен змінних підпрограми bubbleSort для сортування елементів масиву newArr1D.

Змінна	Тип	Ім’я	Призначення
Масив, елементи якого буду сортуватися	Дійсний масив	arr []	Вхідне дане
Розмір масиву arr	Цілий (константа)	ARR_SIZE	Вхідне дане
Лічильник арифметичного циклу	Цілий	i	Поточне дане
Лічильник арифметичного циклу	Цілий	j	Поточне дане
Змінна для обміну місцями елементів одновимірного масиву	Дійсний	temp	Поточне дане

Таким чином, формулювання завдання зводить до:

1. **Опису підпрограми float** getRandomReal2DArray(const int ROWS_NUM, const int COLUMNS_NUM) для задання елементів двовимірного масиву arr2D.** В системі з двох арифметичних циклів (для i від 1 до ROWS_NUM – заголовок зовнішнього циклу, для j від 1 до COLUMNS_NUM – заголовок внутрішнього циклу), в тілі внутрішнього циклу, знаходимо елемент arr[i][j] = rand(). Підпрограма повертає двовимірний масив arr2D.
2. **Опису підпрограми float* getConditional1DArray(float** arr2D, const int ROWS_NUM, const int COLUMNS_NUM) для задання елементів одновимірного масиву arr1D.** В системі з двох арифметичних циклів (для i від 1 до ROWS_NUM – заголовок зовнішнього циклу, для j від 1 до COLUMNS_NUM – заголовок внутрішнього циклу), в тілі зовнішнього циклу, відбувається ініціалізація rowAverage = 0, запуск та завершення виконання внутрішнього циклу, виконання таких дій: rowAverage /= COLUMN_NUM, arr1D[i] = rowAverage. В тілі внутрішнього циклу, виконується дія rowAverage += arr2D[i][j]. Підпрограма повертає одновимірний масив arr1D.
3. **Опису підпрограми void copyArray1D(float* arr1, float* arr2, const int ARR_SIZE) для копіювання елементів одного одновимірного масиву в другий одновимірний масив.** В тілі арифметичного циклу з заголовком «для j від 1 до ARR_SIZE» виконується дія arr2[i] = arr1[i].
4. **Опису підпрограми void bubbleSort(float* arr, const int ARR_SIZE) для сортування елементів масиву arr за спаданням.** В системі з двох арифметичних циклів (для i від 1 до ROWS_NUM - 1 – заголовок зовнішнього циклу, для j від 1 до COLUMNS_NUM – i - 1 – заголовок внутрішнього циклу), в тілі внутрішнього циклу, якщо arr[j] < arr[j + 1], то виконуються наступні дії: ініціалізація змінної temp = arr[j], arr[j] = arr[j + 1], arr[j + 1] = temp.

5. **Опису основного алгоритму.** Вважаємо, що ROWS_NUM = 6, COLUMNS_NUM = 5 відповідно до умови. Задаємо двовимірний масив arr2D, одновимірний масив arr1D, одновимірний масив newArr1D за допомогою підпрограм getRandomReal2DArray, getConditional1DArray, copyArray1D відповідно. Обробляємо newArr1D за допомогою bubbleSort та виводимо його.

Розіб’ємо алгоритм роботи основної програми на кроки:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію задання елементів масивів arr2D, arr1D та newArr1D за допомогою власних підпрограм.

Крок 3. Деталізуємо дію обробки елементів масиву newArr1D за допомогою bubbleSort.

Розіб’ємо алгоритм роботи підпрограми float** getRandomReal2DArray(const int ROWS_NUM, const int COLUMNS_NUM) для задання елементів двовимірного масиву arr2D:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію знаходження елементів arr2D за допомогою системи двох арифметичних циклів.

Розіб’ємо алгоритм роботи float* getConditional1DArray(float** arr2D, const int ROWS_NUM, const int COLUMNS_NUM) для задання елементів одновимірного масиву arr1D:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію знаходження елементів arr1D за допомогою системи двох арифметичних циклів.

Розіб’ємо алгоритм роботи підпрограми void copyArray1D(float* arr1, float* arr2, const int ARR_SIZE) для копіювання елементів одного одновимірного масиву в другий одновимірний масив:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію знаходження елементів масиву arr2 за допомогою арифметичного циклу.

Розіб’ємо алгоритм роботи підпрограми void bubbleSort(float* arr, const int ARR_SIZE) для сортування елементів масиву arr за спаданням:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію задання системи вкладених циклів.

Крок 3. Деталізуємо дію обміну двох елементів масиву в разі виконання необхідної умови.

Програмні специфікації зобразимо у псевдокоді та в графічній формі в блок-схемі алгоритму.

Псевдокод.

Основна програма:

Крок 1

початок

задання елементів масивів arr2D, arr1D та newArr1D за допомогою власних підпрограм

обробка елементів масиву newArr1D за допомогою bubbleSort

вивід newArr1D

кінець

Крок 2

початок

arr2D = getRandomReal2DArray(ROWS_NUM, COLUMNS_NUM)

arr1D = getConditional1DArray(arr2D, ROWS_NUM, COLUMNS_NUM)

copyArray1D(arr1D, newArr1D, ROWS_NUM)

обробка елементів масиву newArr1D за допомогою bubbleSort

вивід newArr1D

кінець

Крок 3

початок

arr2D = getRandomReal2DArray(ROWS_NUM, COLUMNS_NUM)

arr1D = getConditional1DArray(arr2D, ROWS_NUM, COLUMNS_NUM)

copyArray1D(arr1D, newArr1D, ROWS_NUM)

bubbleSort(newArr1D, ROWS_NUM)

вивід newArr1D

кінець

Підпрограма float getRandomReal2DArray(const int ROWS_NUM, const int COLUMNS_NUM):**

Крок 1

початок

знаходження елементів arr2D за допомогою системи двох арифметичних циклів

повернути arr2D

кінець

Крок 2

початок

повторити

для i **від** 1 **до** ROWS_NUM

повторити

для j **від** 1 **до** COLUMNS_NUM

arr2D[i][j] = rand()

все повторити

все повторити

повернути arr2D

кінець

Підпрограма float* getConditional1DArray(float arr2D, const int ROWS_NUM, const int COLUMNS_NUM):**

Крок 1

початок

знаходження елементів arr1D за допомогою системи двох арифметичних циклів

повернути arr1D

кінець

Крок 2

початок

повторити

для i **від** 1 **до** ROWS_NUM

повторити

rowAverage = 0

для j **від** 1 **до** COLUMNS_NUM

rowAverage += arr2D[i][j]

rowAverage /= COLUMNS_NUM

arr1D[i] = rowAverage

все повторити

все повторити

повернути arr1D

кінець

Підпрограма void copyArray1D(float* arr1, float* arr2, const int ARR_SIZE):

Крок 1

початок

знаходження елементів масиву arr2 за допомогою арифметичного циклу

кінець

Крок 2

початок

повторити

для i **від** 1 **до** ARR_SIZE

arr2[i] = arr1[i]

все повторити

кінець

Підпрограма void bubbleSort(float* arr, const int ARR_SIZE):

Крок 1

початок

задання системи вкладених циклів

кінець

Крок 2

початок

повторити

для i **від** 1 **до** ARR_SIZE - 1

повторити

для j **від** 1 **до** ARR_SIZE - i – 1

обмін двох елементів масиву в разі виконання необхідної умови

все повторити

все повторити

кінець

Крок 3

початок

повторити

для i **від** 1 **до** ARR_SIZE - 1

повторити

для j **від** 1 **до** ARR_SIZE - i – 1

якщо arr[j] < arr[j + 1]

то

temp = arr[j]

arr[j] = arr[j + 1]

arr[j + 1] = temp

все якщо

все повторити

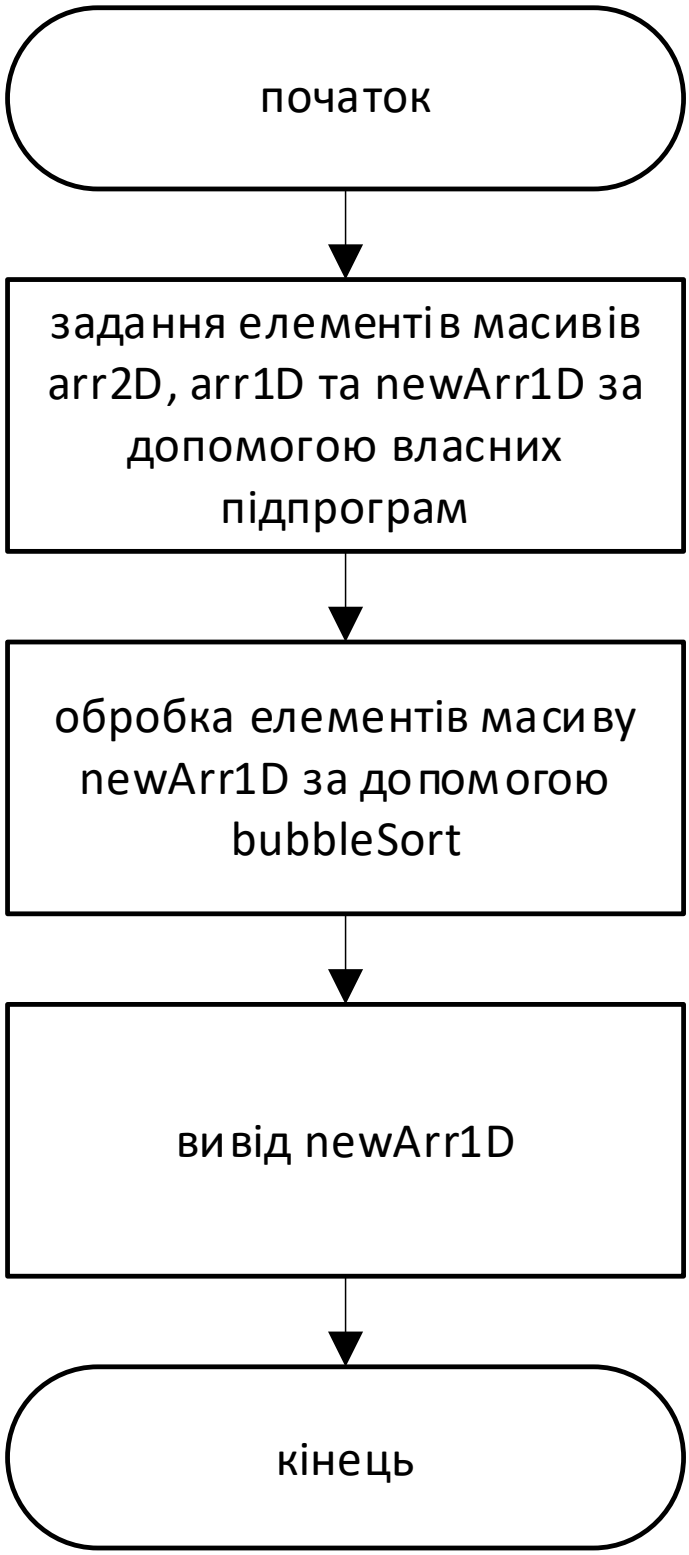
все повторити

кінець

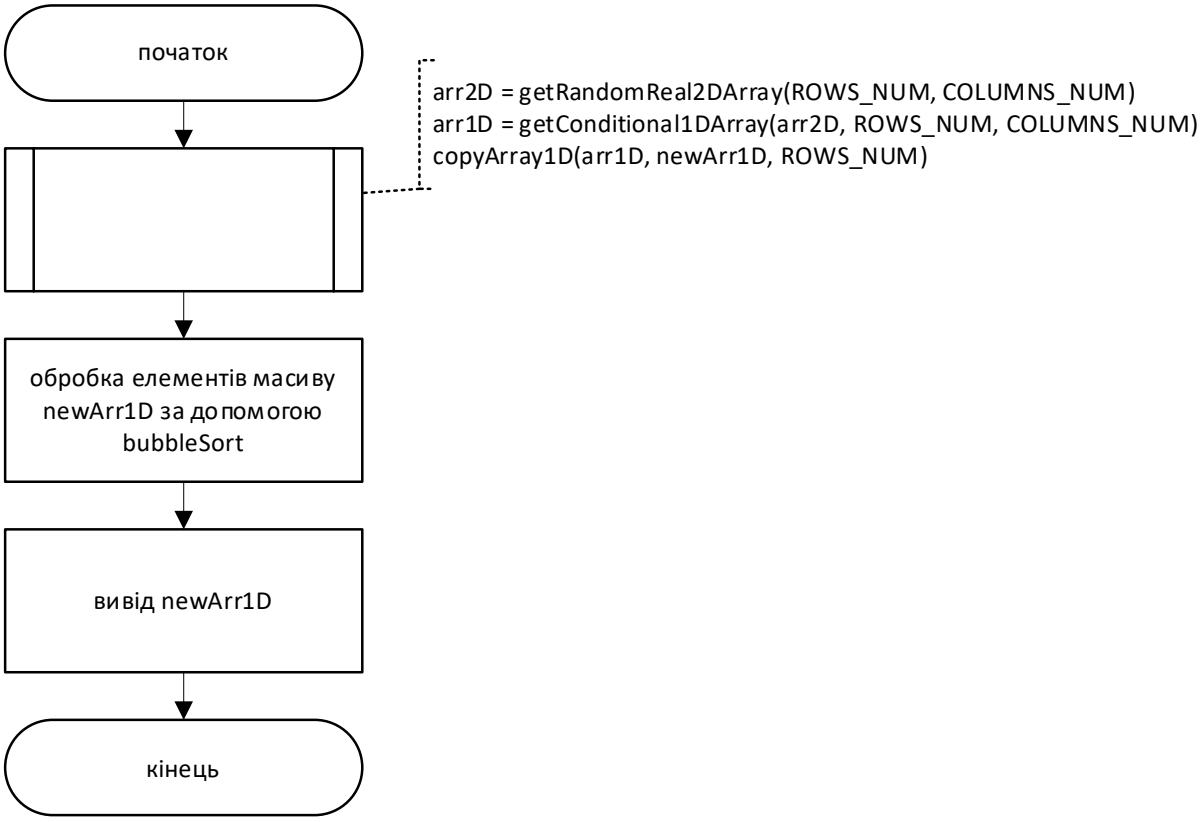
Блок-схема алгоритму.

Основна програма:

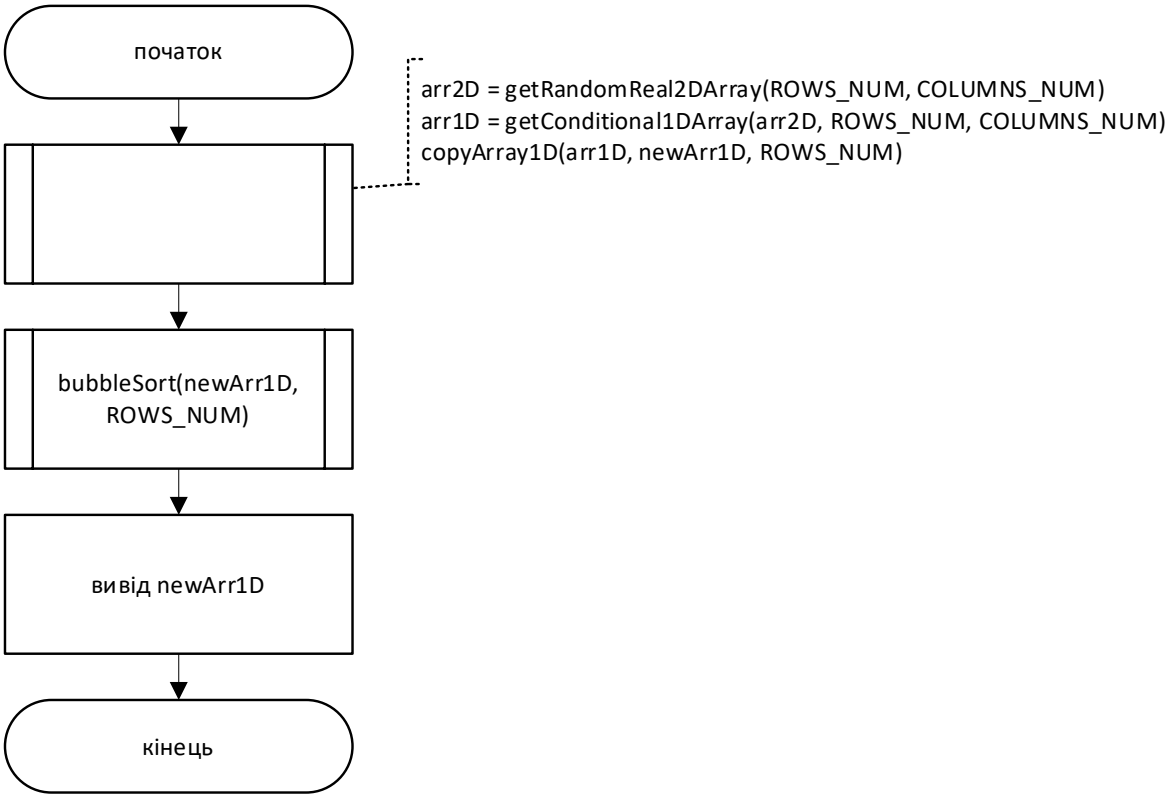
Крок 1



Крок 2

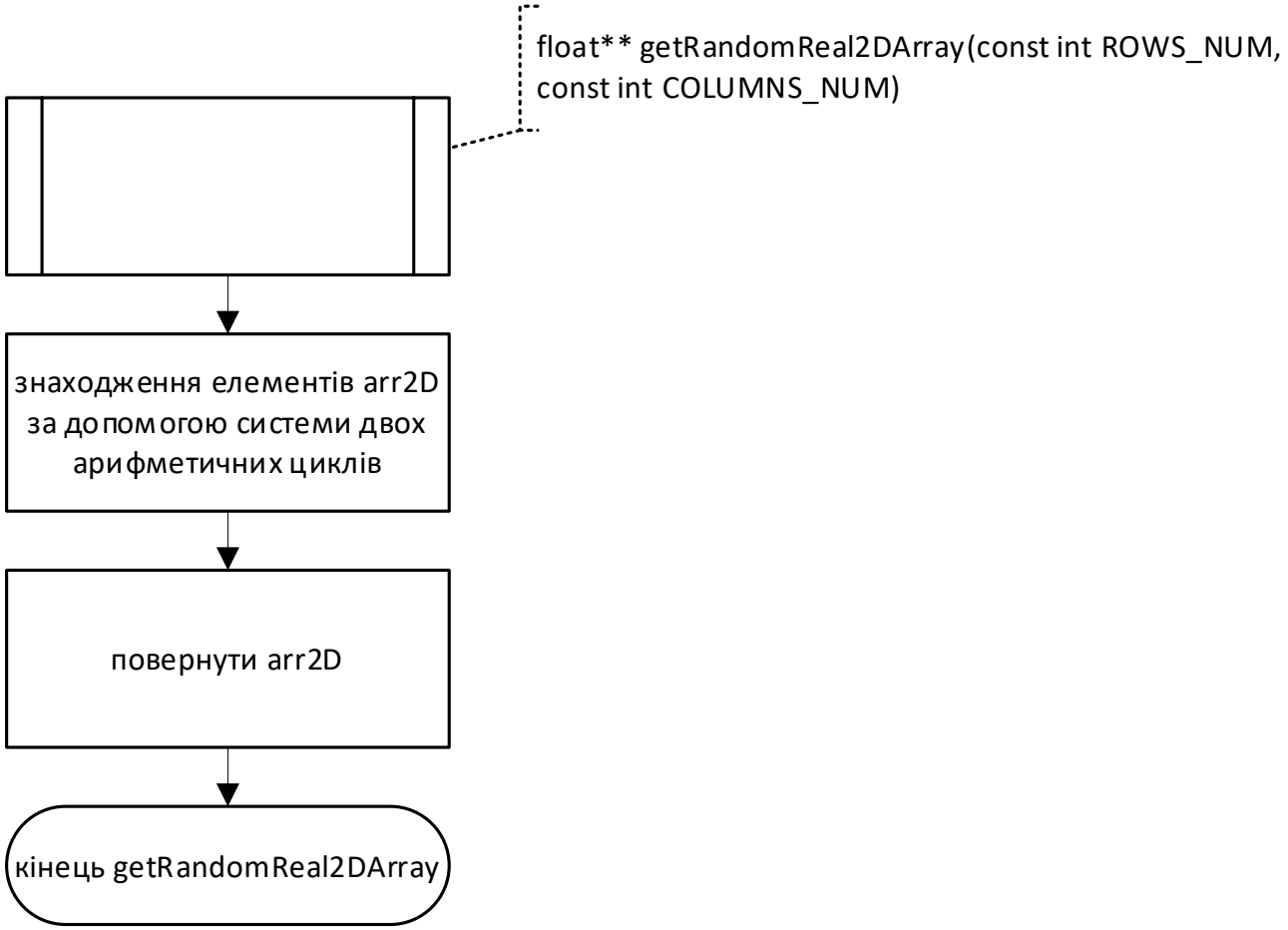

початок
arr2D = getRandomReal2DArray(ROWS_NUM, COLUMNS_NUM)
arr1D = getConditional1DArray(arr2D, ROWS_NUM, COLUMNS_NUM)
copyArray1D(arr1D, newArr1D, ROWS_NUM)
обробка елементів масиву newArr1D за допомогою bubbleSort
вивід newArr1D
кінець

Крок 3

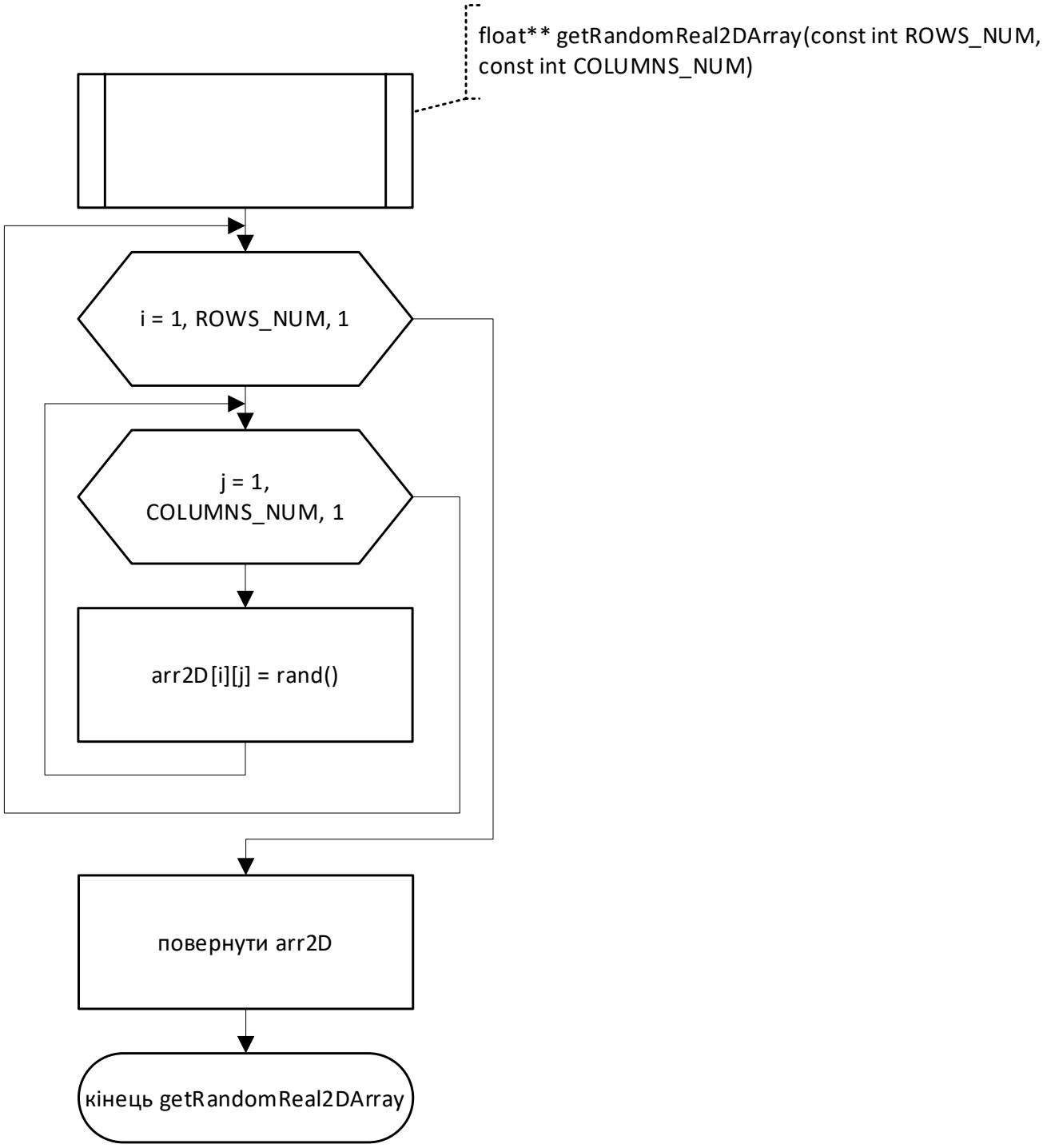

початок
arr2D = getRandomReal2DArray(ROWS_NUM, COLUMNS_NUM)
arr1D = getConditional1DArray(arr2D, ROWS_NUM, COLUMNS_NUM)
copyArray1D(arr1D, newArr1D, ROWS_NUM)
bubbleSort(newArr1D, ROWS_NUM)
вивід newArr1D
кінець

Підпрограма float getRandomReal2DArray(const int ROWS_NUM, const int COLUMNS_NUM):**

Крок 1

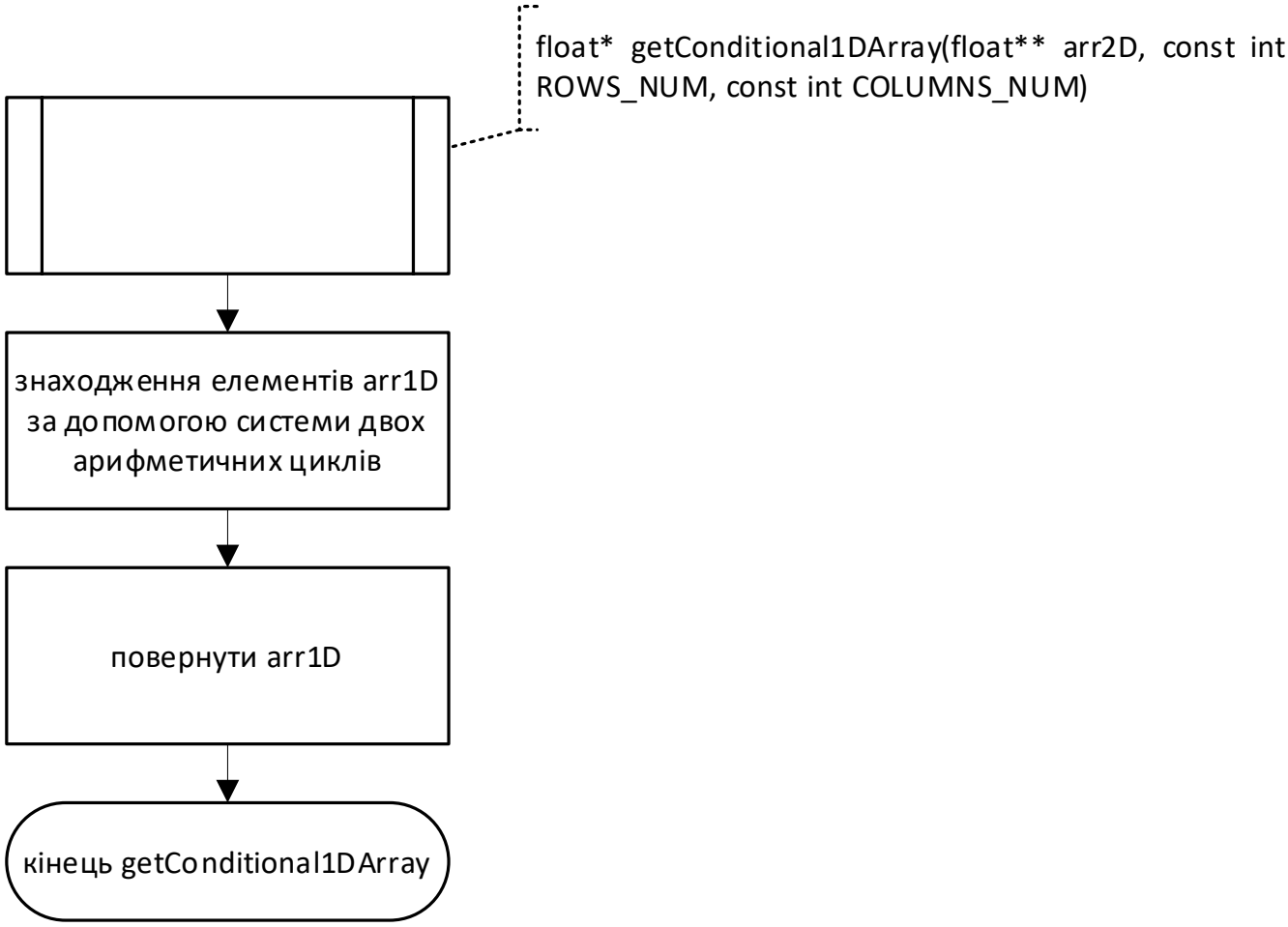


Крок 2

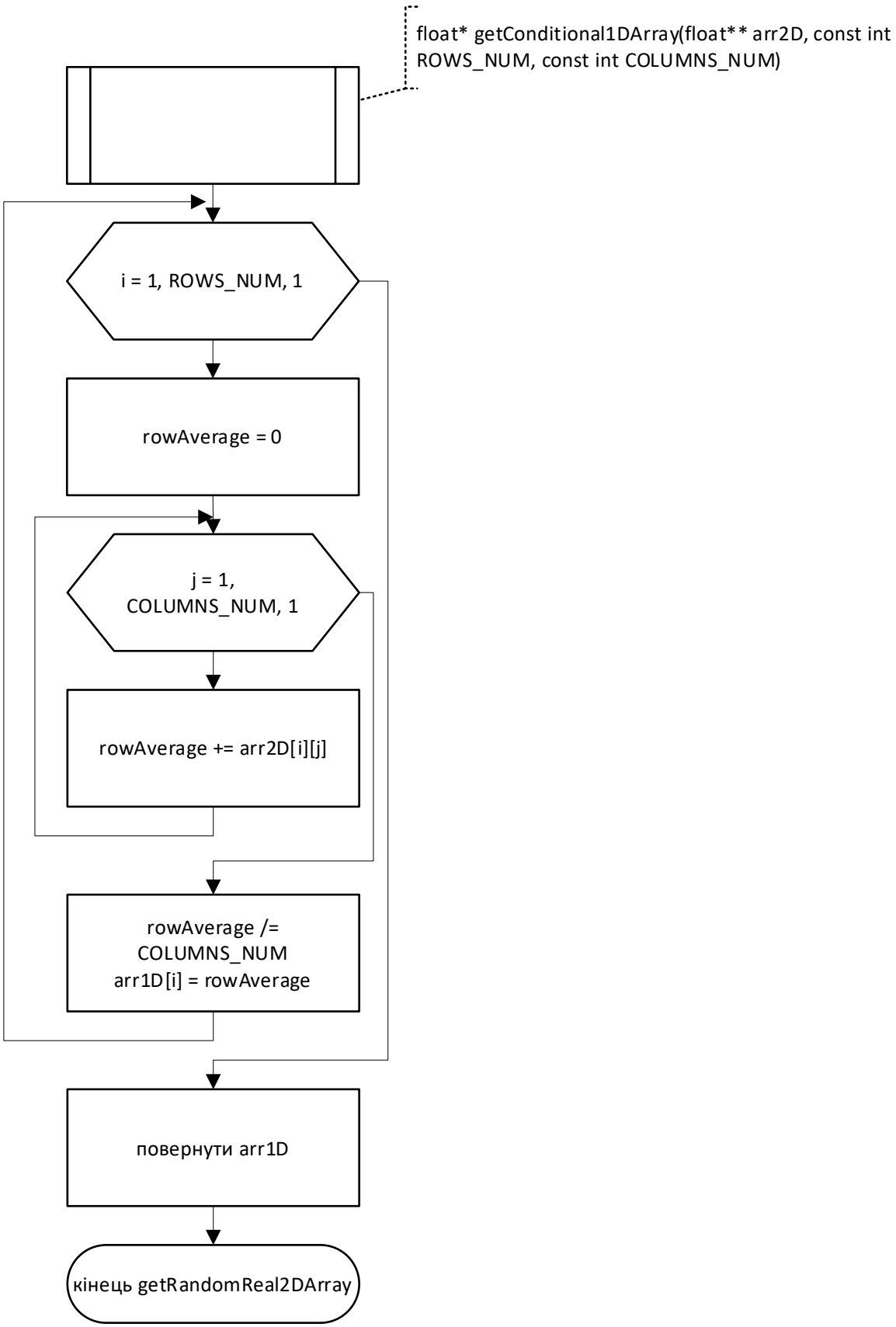

float** getRandomReal2DArray(const int ROWS_NUM, const int COLUMNS_NUM)
i = 1, ROWS_NUM, 1
j = 1, COLUMNS_NUM, 1
arr2D[i][j] = rand()
повернути arr2D
кінець getRandomReal2DArray

Підпрограма float* getConditional1DArray(float arr2D, const int ROWS_NUM, const int COLUMNS_NUM):**

Крок 1

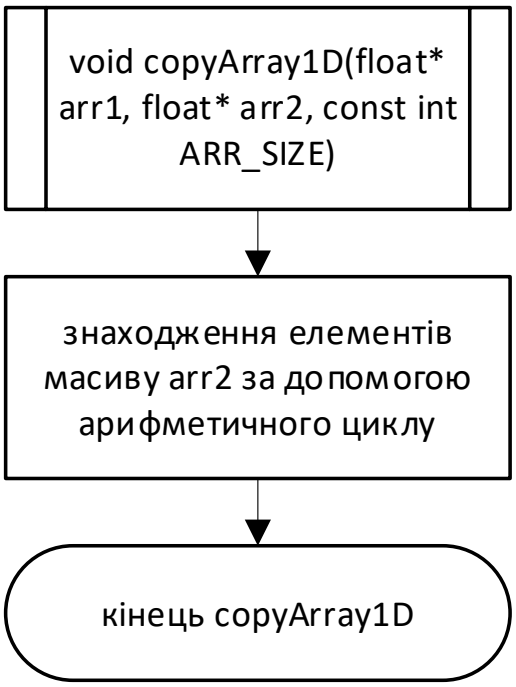


Крок 2

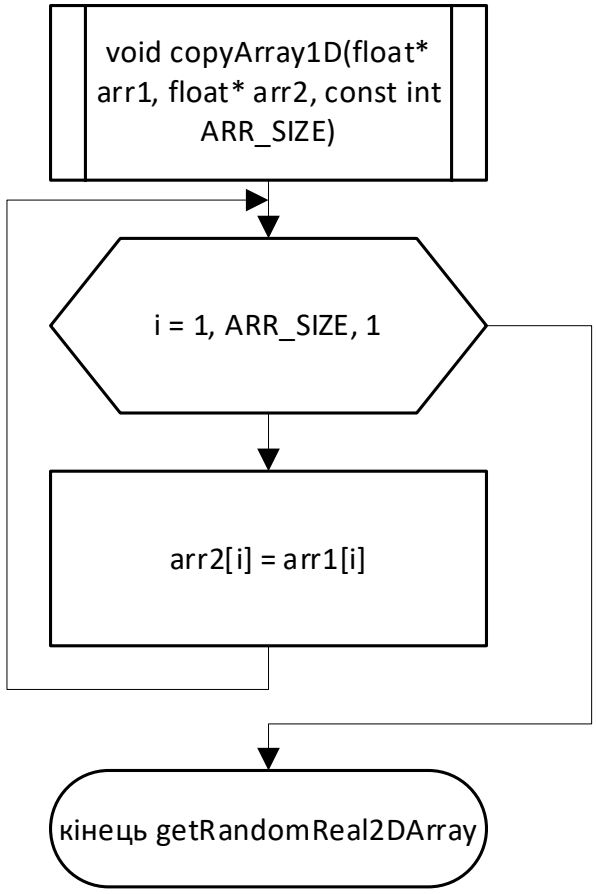


Підпрограма void copyArray1D(float* arr1, float* arr2, const int ARR_SIZE):

Крок 1

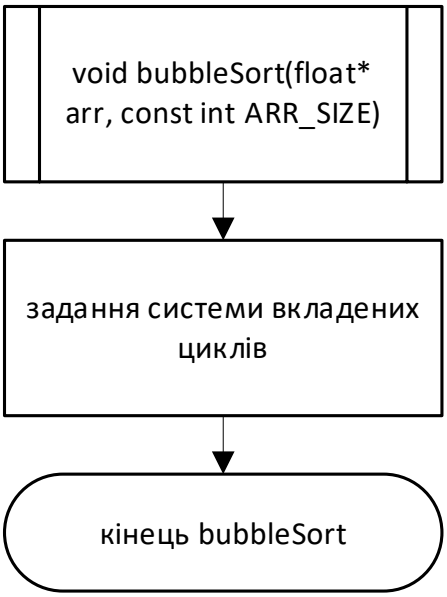


Крок 2

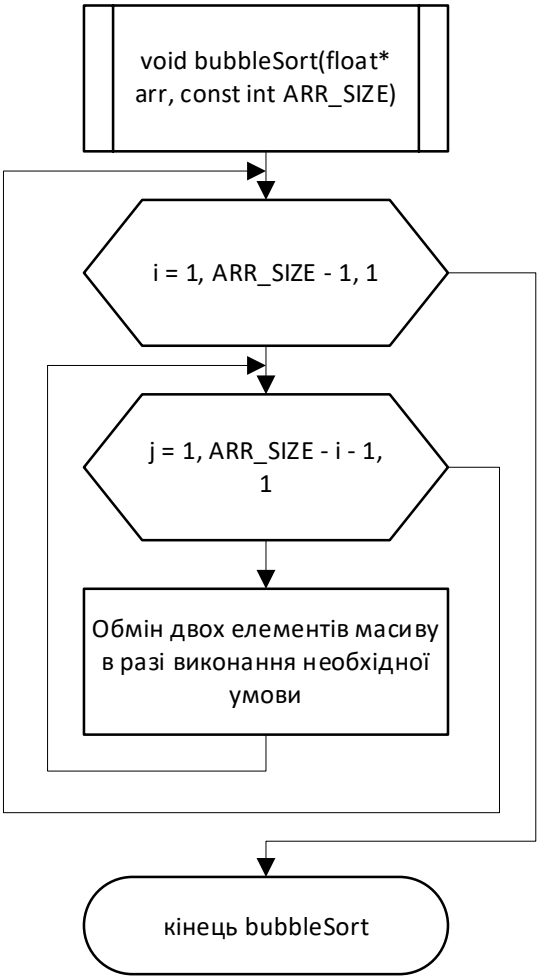


Підпрограма void bubbleSort(float* arr, const int ARR_SIZE):

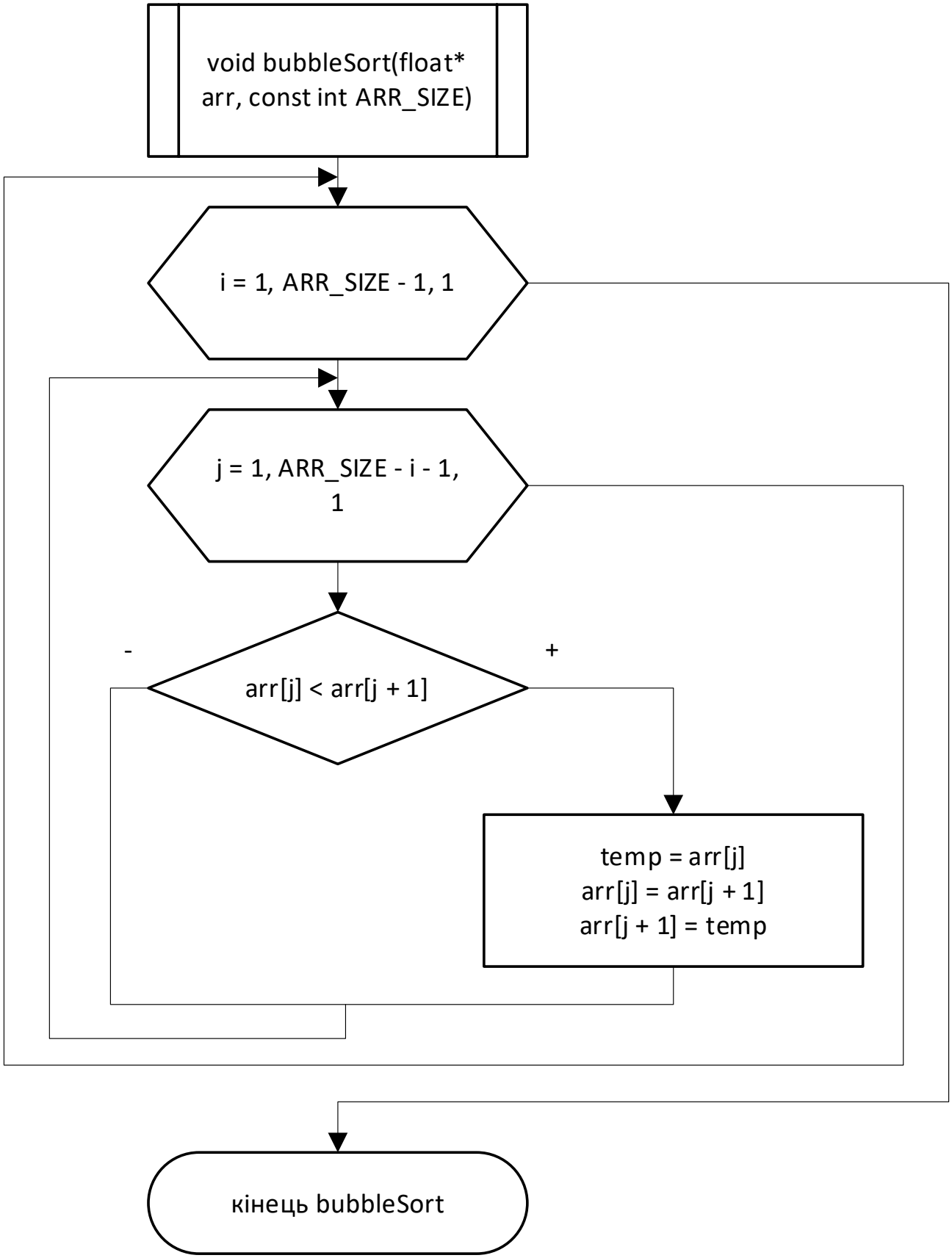
Крок 1



Крок 2



Крок 3



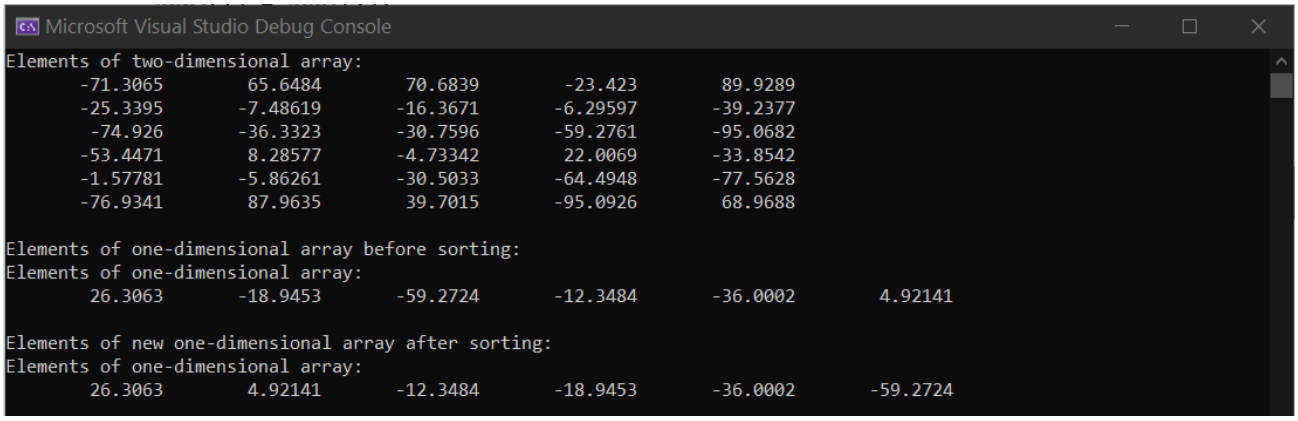
Реалізація алгоритму на мові C++:

```cpp
#include <iostream>
#include <ctime>
#include <cstdlib>
#include <iomanip>

float getRandomRealNum();

float** getRandomReal2DArray(const size_t ROWS_NUM, const size_t COLUMNS_NUM);

void array2DOutput(float** arr2D, const size_t ROWS_NUM, const size_t COLUMNS_NUM);

void deleteArray2D(float** arr2D, const size_t ROWS_NUM);

void bubbleSort(float arr[], const size_t ARR_SIZE);

float* getConditional1DArray(float** arr2D, const size_t ROWS_NUM, const size_t
COLUMNS_NUM);

void array1DOutput(float* arr1D, const size_t ARR_SIZE);

void deleteArray1D(float* arr1D);

void copyArray1D(float* arr1, float* arr2, const size_t ARR_SIZE);

int main()
{
       const size_t ROWS_NUM{ 6 }, COLUMNS_NUM{ 5 };
    float** arr2D;
    arr2D = getRandomReal2DArray(ROWS_NUM, COLUMNS_NUM);
    array2DOutput(arr2D, ROWS_NUM, COLUMNS_NUM);

    float* arr1D;
    arr1D = getConditional1DArray(arr2D, ROWS_NUM, COLUMNS_NUM);
    std::cout << "\nElements of one-dimensional array before sorting:\n";
    array1DOutput(arr1D, ROWS_NUM);

    float* newArr1D = new float[ROWS_NUM];
    copyArray1D(arr1D, newArr1D, ROWS_NUM);
    bubbleSort(newArr1D, ROWS_NUM);
    std::cout << "\nElements of new one-dimensional array after sorting:\n";
    array1DOutput(newArr1D, ROWS_NUM);

    deleteArray2D(arr2D, ROWS_NUM);
    deleteArray1D(arr1D);
    deleteArray1D(newArr1D);
    return 0;
}

float getRandomRealNum()
{
       float randomRealNum;
       const float MY_RAND_MAX = 100.0, MY_RAND_MIN = -100.0;
       randomRealNum = MY_RAND_MIN + static_cast <float> (rand()) / (static_cast <float>
(RAND_MAX / (MY_RAND_MAX - MY_RAND_MIN)));
       return randomRealNum;
}

float** getRandomReal2DArray(const size_t ROWS_NUM, const size_t COLUMNS_NUM)
{
    srand(static_cast <unsigned int> (time(NULL)));
```

```
    float** arr2D = new float* [ROWS_NUM];

    for (size_t i{ 0 }; i < ROWS_NUM; ++i) {
        arr2D[i] = new float[COLUMNS_NUM];
    }

    for (size_t i{ 0 }; i < ROWS_NUM; ++i) {
        for (size_t j{ 0 }; j < COLUMNS_NUM; ++j) {
            arr2D[i][j] = getRandomRealNum();
        }
    }
    return arr2D;
}

void array2DOutput(float** arr2D, const size_t ROWS_NUM, const size_t COLUMNS_NUM)
{
    std::cout << "Elements of two-dimensional array:\n";
    for (size_t i{ 0 }; i < ROWS_NUM; ++i) {
        for (size_t j{ 0 }; j < COLUMNS_NUM; ++j) {
            std::cout << std::setw(15) << arr2D[i][j];
        }
        std::cout << "\n";
    }
}

void deleteArray2D(float** arr2D, const size_t ROWS_NUM)
{
    for (size_t i{ 0 }; i < ROWS_NUM; ++i) {
        delete[] arr2D[i];
    }
    delete[] arr2D;
}

void bubbleSort(float arr[], const size_t ARR_SIZE)
{
    for (size_t i{ 0 }; i < ARR_SIZE - 1; ++i) {
        for (size_t j{ 0 }; j < ARR_SIZE - i - 1; ++j) {
            if (arr[j] < arr[j + 1]) {
                float temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

float* getConditional1DArray(float** arr2D, const size_t ROWS_NUM, const size_t COLUMNS_NUM)
{
    float* arr1D = new float[ROWS_NUM];
    for (size_t i{ 0 }; i < ROWS_NUM; ++i) {
        float rowAverage{};
        for (size_t j{ 0 }; j < COLUMNS_NUM; ++j) {
            rowAverage += arr2D[i][j];
        }
        rowAverage /= static_cast <float> (COLUMNS_NUM);
        arr1D[i] = rowAverage;
    }
    return arr1D;
}

void array1DOutput(float* arr1D, const size_t ARR_SIZE)
```

```
{
    std::cout << "Elements of one-dimensional array:\n";
    for (size_t i{ 0 }; i < ARR_SIZE; ++i) {
        std::cout << std::setw(15) << arr1D[i];
    }
    std::cout << "\n";
}

void deleteArray1D(float* arr1D)
{
    delete[] arr1D;
}

void copyArray1D(float* arr1, float* arr2, const size_t ARR_SIZE)
{
    for (size_t i{ 0 }; i < ARR_SIZE; ++i) {
        arr2[i] = arr1[i];
    }
}
```

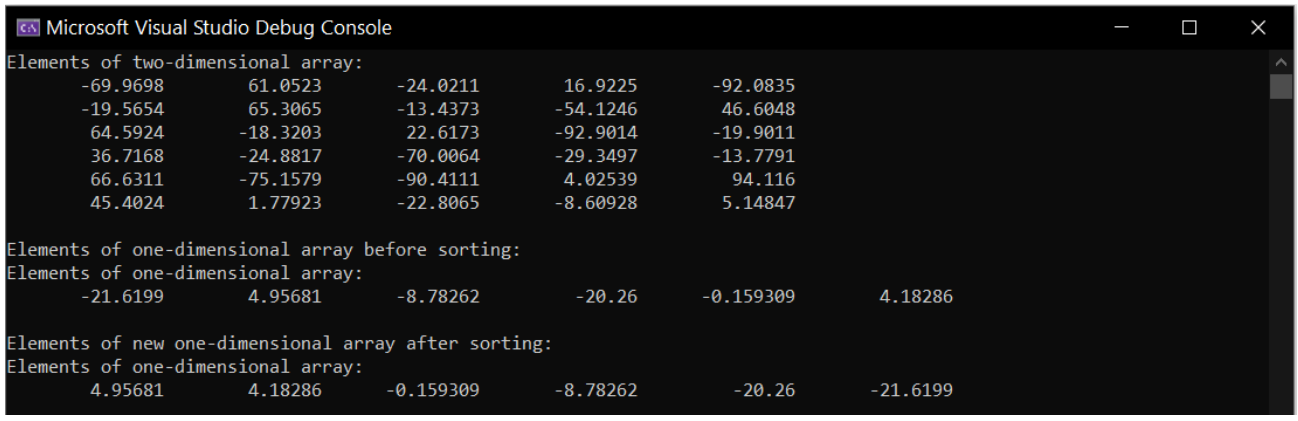
Тестування програми:



arr1D[0] = (-71.3065 + 65.6484 + 70.6839 - 23.423 + 89.9289) / 5.0 = 26.3063

…

arr1D[5] = (-76.9341 + 87.9635 + 39.7015 - 95.0926 + 68.9688) / 5.0 = 4.92141



arr1D[0] = (-69.9698 + 61.0523 - 24.0211 + 16.9225 - 92.0835) / 5.0 = -21.6199

…

arr1D[5] = (45.4024 + 1.77923 - 22.8065 - 8.60928 + 5.14847) / 5.0 = 4.18286

Висновки. Таким чином, в результаті виконання лабораторної роботи було досліджено алгоритми пошуку (на прикладі пошуку середнього арифметичного значення елементів рядків двовимірного масиву дійсних чисел) та сортування (на прикладі реалізації алгоритму сортування обміном за спаданням) та набуто практичних навичок використання цих алгоритмів під час складання програмних специфікацій. Особливістю виконання цього варіанту лабораторної є написання кількох власних підпрограм та задання елементів двовимірного масиву за допомогою функції rand().