МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Факультет систем управління літальних апаратів
Кафедра систем управління літальних апаратів

Лабораторна робота № 8

з дисципліни «Алгоритмізація та програмування»
на тему «Реалізація алгоритмів сортування та робота з файлами на мові С ++»

ХАІ.301. 175. 318. 08 ЛР

Виконав студент гр. 318

Каріна ГЛЄБОВА
(підпис, дата) (П.І.Б.)

Перевірив
к.т.н., доц. Олена ГАВРИЛЕНКО
(підпис, дата) (П.І.Б.)

2025

МЕТА РОБОТИ

Вивчити теоретичний матеріал по алгоритмам обробки масивів на мові C++, а також бібліотеки для роботи з файлами і реалізувати оголошення, введення з файлу, обробку і виведення в файл одновимірних і двовимірних масивів на мові C ++ в середовищі Visual Studio.

ПОСТАНОВКА ЗАДАЧІ

Завдання 1. За допомогою текстового редактору створити текстовий файл «array_in_n.txt» з елементами вихідного масиву (n - номер варіанта). У програмі на C++ зчитати і перетворити цей масив відповідно до свого варіанту завдання, ім'я файлу і необхідні змінні ввести з консолі. Вивести результати у файл «array_out_n.txt».

Array79. Дан масив розміру N. Здійснити зрушення елементів масиву вправо на одну позицію (при цьому A1 перейде в A2, A2 - в A3, ..., AN-1 - в AN, а початкове значення останнього елемента буде втрачено). Перший елемент отриманого масиву покласти рівним 0.

Завдання 2. За допомогою текстового редактору створити текстовий файл «matr_in_n.txt» з елементами вихідного двовимірного масиву (n – номер варіанта). У програмі зчитати і обробити матрицю відповідно до свого варіанту завдання, ім'я файлу і необхідні змінні ввести з консолі. Дописати результати в той же файл.

Matrix44. Дана матриця розміру M × N. Знайти мінімальний серед елементів тих рядків, які впорядковані або по зростанню, або по спадаючій. Якщо впорядковані рядки в матриці відсутні, то вивести 0.

Завдання 3. Вивчити метод сортування відповідно до свого варіанту, проаналізувати його складність і продемонструвати на прикладі з 7-ми елементів. Реалізувати у вигляді окремої функції алгоритм сортування елементів масиву. Зчитування і виведення відсортованого масиву організувати на файлах.

Sort3. Вивчити метод сортування вибір, проаналізувати його складність і продемонструвати на прикладі з 7-ми елементів цілого типу у порядку зростання.

Завдання 4. Введення, виведення, обробку масивів реалізувати окремими функціями з параметрами.

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Array 79.

Вхідні дані (ім’я, опис, тип, обмеження):

const N = 20 – максимальний розмір масиву, ціле, константа.

array_in_8.txt – вхідний файл для запису масиву;

Вихідні дані (ім’я, опис, тип):

array_out_8.txt – вихідний файл для виведення результату;

Лістинг коду вирішення задачі Array 79 наведено в дод. А (стор. 6-11).

Екран роботи програми показаний на рис. Б.1. (дод. Б, стор. 12-14)

Алгоритм зчитування відбувається так: відкривається початковий файл array_in_8.txt, куди записується початковий масив, та перевіряється наявність помилки при відкритті. Якщо помилки немає, зчитується розмір масиву і його елементи, після чого файл закривається та програма вирішує завдання. У випадку виникнення помилки виводиться відповідне сповіщення та програма завершується.

Алгоритм запису в файл відбувається так: відкривається файл для запису результату array_out_8.txt, елементи із зсувом вправо виводяться на екран, після чого файл закривається.

Приклад діаграми для завдання Array 79 наведено на рис. Б.4. (дод. Б, стор. 12-14)

Завдання 2.

Вирішення задачі Matrix 44.

Вхідні дані (ім’я, опис, тип, обмеження):

const M = 20 – максимальний розмір матриці, ціле, константа.

row – рядки матриці, цілі числа;

col – стовпці матриці, цілі числа;

matr_in_8.txt – вхідний файл для запису матриці;

Вихідні дані (ім’я, опис, тип):

matr_out_8.txt – вихідний файл для виведення результату.

Лістинг коду вирішення задачі Matrix 44 наведено в дод. А (стор. 6-11).

Екран роботи програми показаний на рис. Б.2. (дод. Б, стор. 12-14)

Алгоритм зчитування відбувається так: відкриття файлу для запису початкової матриці matrix_in_8.txt. При виникненні помилки виводиться відповідне сповіщення та вирішення програми припиняється. У випадку, якщо помилки не виявлено, зчитуються кількість рядків та стовпців і кожен елемент окремо. Після цього файл закривається.

Алгоритм запису в файл відбувається так: виведення та відкриття файлу для запису результатів matrix_out_8.txt, запис у нього початкової матриці та знайдений мінімальний елемент серед відсортованих по зростанню та спаданню рядків, закриття файлу після зчитування.

Приклад діаграми для завдання Matrix 44 наведено на рис. Б.5. (дод. Б, стор. 12-14)

Завдання 3.

Сортування масиву.

Вхідні дані (ім’я, опис, тип, обмеження):

const S = 7 – максимальний розмір масиву, ціле, константа;

sort_out_8.txt – вхідний файл для запису масиву;

Вихідні дані (ім’я, опис, тип):

sort_out_8.txt – вихідний файл для виведення результату.

Лістинг коду вирішення завдання 3 наведено в дод. А (стор. 6-11)

Екран роботи програми показаний на рис. Б.3. (дод. Б, стор. 12-14)

Алгоритм виконання сортування масиву відбувається так: проводиться пошук мінімального елемента серед всіх, проходячись покроково по кожному окремо. При знаходженні такого елемента йому присвоюється індекс і та він переміщується у початок масиву, після чого продовжується пошук серед невідсортованої частини. У випадку знаходження меншого за і елемента, відбувається оновлення значення мінімального елементу на j та проходить обмін місцями між цими двома елементами. Дія повторюється з усіма подальшими елементами.

Приклади діаграм для завдання 3 наведено на рис. Б.6 та Б.7. (дод. Б, стор. 12-14)

ВИСНОВКИ

Було вивчено теоретичний матеріал по алгоритмам обробки масивів на мові С++ і бібліотеки для роботи з файлами. На практиці було реалізовано оголошення, введення з файлу, обробку і виведення в файл одновимірних і двовимірних масивів на мові С ++ в середовищі Visual Studio.

ДОДАТОК А
Лістинг коду програми

```
#include <iostream>
#include "windows.h"
#include "array_utils.h"
#include "matrix_utils.h"
using namespace std;

//За допомогою текстового редактору створити текстовий файл «array_in_n.txt» з елементами
// вихідного масиву(n – номер варіанта).У програмі на С++ зчитати і перетворити цей масив
// відповідно до свого варіанту завдання(див.лаб.роб.No6, завд.2), ім'я файлу і необхідні змінні
// ввести з консолі.Вивести результати у файл «array_out_n.txt».*/
void task_array79(); // завдання 1

//За допомогою текстового редактору створити текстовий файл «matr_in_n.txt» з елементами
// вихідного двовимірного масиву(n – номер варіанта).У програмі зчитати і обробити матрицю відповідно
// до свого варіанту завдання(лаб.роб.No7, завд.1), ім'я файлу і необхідні змінні ввести з консолі.
// Дописати результати в той же файл.
void task_matrix71(); // завдання 2

// Вивчити метод сортування відповідно до свого варіанту (див табл. 1), проаналізувати його складність
// і продемонструвати на прикладі з 7 – ми елементів(відповідно до свого варіанту).Реалізувати у вигляді
// окремої функції алгоритм сортування елементів масиву.Зчитування і виведення відсортованого
// масиву організувати на файлах.
void task_3(); // завдання 3

int main()
{
    SetConsoleOutputCP(1251);
    int menu; // Зміна для номеру завдання
    do
    { // початок циклу
        cout << "Номер завдання:"; //введення номеру завдання
        cin >> menu; // обирання номеру завдання
        cout << endl; //вільна строка
        switch (menu) {
        case 1: task_array79(); break; // 1 – завдання 1
        case 2: task_matrix71(); break; // 2 – завдання 2
        case 3: task_3(); break; // 3 – завдання 3
        case -1: cout << "Вихід..." << endl; break; // -1 – вихід
        default: cout << "Помилка! Лише 1, 2!" << endl; // інший номер – повторити
        }
        cout << endl; // вільна строка
        cout << "+---------------------+" << endl; // строка задля полегшення візуального сприймання тексту
        cout << endl; // вільна строка
    } // кінець циклу
    while (menu != -1); // умова виконання циклу
    return 0;
}
```

```
// Основна функція для виконання завдання
void task_array79() {
    int mas[N];  // масив для зберігання елементів
    int size;    // розмір масиву
    char filename[100];  // назва файлу для введення/виведення
    cout << "Введіть назву файлу (array_in_8.txt): " << endl; // введення назви файлу,
який потрібно відкрити
    cin.ignore(); // звільнення буфера вводу
    cin.getline(filename, 100); // вводимо назву файлу з консолі
    cout << filename << endl; // виведення файлу

    if (get_mas(filename, mas, size)) { // виклик функції для зчитування масиву з
файлу
        cout << "Початковий масив:\t\t\t";// якщо масив успішно зчитано, виводиться
його на екран
        show_mas(mas, size);

        shift_array_right(mas, size); // зсув елементів масиву вправо
        cout << "Масив після зрушення:\t"; // виведення масиву після зрушення

        show_mas(mas, size);
        write_mas(mas, size); // запис обробленого масиву у файл
    }
}

// Функція для виконання завдання 2
void task_matrix71() {
    int matr[M][N];  // матриця розміру M x N
    int row, col;    // кількість рядків і стовпців
    char filename[100];  // ім'я файлу

    cout << "Введіть назву файлу (matr_in_8.txt): " << endl; // введення назви файлу
    cin.ignore(); // звільнення буфера вводу
    cin.getline(filename, 100);  // введення імені файлу

    if (get_matr(filename, matr, row, col)) { // якщо зчитування матриці з файлу
вдалося
        cout << "Початкова матриця: " << endl; // виводимо початкову матрицю
        show_matr(matr, row, col);  // фунуція виведення матриці на екран

        // Оброблення завдання
        int new_col_count = processed_task_2(matr, row, col);  // дублювання стовпця

        cout << "Матриця після дублювання рядка: " << endl; // виведення повідомлення
про дублювання
        show_matr(matr, row, new_col_count);  // виведення матриці після зміни

        write_result_task_2(filename, new_col_count); // запис результату у файл
    }
}

void task_3() { // вирішення завдання 3
    int arr [S];  // масив для сортування
    int n; // кількість елементів масиву
    char filename[100];  // ім'я файлу

    cout << "Введіть назву файлу (sort_in_8.txt): " << endl; // введення назви файлу
    cin.ignore(); // звільнення буфера вводу
    cin.getline(filename, 100);  // введення імені файлу

    if (get_sort(filename, arr, n)) { // якщо зчитування масиву з файлу вдалося
        cout << "Початковий масив: " << endl; // виведення початкового масиву
```

```
        write_sort(filename, arr, n);  // запис початкового масиву у файл

        // сортування масиву
        sort(arr, n);  // виклик функції сортування

        cout << "Відсортований масив: " << endl; // виведення відсортованого масиву
        show_sort(arr, n);  // виведення відсортованого масиву на екран

        write_sort(filename, arr, n); // запис відсортованого масиву в файл
    }
}
```

```
#ifndef ARRAY_UTILS_H
#define ARRAY_UTILS_H

const int N = 20, S = 7; // N – максимальний розмір масиву, S – кількість цифр у
масиві для сортування

bool get_mas(char* filename, int in_mas[N], int& in_n); // функція для зчитування
масиву з файлу
void show_mas(const int in_mas[N], const int n); // функція для виведення елементів
масиву
void write_mas(const int in_mas[N], const int n); // функція для запису масиву у файл
на виході
void shift_array_right(int mas[N], int& n); // функція для зсуву всіх елементів масиву
на одну позицію вправо

void sort(int arr[], int n); // функція для сортування масиву за зростанням
bool get_sort(const char* filename, int arr[], int& n); // функція для  масив із файлу
void write_sort(const char* filename, const int arr[], int n); // Записує
відсортований масив
void show_sort(const int arr[], int n); // Виводить відсортований масив на екран

#endif // ARRAY_UTILS_H
```

```
#include "array_utils.h"
#include <iostream>
#include <fstream>
using namespace std;

//Функція для зчитування елементів масиву з файлу
bool get_mas(char* filename, int in_mas[N], int& in_n) {
    ifstream fin("array_in_8.txt"); // відкриваємо файл для зчитування
    if (!fin.is_open()) { // перевірка на наявність помилки при відкритті файлу
        cout << "Помилка! Не вдалося відкрити файл!" << endl;
        return false;  // повертаємо false, якщо файл не вдалося відкрити
    }
    else {
        fin >> in_n; // розмір масиву зчитується з файлу
        for (int i = 0; i < in_n; i++) {  // елементи масиву зчитуються з файлу
            fin >> in_mas[i];
        }
    }
    fin.close();  // файл закривається після зчитування
    return true;  // повертається true, якщо зчитування пройшло успішно
}

// Функція для виведення елементів масиву до консолі
void show_mas(const int mas[N], const int n) {
    for (int i = 0; i < n; i++) {
        cout << mas[i] << " "; // виводимо елементи масиву
```

```
    }
    cout << endl;  // перехід на новий рядок після виведення всіх елементів
}

// Функція для запису елементів масиву в файл
void write_mas(const int mas[N], const int n) {
    ofstream fout("array_out_8.txt"); // відкриваємо файл для запису
    for (int i = 0; i < n; i++) { // елементи масиву записуються у файл
        fout << mas[i] << " ";
    }
    fout.close(); // після запису файл закривається
}

// Функція для обробки завдання: зсув елементів масиву вправо
void shift_array_right(int mas[N], int& n) {
    if (n > 1) { // перевірка, чи масив містить більше одного елементу
        // починаємо з останнього елементу і зсуваємо кожен елемент вправо
        for (int i = n - 1; i > 0; i--) {
            mas[i] = mas[i - 1];
        }
        mas[0] = 0; // перший елемент стає 0
    }
}

//Визначення функцій для завдання 3

// Функція для сортування масиву методом вибору
void sort(int arr[], int n) {
    for (int i = 0; i < n - 1; i++) { // пошук мінімального елемента у невідсортованій
частині
        int minIndex = i; // індекс мінімального елемента на поточному кроці
        for (int j = i + 1; j < n; j++) { // пошук мінімального елемента в залишковій
частині масиву
            if (arr[j] < arr[minIndex]) { // якщо знаходимо менший елемент,
                minIndex = j; // оновлюємо індексу мінімального елемента
            }
        }
        if (minIndex != i) {  // обмін місцями мінімального елемента з елементом на
позиції i
            swap(arr[i], arr[minIndex]); // обмін місцями
        }
    }
}

// Функція для зчитування масиву з файлу
bool get_sort(const char* filename, int arr[], int& n) {
    ifstream fin(filename); // відкриття файлу для зчитування
    if (!fin.is_open()) {  // перевірка на успішне відкриття файлу
        cout << "Не вдалося відкрити файл!" << endl; // виведення повідомлення про
помилку
        return false;
    }
    fin >> n;  // зчитування кількості елементів масиву

    if (n > S) {  // перевірка, чи кількість елементів не перевищує максимальний
розмір S
        // виведення сповіщення про помилку
        cout << "Кількість елементів у файлі перевищує максимальний розмір масиву ("
<< S << "). Буде обрано лише " << S << " елементів." << endl;
        n = S;  // обмеження кількості елементів
    }

    for (int i = 0; i < n; i++) { // зчитування елементів масиву з файлу
```

```
        fin >> arr[i]; // зчитування кожного елементу масиву
    }
    fin.close(); // закриття файлу після зчитування
    return true;
}

// Функція для запису масиву в файл
void write_sort(const char* filename, const int arr[], int n) {
    ofstream fout("sort_out_8.txt"); // відкриття файлу для запису
    if (!fout.is_open()) { // перевірка на успішне відкриття файлу
        cout << "Не вдалося відкрити файл для запису!" << endl; // якщо файл не
вдалося відкрити, виводимо помилку
        return; // якщо не вдалося відкрити файл, функція завершується
    }
    for (int i = 0; i < n; i++) { // запис елементів масиву у файл
        fout << arr[i] << " "; // запис кожного елементу масиву через пробіл
    }
    fout.close(); // закриття файлу після запису
}

// Функція для виведення масиву на екран
void show_sort(const int arr[], int n) {
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " "; // виведення кожного елементу масиву
    }
    cout << endl; // після виведення масиву додаємо новий рядок
}


#ifndef MATRIX_UTILS_H
#define MATRIX_UTILS_H

const int M = 20; // максимальний розмір матриці

bool get_matr(char* filename, int in_matr[M][M], int& in_m, int& in_n); // функція для
зчитування матриці з файлу
void write_result_task_2(char* filename, int res); // функція для запису матриці у
файл на виході
void show_matr(const int matr[M][M], const int m, const int n); // функція для
виведення елементів матриці
int processed_task_2(const int matr[M][M], const int m, const int n); // функція для
обробки завдання

#endif // MATRIX_UTILS_H


#include "matrix_utils.h"
#include <iostream>
#include <fstream>
using namespace std;

//Функція для зчитування елементів матриці з файлу
bool get_matr(char* filename, int in_matr[M][M], int& in_m, int& in_n) {
    ifstream fin("matr_in_8.txt"); // відкриття файлу для зчитування
    if (!fin.is_open()) { // якщо виникла помилка
        cout << "Помилка! Не вдалося відкрити файл!" << endl; // виводимо сповіщення
про це
        return false; // повернення значення як false
    }
    else { // в іншому випадку
        fin >> in_m; // зчитується кількість рядків
        fin >> in_n; // та стовпців
        for (int i = 0; i < in_m; i++) { // зчитуються елементи матриці
```

```
            for (int j = 0; j < in_n; j++) {
                fin >> in_matr[i][j]; // кожен елемент зчитується
            }
        }
    }
    fin.close(); // закриття програми
    return true;
}

//Функція для виведення елементів матриці до консолі
void show_matr(const int matr[M][M], const int m, const int n) {
    for (int i = 0; i < m; i++) { // зчитуються рядки матриці
        for (int j = 0; j < n; j++) { // зчитуються стовпці матриці
            cout << matr[i][j] << "  "; // виведення поточного елементу матриці з
пробілом
        }
        cout << endl; // після кожного рядка матриці перехід на новий рядок
    }
}

//Функція для запису результату завдання 2 в файл
void write_result_task_2(char* filename, int res) {
    ofstream fout("matr_out_8.txt"); // виведення файлу з результатами
    fout << "\nПродубльований стовпець: " << res; // виведення результату
    fout.close(); // файл закривається після зчитування
}

// Функція для обробки завдання 2
int processed_task_2(int matr[M][M], int m, int n) {
    // знаходження мінімального елемента у матриці та його стовпець
    int min_val = matr[0][0];  // припущення, що перший елемент - мінімальний
    int min_col = 0; // індекс стовпця, де знаходиться мінімальний елемент

    // проходимо по всіх елементах матриці, щоб знайти мінімальний елемент
    for (int i = 0; i < m; i++) { // проходимо по рядках матриці
        for (int j = 0; j < n; j++) { // проходимо по стовпцях матриці
            if (matr[i][j] < min_val) { // якщо поточний елемент менший за
мінімальний,
                min_val = matr[i][j]; // оновлення мінімального значення
                min_col = j; // оновлення індекса стовпця з мінімальним елементом
            }
        }
    }

    for (int i = 0; i < m; i++) { // дублювання стовпця, що містить мінімальний
елемент
        // зсув всіх елементів праворуч від стовпця, що містить мінімум
        for (int j = n; j > min_col; j--) {
            matr[i][j] = matr[i][j - 1];
        }
        matr[i][min_col + 1] = matr[i][min_col]; // дублювання значення мінімального
стовпця
    }

    n++; // оновлення кількості стовпців

    return n;  // повернення нової кількості стовпців
}
```

ДОДАТОК Б
Скрін-шоти вікна виконання програми

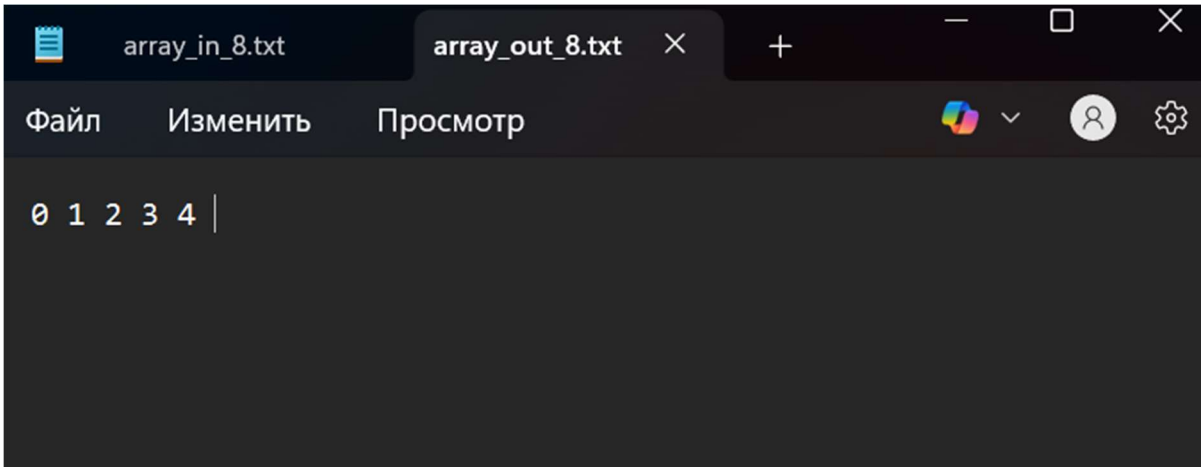

Рисунок Б.1 – Екран виконання програми для вирішення завдання Array 79

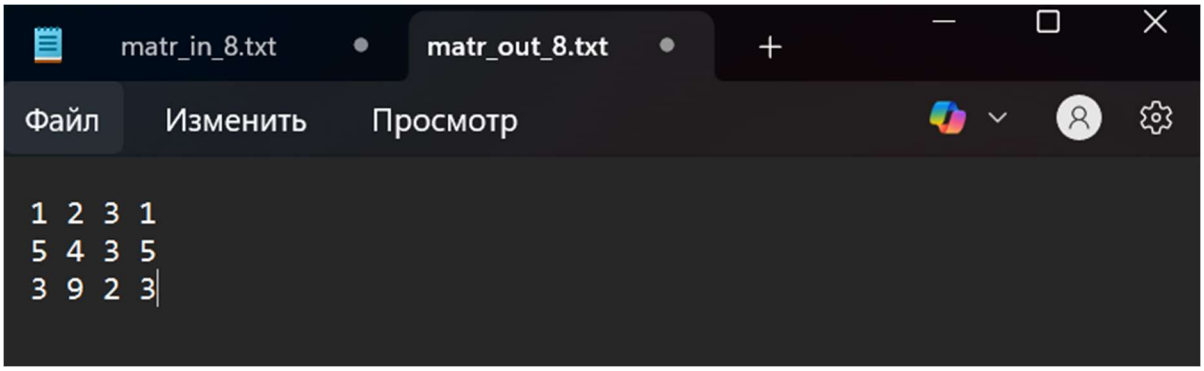

Рисунок Б.2 – Екран виконання програми для вирішення завдання Matrix 44

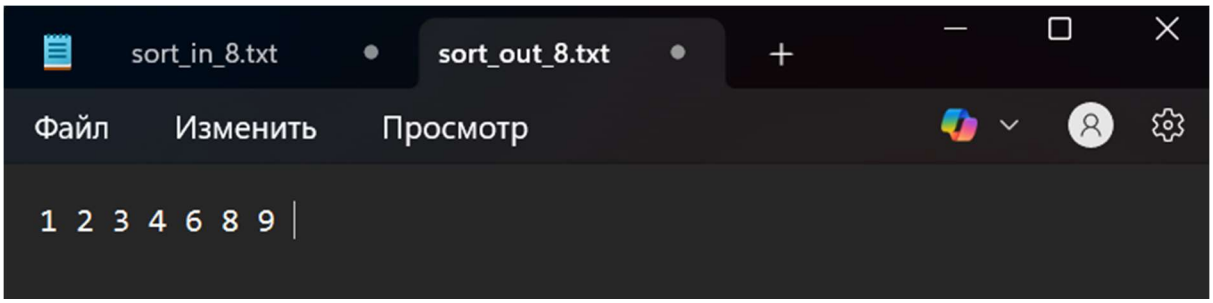

Рисунок Б.3 – Екран виконання програми для вирішення завдання Sort 3

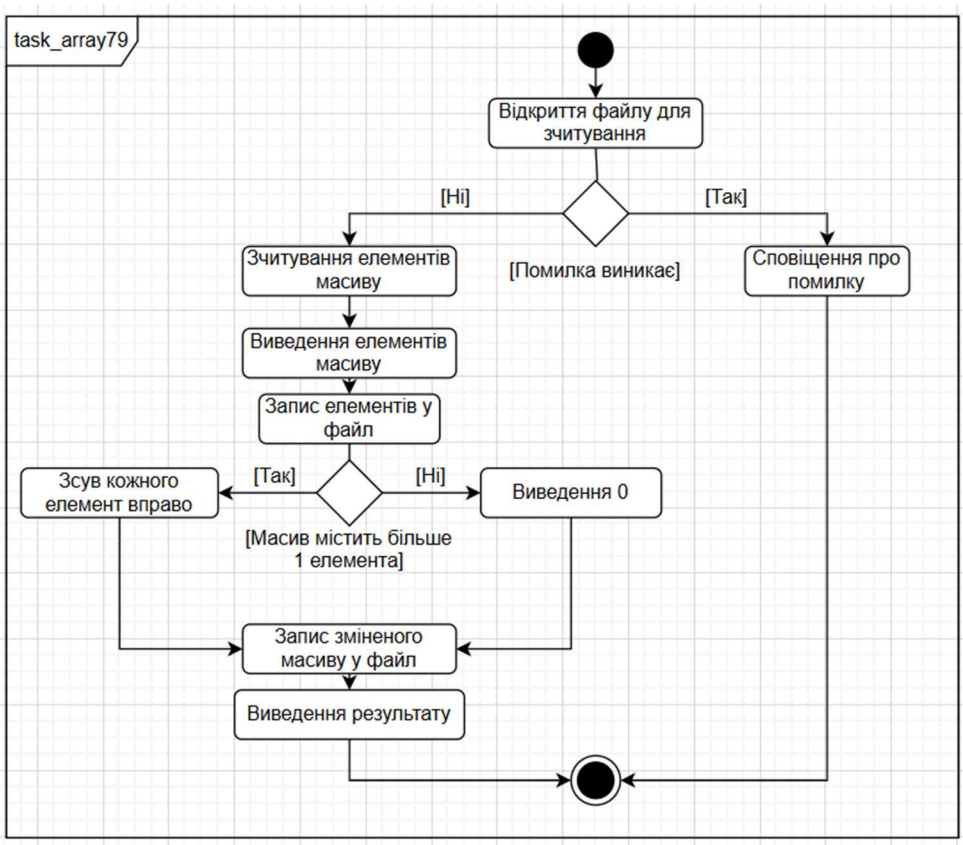

Рисунок Б.4 – Діаграма для завдання Array 79

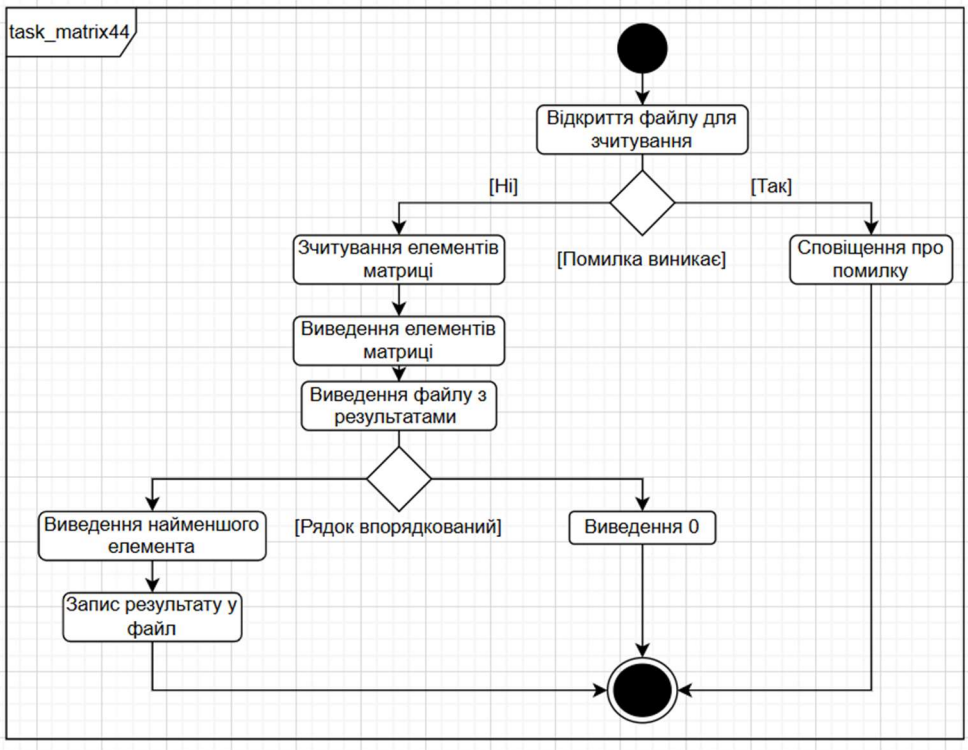

Рисунок Б.5 – Діаграма для завдання Matrix 44

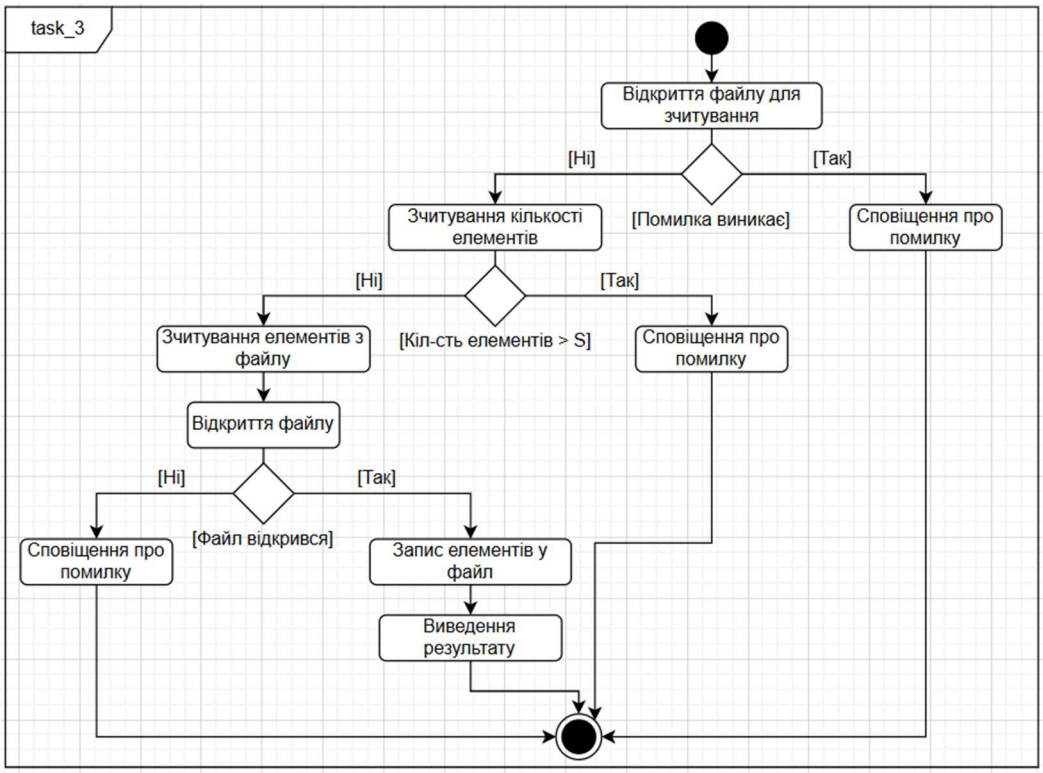

Рисунок Б.6 – Діаграма для функції зчитування масиву завдання 3

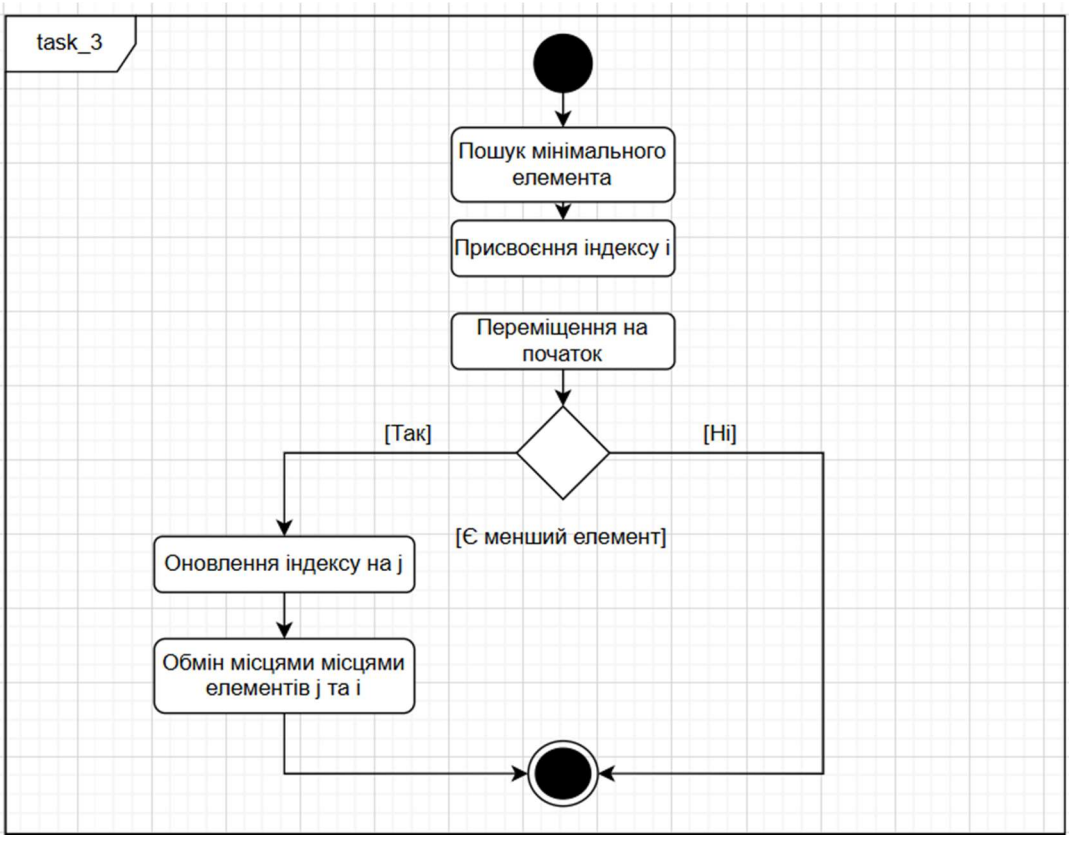

Рисунок Б.7 – Діаграма для функції сортування масиву завдання 3