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 26

Виконав студент ІП-12, Саркісян Валерія Георгіївна

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

Київ 2021

Лабораторна робота 8
Дослідження алгоритмів пошуку та сортування

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

Розробити алгоритм та написати програму, яка складається з наступних дій:
1. Опису змінної індексованого типу (двовимірний масив) згідно з варіантом (табл. 1).
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.

26	6 x 5	Дійсний	Із середнього арифметичного від'ємних значень елементів рядків двовимірного масиву. Відсортувати методом бульбашки за спаданням.

1. ***Постановка задачі.***
   Результатом розв'язку є одновимірний масив, відсортований методом бульбашки за спаданням. За умовою спочатку потрібно згенерувати матрицю розмірністю m×n, де m=6, n=5 та заповнити її випадковими дійсними числами з діапазону від -100 до 100. Далі необхідно створити одновимірний масив, елементами якого будуть числа, що є середніми арифметичними від'ємних значень елементів рядків вхідного двовимірного масиву, і відсортувати утворений масив методом бульбашки за спаданням.

2. ***Побудова математичної моделі***.

Змінна	Тип	Ім'я	Призначення
Кількість рядків масиву	цілий, сталий	m	Початкове дане, константа
Кількість стовпців масиву	цілий, сталий	n	Початкове дане, константа
Початковий двовимірний масив	дійсний	arr[m][n]	Вхідне дане
Вихідний одновимірний масив	дійсний	newArr[m]	результат
Лічильник	цілий	i	Проміжне дане
Лічильник	цілий	j	Проміжне дане
Генерація двовимірного масива	Функція, порожній	generateArr()	Проміжне дане
Виведення двовимірного масива	Функція, порожній	printArr()	Проміжне дане
Генерація одновимірного масива	Функція, порожній	createNewArr()	Проміжне дане

Виведення одновимірного масива	Функція, порожній	printNewArr()	Проміжне дане
Сортування одновимірного масива	Функція, порожній	sortNewArr()	результат
двовимірний масив	дійсний	A	Формальний параметр, проміжне дане
Кількість рядків	цілий	rows	Формальний параметр, проміжне дане
Кількість стовпців	цілий	columns	Формальний параметр, проміжне дане
одновимірний масив	дійсний	B	Формальний параметр, проміжне дане
Випадкові числа	Функція	rand()	Генерація випадкових чисел
Сума від'ємних елементів рядка двовимірного масива	дійсний	sum	Проміжне дане
Кількість від'ємних елементів рядка двовимірного масива	дійсний	kilk	Проміжне дане
Середнє арифметичне від'ємних елементів рядка двовимірного масива	дійсний	average	Проміжне дане
Кількість рядків	цілий	lengths	Формальний параметр, проміжне дане
Допоміжна змінна	дійсний	temp	Проміжне дане

3. ***Розв'язання.***

Основна програма:

Крок 1. Визначимо основні дії.

Крок 2. Деталізуємо дію ініціалізації змінних.

Крок 3. Деталізуємо дію заповнення двовимірного масиву випадковими дійсними числами.

Крок 4. Деталізуємо дію виведення двовимірного масиву.

Крок 5. Деталізуємо дію знаходження елементів та створення одновимірного масиву.

Крок 6. Деталізуємо дію виведення одновимірного масиву.

Крок 7. Деталізуємо дію сортування одновимірного масиву методом бульбашки за спаданням.

Підпрограми:

Крок 3.1. Деталізуємо дію заповнення двовимірного масиву випадковими дійсними числами.

Крок 4.1. Деталізуємо дію виведення двовимірного масиву.

Крок 5.1. Деталізуємо дію знаходження елементів та створення одновимірного масиву.
Крок 6.1. Деталізуємо дію виведення одновимірного масиву.
Крок 7.1. Деталізуємо дію сортування одновимірного масиву методом бульбашки за спаданням.

Псевдокод.

Крок 1
Початок
Ініціалізація змінних
заповнення двовимірного масиву
виведення двовимірного масиву
створення одновимірного масиву
виведення одновимірного масиву
сортування одновимірного масиву
Кінець

Крок 2
Початок
m = 6, n = 5
заповнення двовимірного масиву
виведення двовимірного масиву
створення одновимірного масиву
виведення одновимірного масиву
сортування одновимірного масиву
Кінець

Крок 3
Початок
m = 6, n = 5
generateArr (arr, m, n)
виведення двовимірного масиву
створення одновимірного масиву
виведення одновимірного масиву
сортування одновимірного масиву
Кінець

Крок 4
Початок
m = 6, n = 5

generateArr (arr, m, n)
printArr (arr, m, n)
створення одновимірного масиву
виведення одновимірного масиву
сортування одновимірного масиву
Кінець

Крок 5
Початок
m = 6, n = 5
generateArr (arr, m, n)
printArr (arr, m, n)
createNewArr (newArr, arr, m, n)
виведення одновимірного масиву
сортування одновимірного масиву
Кінець

Крок 6
Початок
m = 6, n = 5
generateArr (arr, m, n)
printArr (arr, m, n)
createNewArr (newArr, arr, m, n)
printNewArr (newArr, m)
сортування одновимірного масиву
Кінець

Крок 7
Початок
m = 6, n = 5
generateArr (arr, m, n)
printArr (arr, m, n)
createNewArr (newArr, arr, m, n)
printNewArr (newArr, m)
sortNewArr (newArr, m)
Кінець

Псевдокод підпрограм.

Крок 3.1. Функція **generateArr(A, rows, columns)**
Початок generateArr(A, rows, columns)
Для i від 0 до rows із кроком 1 **повторити**

Для j від 0 до columns із кроком 1 **повторити**
A[i][j] = rand() / RAND_MAX * 2 * 100 - 100
Все повторити
Все повторити
Кінець generateArr

Крок 4.1. Функція **printArr(A, rows, columns)**
Початок printArr(A, rows, columns)
Для i від 0 до rows із кроком 1 **повторити**
Для j від 0 до columns із кроком 1 **повторити**
вивести A[i][j]
Все повторити
Все повторити
Кінець printArr

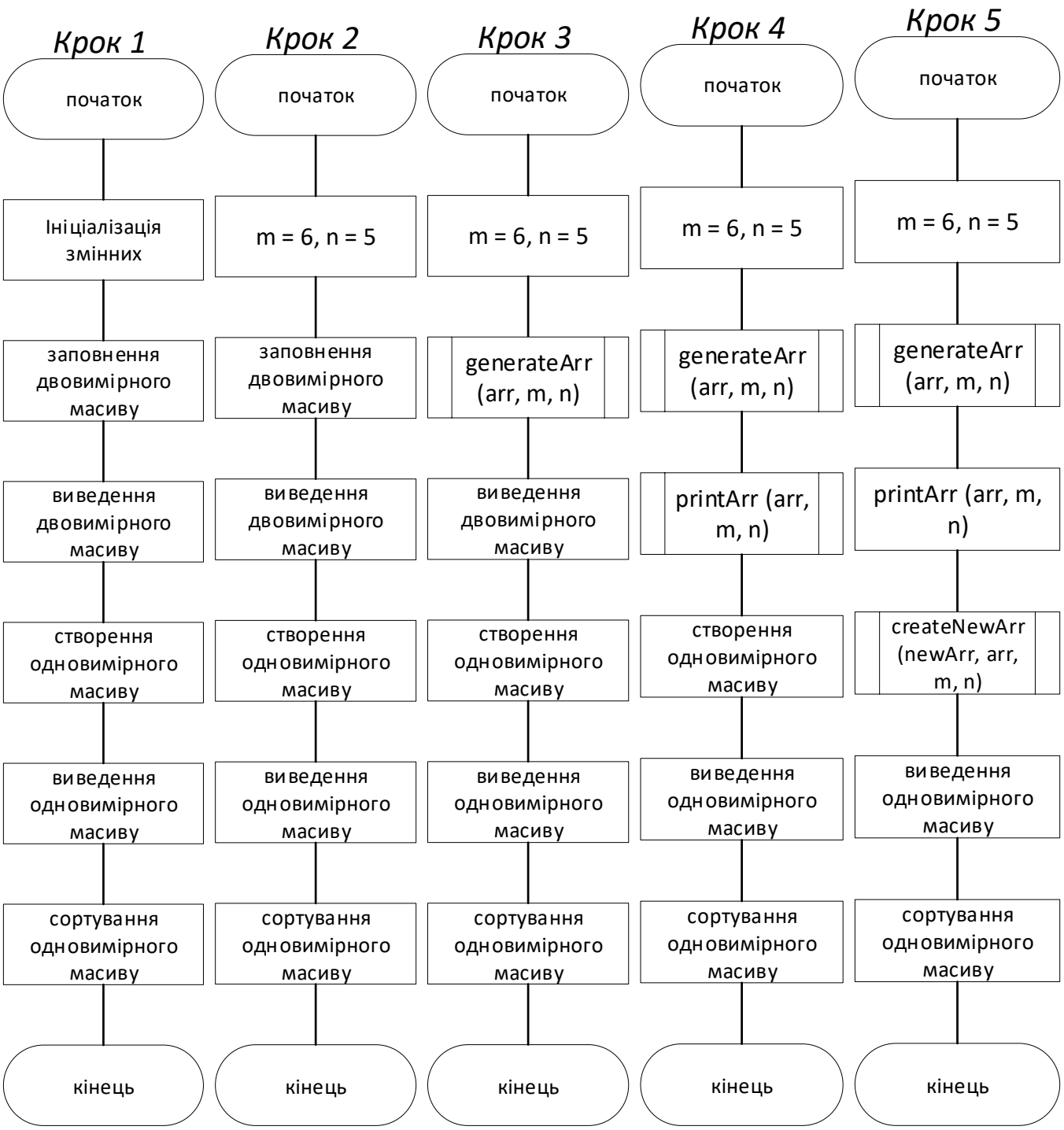
Крок 5.1. Функція **createNewArr (B, A, rows, columns)**
Початок createNewArr (B, A, rows, columns)
Average=0
Для i від 0 до rows із кроком 1 **повторити**
sum=0; kilk=0
Для j від 0 до columns із кроком 1 **повторити**
Якщо A[i][j] < 0 **то**
sum = sum + A[i][j]
kilk = kilk + 1
Все Якщо
Все повторити
Average = sum / kilk
B[i] = average
Все повторити
Кінець createNewArr

Крок 6.1. Функція **printNewArr (B, lenghts)**
Початок printNewArr (B, lenghts)
Для i від 0 до lengths із кроком 1 **повторити**
вивести B[i]
Все повторити
Кінець printNewArr

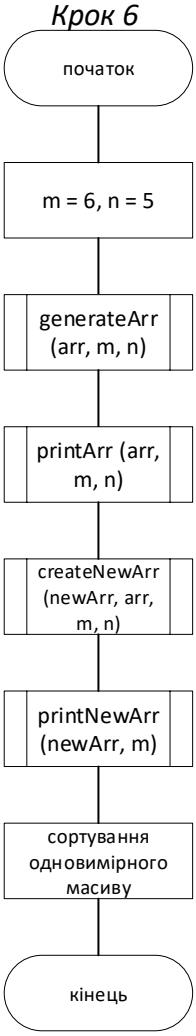
Крок 7.1. Функція **sortNewArr (B, lengths)**
Початок sortNewArr (B, lengths)
Для i від 0 до (lengths – 1) із кроком (попереднім) 1 **повторити**
Для j від 0 до (lengths – 1) із кроком (попереднім) 1 **повторити**

Якщо B[j] < B[j + 1] **то**
temp = B[j + 1]
B[j + 1] = B[j]
B[j] = temp
Все Якщо
Все повторити
Все повторити
printNewArr (B, lenghts)
Кінець sortNewArr

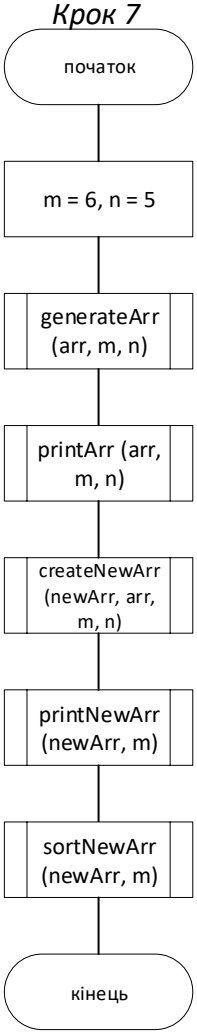
4. ***Блок-схема.***


Крок 1
початок
Ініціалізація змінних
заповнення двовимірного масиву
виведення двовимірного масиву
створення одновимірного масиву
виведення одновимірного масиву
сортування одновимірного масиву
кінець
Крок 2
початок
m = 6, n = 5
заповнення двовимірного масиву
виведення двовимірного масиву
створення одновимірного масиву
виведення одновимірного масиву
сортування одновимірного масиву
кінець
Крок 3
початок
m = 6, n = 5
generateArr (arr, m, n)
виведення двовимірного масиву
створення одновимірного масиву
виведення одновимірного масиву
сортування одновимірного масиву
кінець
Крок 4
початок
m = 6, n = 5
generateArr (arr, m, n)
printArr (arr, m, n)
створення одновимірного масиву
виведення одновимірного масиву
сортування одновимірного масиву
кінець
Крок 5
початок
m = 6, n = 5
generateArr (arr, m, n)
printArr (arr, m, n)
createNewArr (newArr, arr, m, n)
виведення одновимірного масиву
сортування одновимірного масиву
кінець

Крок 6

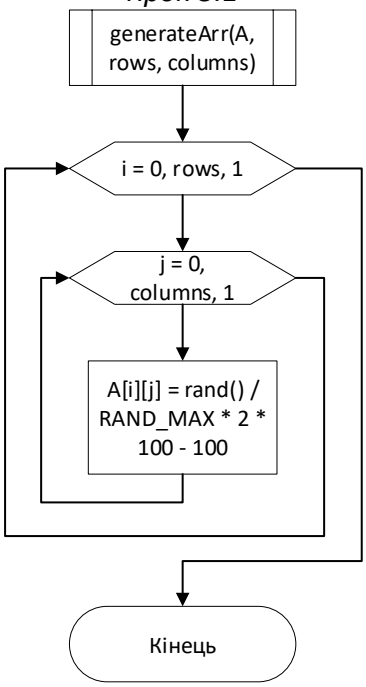


Крок 7

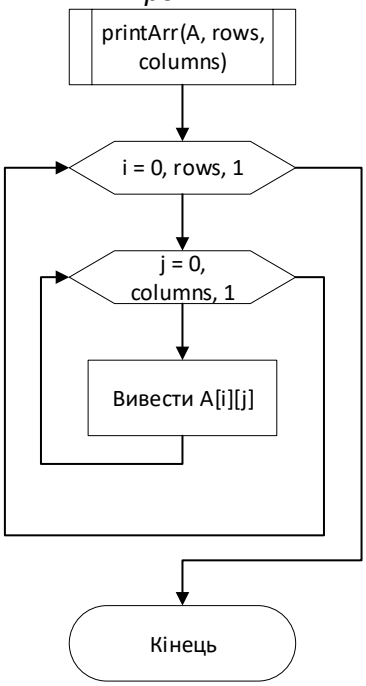


Підпрограми:

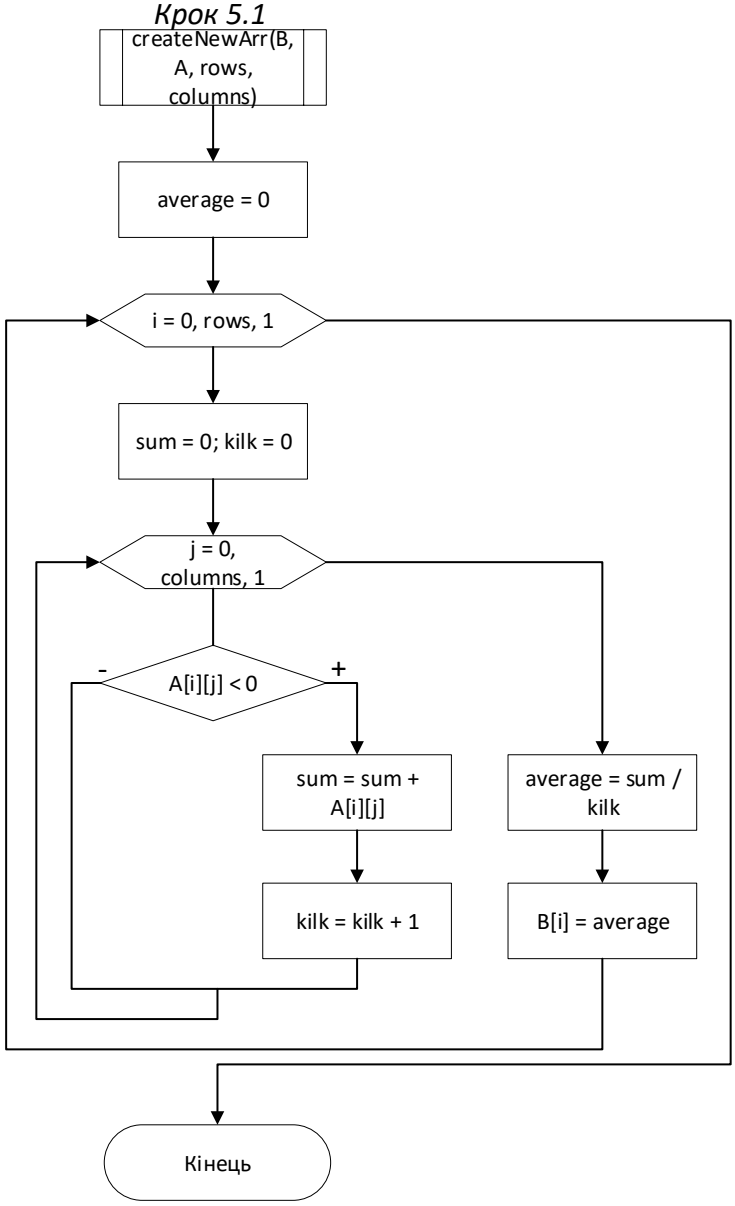
Крок 3.1



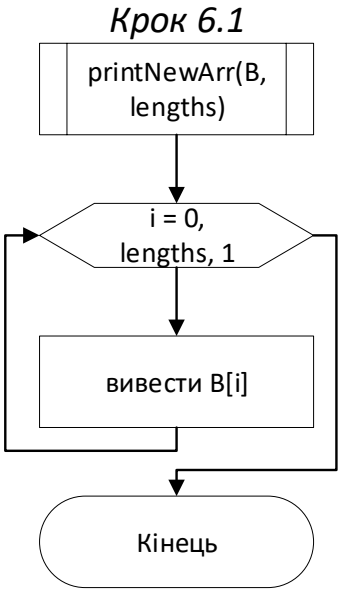
Крок 4.1



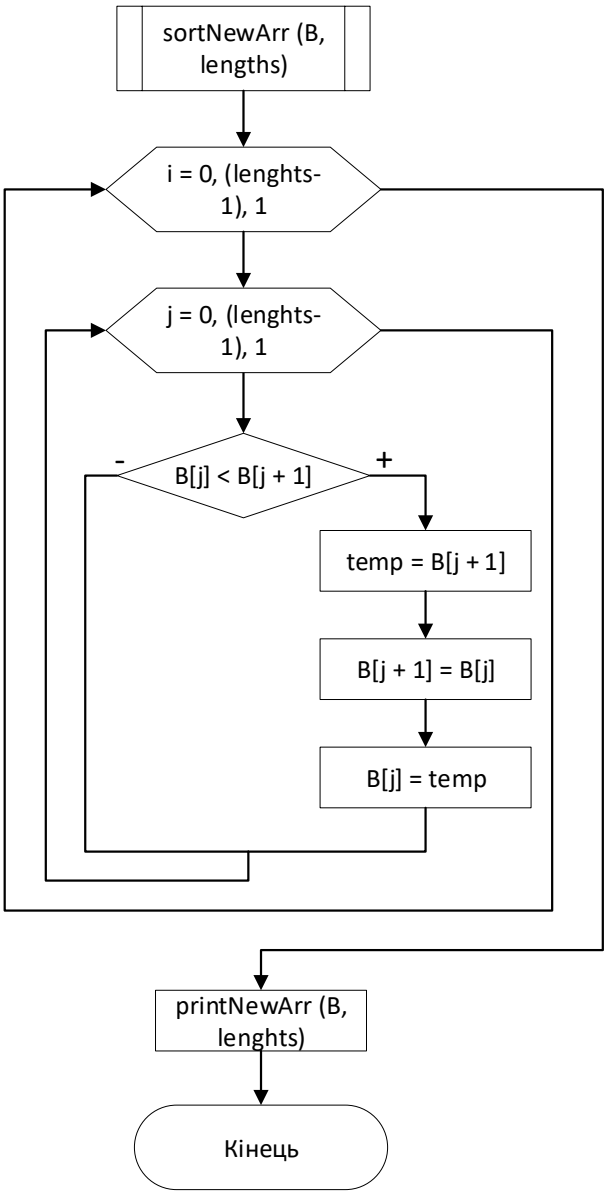
Крок 5.1



Крок 6.1



Крок 7.1



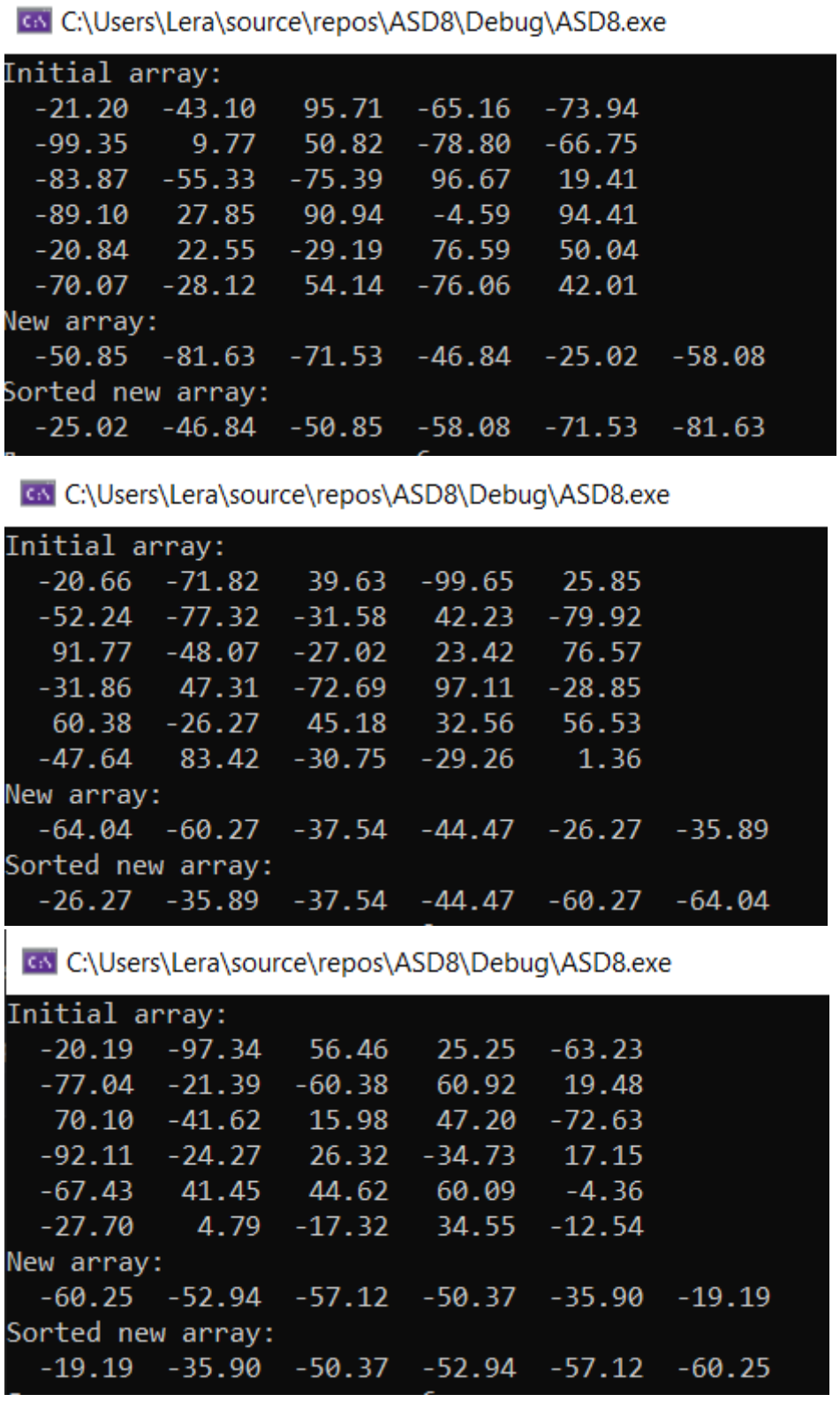
5. *Код програми (C++):*

```
#include <iostream>
#include <time.h>
using namespace std;

void generateArr(float**, int, int);
void printArr(float**, int, int);
void createNewArr(float*, float**, int, int);
void printNewArr(float*, int);
void sortNewArr(float*, int);

int main() {
   const int m = 6, n = 5;
   float** arr = new float* [m];
   for (int i = 0; i < m; i++) {
         arr[i] = new float[n];
   }
   generateArr(arr, m, n);
   cout << "Initial array:\n";
   printArr(arr, m, n);
   float* newArr = new float[m];
```

```
    createNewArr(newArr, arr, m, n);
    cout << "New array:\n";
    printNewArr(newArr, m);
    cout << "\nSorted new array:\n";
    sortNewArr(newArr, m); cout << endl;
    for (int i = 0; i < m; i++) {
          delete[] arr[i];
    }
    delete[] arr;
    delete[]newArr;
    system("pause");
}

void generateArr(float** A, int rows, int columns) {
    srand(time(NULL));
    for (int i = 0; i < rows; i++) {
          for (int j = 0; j < columns; j++) {
                A[i][j] = (float)rand() / RAND_MAX * 2 * 100 - 100;
          }
    }
}

void printArr(float** A, int rows, int columns) {
    for (int i = 0; i < rows; i++) {
          for (int j = 0; j < columns; j++) {
                printf("%8.2f", A[i][j]);
          }
          printf("%s\n", "");
    }
}

void createNewArr(float* B, float** A, int rows, int columns) {
    float average = 0;
    for (int i = 0; i < rows; i++) {
          float sum = 0;
          int kilk = 0;
          for (int j = 0; j < columns; j++) {
                if (A[i][j] < 0) {
                      sum = sum + A[i][j];
                      kilk++;
                }
          }
          float average = sum / kilk;
          B[i] = average;
    }
}

void printNewArr(float* B, int lengths) {
    for (int i = 0; i < lengths; i++) {
          printf("%8.2f", B[i]);
    }
}

void sortNewArr(float* B, int lengths) {
    for (int i = 0; i < lengths - 1; ++i) {
          for (int j = 0; j < lengths - 1; ++j) {
                if (B[j] < B[j + 1]) {
                      float temp = B[j + 1];
                      B[j + 1] = B[j];
                      B[j] = temp;
```

```
                }
            }
        }
        printNewArr(B, lengths);
    }
```

Тестування:



1. ***Висновки.***

   На цій лабораторній роботі було досліджено алгоритми пошуку та сортування, набуто практичних навичок використання цих алгоритмів під час складання

програмних специфікацій. Для розв’язання поставленої задачі було використано метод сортування бульбашкою, який полягає у порівнянні двох сусідніх елементів масива і зміни їх положення в разі, якщо вони не впорядковані.