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.301.173.310.02 ЛР

Виконав студент гр. _____310_____

18.03.2024_____Софія ПОЛЯКОВА_____
(підпис, дата) (П.І.Б.)

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

2023

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++ перетворити масив відповідно до свого варіанту завдання, ім'я файлу і необхідні змінні ввести з консолі. Вивести результати у файл «array_out_n.txt».

Array110. Дан цілочисельний масив розміру N. Продублювати в ньому всі парні числа.

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

Matrix21. Дана матриця розміру M × N. Для кожного рядка матриці з непарним номером (1, 3, ...) знайти середнє арифметичне її елементів. Умовний оператор не використовувати.

Завдання 3. Вивчити метод сортування відповідно до свого варіанту, проаналізувати його складність і продемонструвати на прикладі з 7-ми елементів (відповідно до свого варіанту). Реалізувати у вигляді окремої функції алгоритм сортування елементів масиву. Також окремими функціями реалізувати зчитування масиву з текстового файлу і виведення відсортованого масиву в консоль.

Sort 2. Метод сортування: Двійкові вставки, зменшення, дійсний.

Завдання 4. Для багаторазового виконання будь-якого з трьох зазначених вище завдань на вибір розробити алгоритм організації меню в командному вікні. Введення, виведення, обробку масивів реалізувати окремими функціями з параметрами.

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Array110.

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

1) файл «array_in110.txt» — для вводу даних у програму;
2) size — розмір масиву зчитаний з файлу, цілі числа;
3) arr[] — масив зчитаний з файлу, чілі числа

Вихідні дані (ім’я, опис, тип):

1) файл «array_out110.txt» — для виведення даних з програми;
2) newSize — перетворений масив, цілі числа;

Алгоритм вирішення показано на рис. 1 та 2

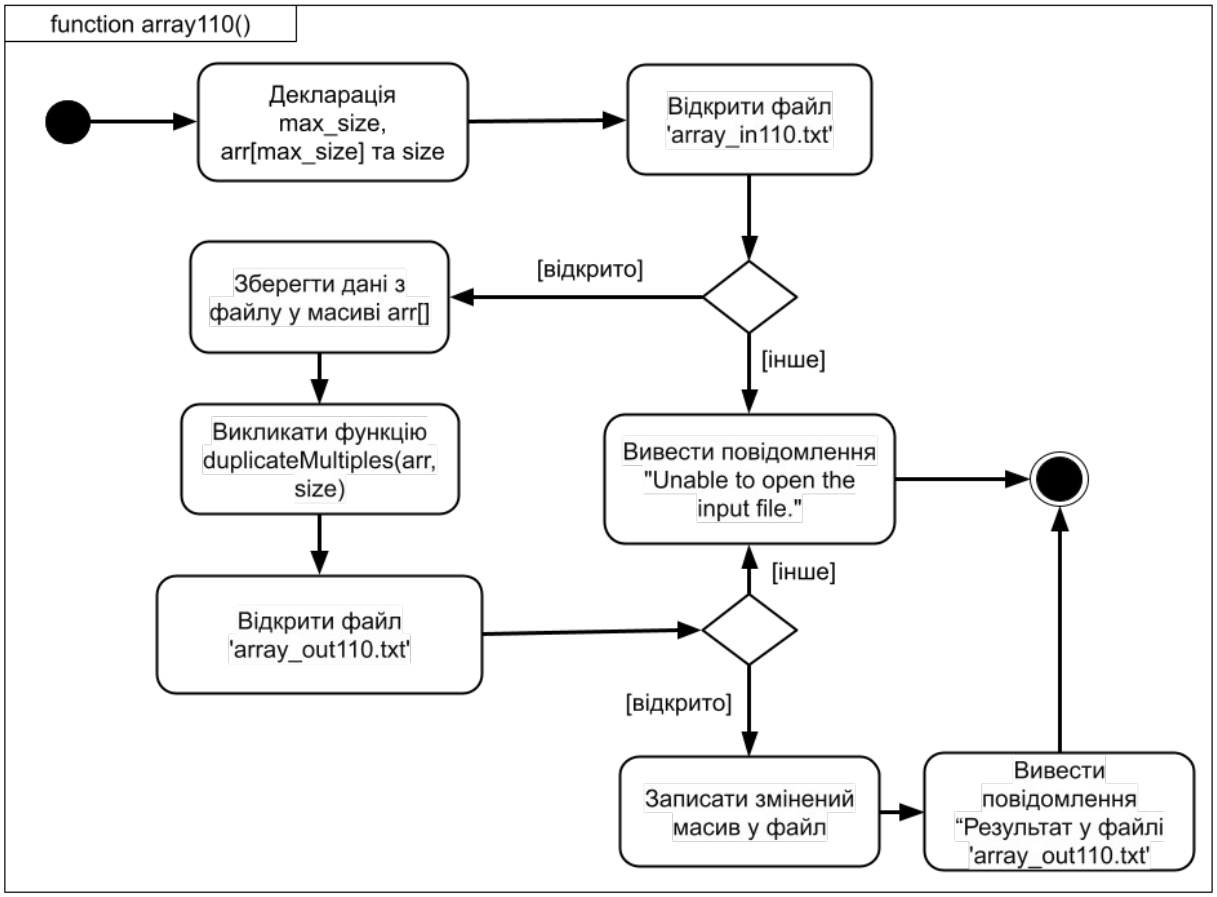

Рисунок 1 – Алгоритм вирішення Array110

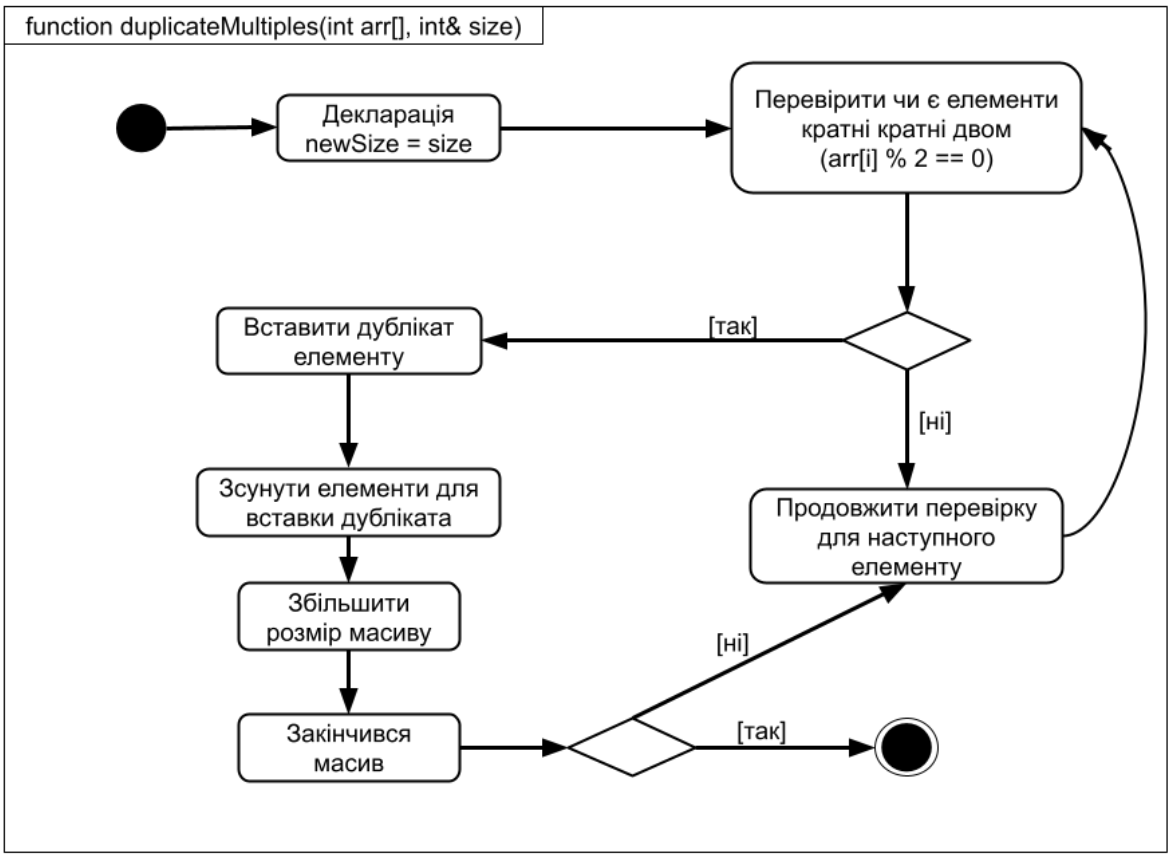

Рисунок 2 – Алгоритм вирішення Array110
(duplicateMultiples(int arr[], int& size)

Лістинг коду вирішення задачі Array110 наведено в дод. А (стор. 10).
Екран роботи програми показаний на рис. Б.1, Б.3-4.

Завдання 2.

Вирішення задачі Matrix21.

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

1) файл «matr_21.txt» — для вводу даних у програму;
2) matrix[M][N] — розміри матриці, цілі числа, 0 < matrix[M][N] < 20;
3) rowCount — введення рядків матриці, цілі числа;
4) colCount — введення стовпців матриці, цілі числа.

Вихідні дані (ім’я, опис, тип):

1) avg — середнє арифметичне непарних стовпців, дійсний тип;

Алгоритм вирішення показано на рис. 3

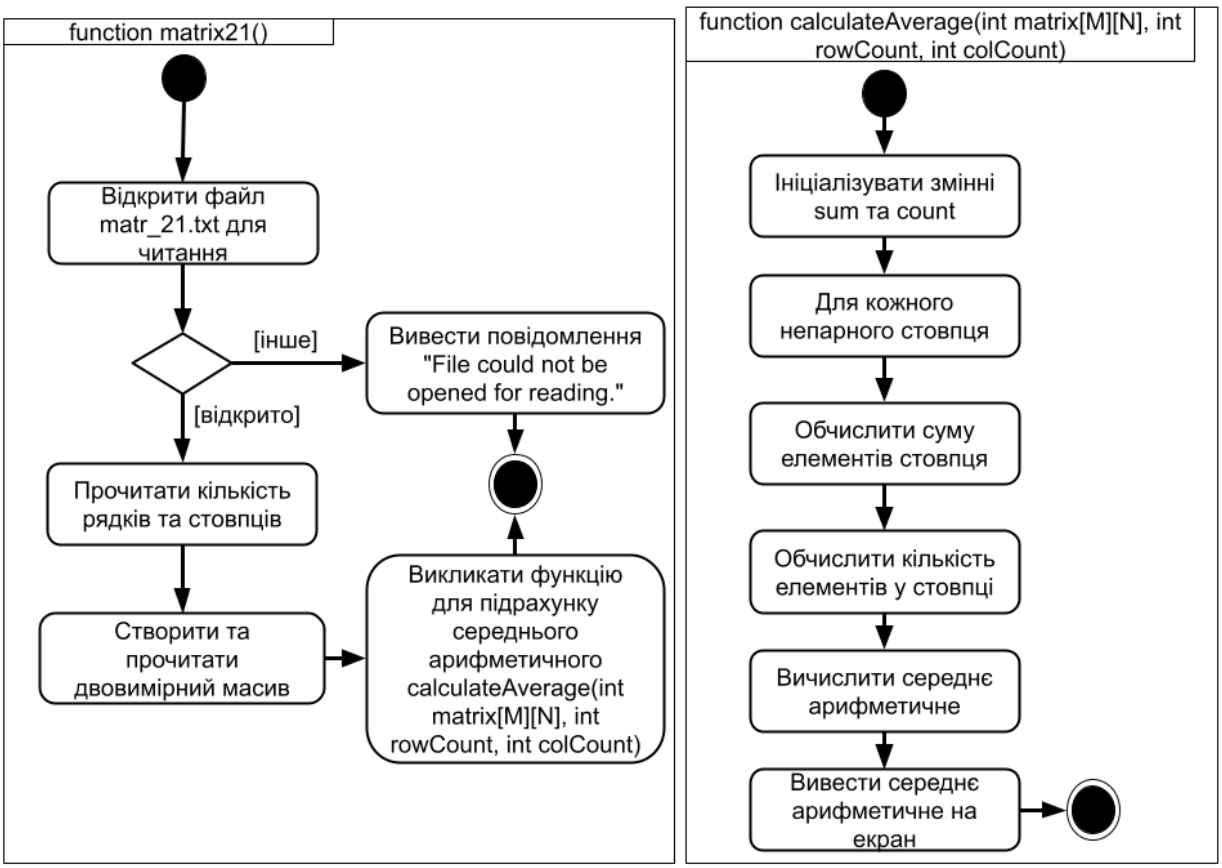

Рисунок 3 – Алгоритм вирішення Matrix21

Лістинг коду вирішення задачі Matrix21 наведено в дод. А (стор. 11).

Екран роботи програми показаний на рис. Б.1, Б.2.

Завдання 3.

Вирішення задачі Сортування 2 (Sort 2).

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

1) файл «sort.txt» — для вводу даних у програму;

2) n — ввидення розміру масива, ціле число;

3) arr — масив для сортування, дійсний тип.

Вихідні дані (ім’я, опис, тип):

2) arr[i] — відсортований дані, дійсний тип;

Алгоритм вирішення показано на рис. 4 та 5

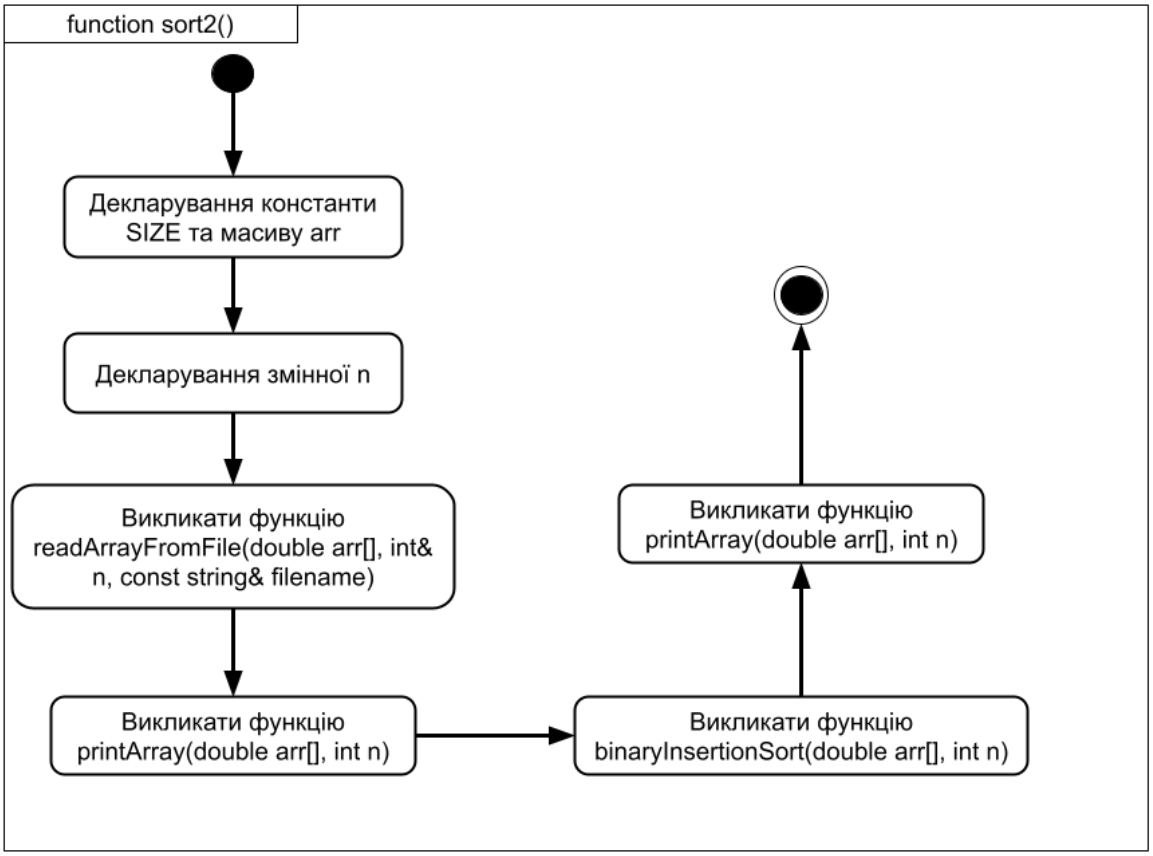

Рисунок 4 – Алгоритм вирішення Сортування 2 (Sort 2)

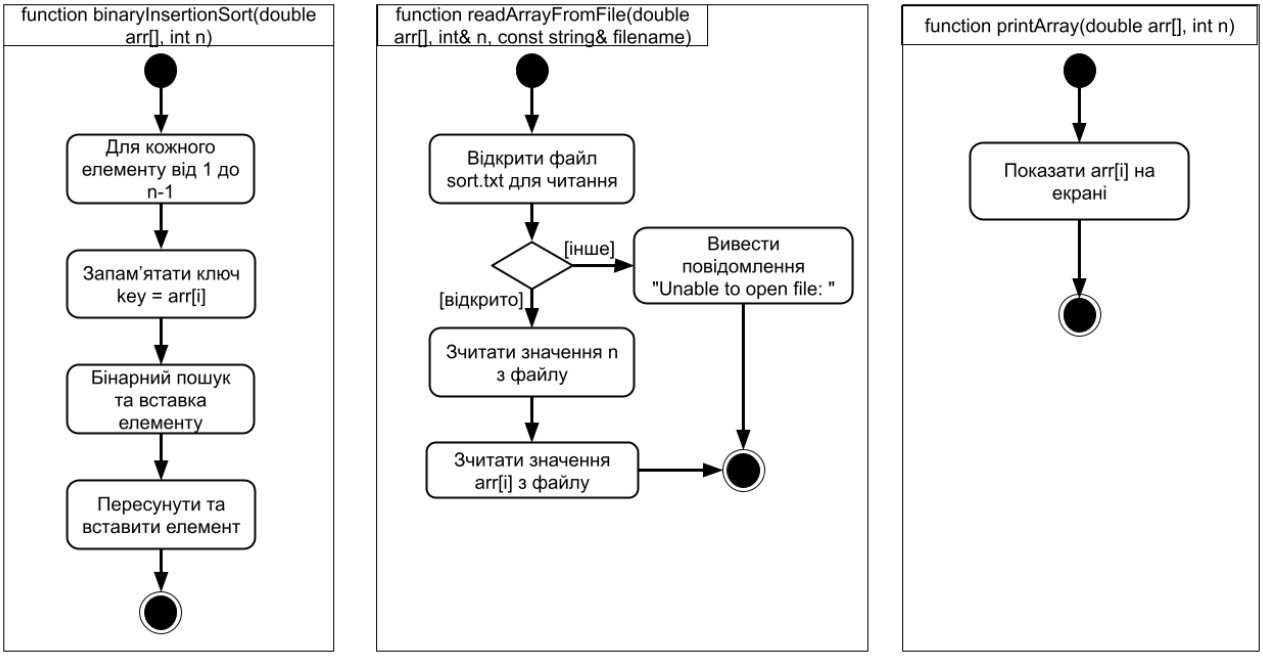

Рисунок 5 – Алгоритм вирішення Сортування 2 (Sort 2) (додаткові)

Лістинг коду вирішення задачі Сортування 2 (Sort 2) наведено в дод. А (стор. 12).

Екран роботи програми показаний на рис. Б.1, Б.5.

Завдання 4.

Вирішення задачі main () (Menu)

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

task_num – число, яке є номером задачі для вирішення, ціле число.

Вихідні дані (ім’я, опис, тип):

в залежності від введеного числа, перехід до виповнення задач 1, 2 або 3.

Алгоритм вирішення показано на рис. 6.

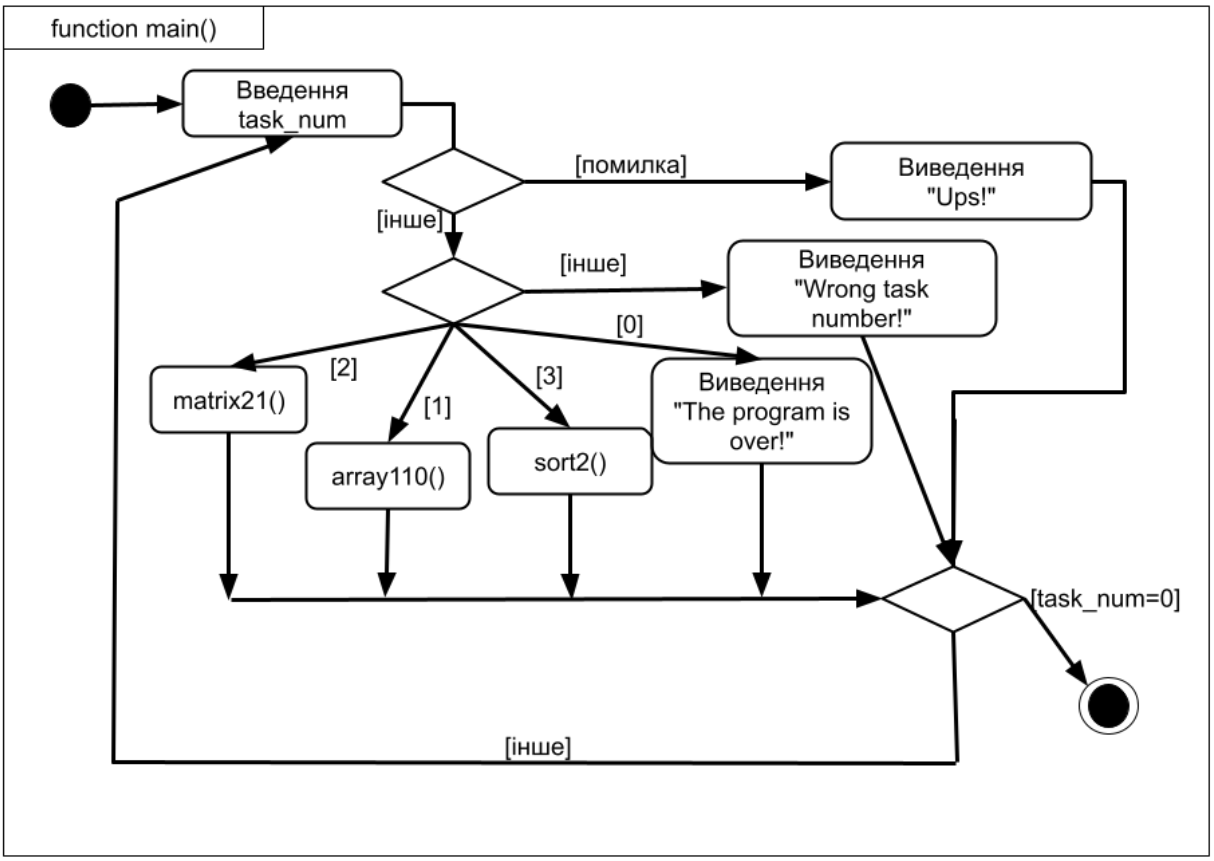

Рисунок 6 – Алгоритм вирішення main () (Menu)

Лістинг коду вирішення задачі main () (Menu) наведено в дод. А (стор. 10).

Екран роботи програми показаний на рис. Б.1.

ВИСНОВКИ

Вивчино теоретичний матеріал по алгоритмам обробки масивів на мові С++, а також бібліотеки для роботи з файлами і реалізовано оголошення, введення з файлу, обробку і виведення в файл одновимірних і двовимірних масивів на мові С ++. Закріплено на практиці структурування програм. Отримано навички з сортування масивів.

ДОДАТОК А
Лістинг коду програми

```
#include <iostream>
#include <fstream>
using namespace std;
const int N = 20;
const int M = 20;

void array110();
void duplicateMultiples(int arr[], int& size);

void matrix21();
double calculateAverage(int matrix[M][N], int rowCount, int colCount);

void sort2();
void binaryInsertionSort(double arr[], int n);
void readArrayFromFile(double arr[], int& n, const string& filename);
void printArray(double arr[], int n);

int main () { // menu
    int task_num;  // diclaration
    do {
        cout << " Task number (0 - exit):  ";
        cin >> task_num;
        if (!cin) {
            cout << " Ups!" << endl; continue;
            // error notification
        }
        switch (task_num) {
            case 1 : array110(); break; // task 1
            case 2 : matrix21(); break; // task 2
            case 3 : sort2(); break; // task 3
            case 0 : cout << " The program is over!" << endl; break;
            default : cout << " Wrong task number!" << endl;
            // output for incorrect numbers
        }
    } while (task_num != 0); // end the program
    return 0;
}
// task 1
void array110(){
    cout << " *** Task 1 Array 110 *** " << endl;
    const int max_size = 20; // maximum array size
    int arr[max_size];
    int size = 0;
    // entering data from a file
    ifstream filein("array_in110.txt");
    if (!filein.is_open()) {
```

```
        cout << "Unable to open the input file." << endl;
        return;
    }
    // read array size
    filein >> size;
    // reading array elements
    for (int i = 0; i < size; ++i) {
        filein >> arr[i];
    }
    filein.close();
    // duplication of multiples of two elements
    duplicateMultiples(arr, size);
    // outputting results to a file
    ofstream fileout("array_out110.txt");
    if (!fileout.is_open()) {
        cout << "Unable to open the output file." << endl;
        return;
    }
    // writing a modified array
    for (int i = 0; i < size; ++i) {
        fileout << arr[i] << " ";
    }
    fileout.close();
    cout << " Result in the file array_out110.txt " << endl;
    return;
}
// function for duplicating multiples of two elements
void duplicateMultiples(int arr[], int& size) {
    int newSize = size; // New array size
    for (int i = 0; i < size; ++i) {
        if (arr[i] % 2 == 0) {
            // increase the array size and insert a duplicate of it
            for (int j = newSize - 1; j > i; j--) {
                arr[j + 1] = arr[j];
            }
            arr[i + 1] = arr[i]; // inserting a duplicate
            ++newSize; // increase the array size
            ++i; // skip the element just inserted
        }
    }
    size = newSize; // updating the array size
}
// task 2
void matrix21() {
    cout << " *** Task 2 Matrix 21 *** " << endl;
    // opening a file for reading
    ifstream file;
    file.open("matr_21.txt");
    // checking whether the file was successfully opened
    if (!file.is_open()) {
```

```
        cout << "File could not be opened for reading." << endl;
        return;
    }
    // reading data from a file
    int rowCount, colCount;
    file >> rowCount >> colCount;
    // creating and reading a two-dimensional array
    int matrix[M][N];
    cout << "Matrix read from file:" << endl;
    for (int i = 0; i < rowCount; ++i) {
        for (int j = 0; j < colCount; ++j) {
            file >> matrix[i][j];
            cout << matrix[i][j] << " ";
        }
        cout << endl;
    }
    file.close();
    // function call to calculate the arithmetic mean of odd columns
    calculateAverage(matrix, rowCount, colCount);
    return;
}

// function for finding the arithmetic mean of column elements of a matrix
double calculateAverage(int matrix[M][N], int rowCount, int colCount) {
    double sum = 0.0;
    int count = 0;
    for (int j = 0; j < colCount; j += 2) {
        sum = 0;
        count = 0;
        for (int i = 0; i < rowCount; i++) {
            sum += matrix[i][j];
            count++;
        }
        double avg = sum / count;
        cout << "Average of column " << j+1 << ": " << avg << endl;
    }
    return 0.0;
}
// task 3
void sort2() {
    cout << " *** Task 3 Sort 2 *** " << endl;
    const int SIZE = 10;
    double arr[SIZE];
    int n;
    // reading an array from a file
    readArrayFromFile(arr, n, "sort.txt");
    // outputting the array
    cout << "Original array:" << endl;
    printArray(arr, n);
    // sort by binary insertion method
```

```
    binaryInsertionSort(arr, n);
    // outputting a sorted array
    cout << "Sorted array:" << endl;
    printArray(arr, n);
    return;
}
// binary search for a place to insert
void binaryInsertionSort(double arr[], int n) {
    for (int i = 1; i < n; ++i) {
        double key = arr[i];
        int left = 0;
        int right = i - 1;
        // binary search for a place to insert
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (arr[mid] > key)
                left = mid + 1;
            else
                right = mid - 1;
        }
        // move items to make room for insertion
        for (int j = i - 1; j >= left; --j) {
            arr[j + 1] = arr[j];
        }
        // insert an item into the found location
        arr[left] = key;
    }
}
// opening a file for reading
void readArrayFromFile(double arr[], int& n, const string& filename) {
    ifstream file(filename);
    if (file.is_open()) {
        file >> n;
        for (int i = 0; i < n; ++i) {
            file >> arr[i];
        }
        file.close();
    } else {
        cout << "Unable to open file: " << filename << endl;
    }
}
// function for displaying an array on the screen
void printArray(double arr[], int n) {
    for (int i = 0; i < n; ++i) {
        cout << arr[i] << " ";
    }
    cout << endl;
}
```

ДОДАТОК Б

Скрін-шоти вікна виконання програми

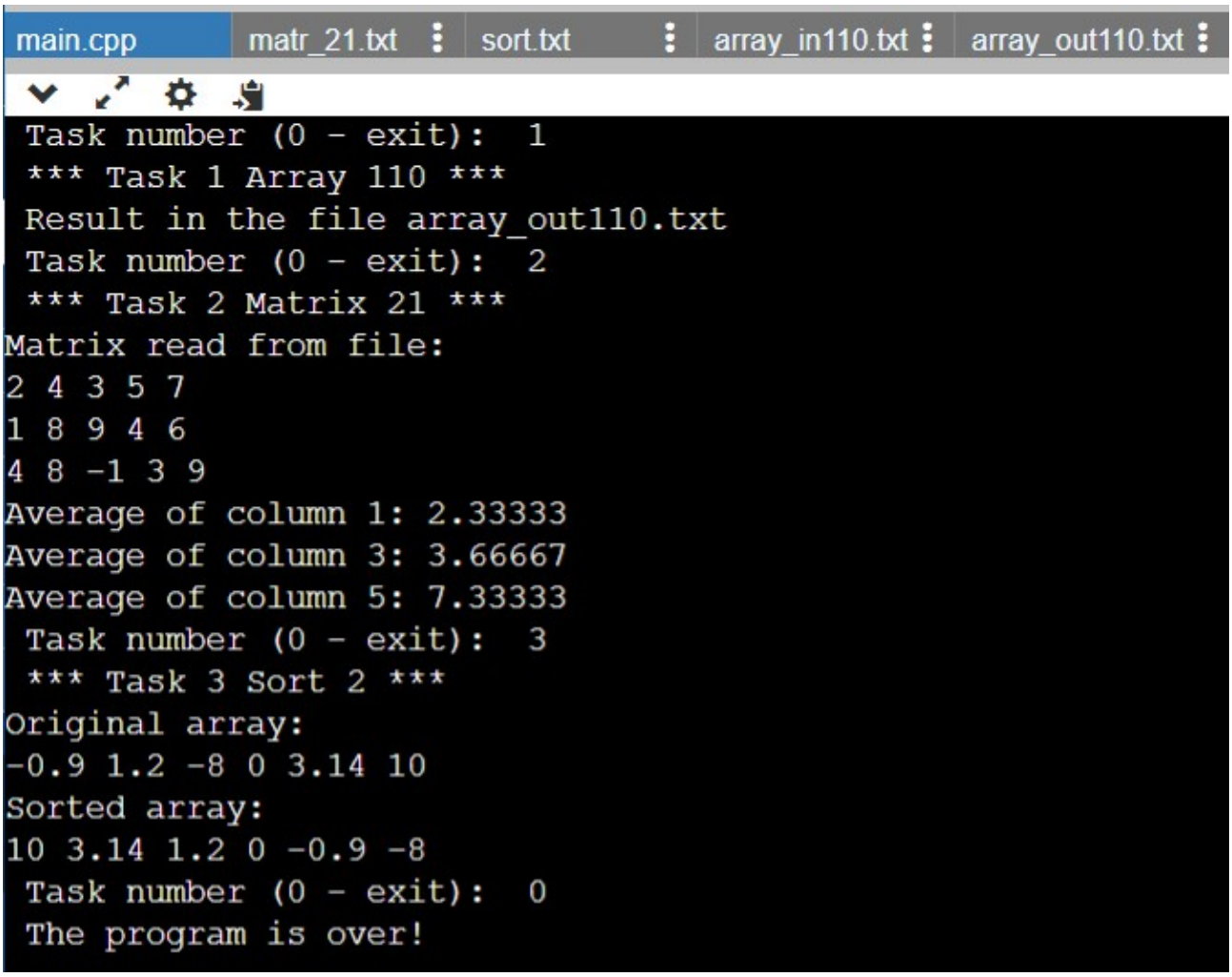

Рисунок Б.1 – Екран виконання програми для вирішення завдання

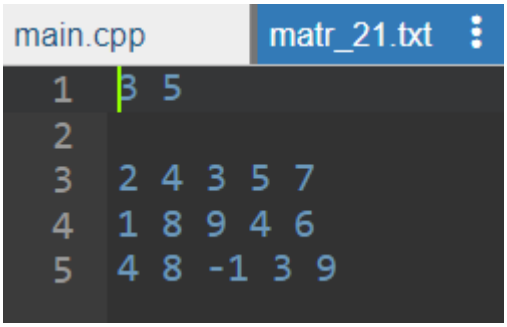

Рисунок Б.2 – Екран виконання програми для вирішення завдання Matrix21

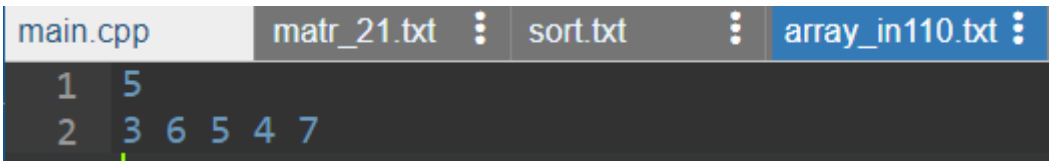

Рисунок Б.3 – Екран виконання програми для вирішення завдання Array110

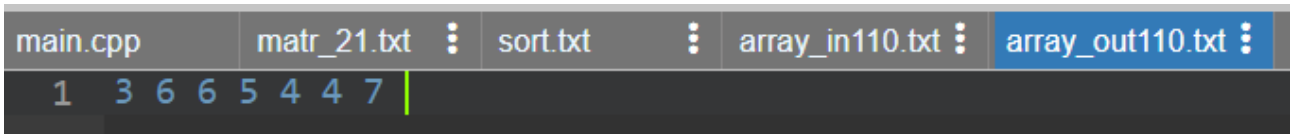

Рисунок Б.4 – Екран виконання програми для вирішення завдання Array110

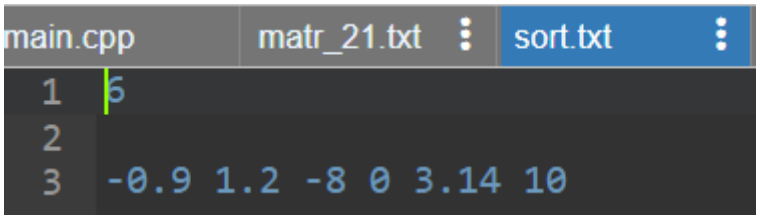

Рисунок Б.5 – Екран виконання програми для вирішення завдання Сортування 2 (Sort 2)