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»
Тема: "Реалізація алгоритмів сортування та робота з файлами на мові С ++"

ХАІ.301.141.319а. 27 ЛР

Виконав студент гр. 319а
Владислав РУДНЄВ
(Підпис,дата) (П.І.Б.)

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

2024

МЕТА РОБОТИ

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

Array72. Дано масив A розміру N і цілі числа K і L ($1 \leq K < L \leq N$). Переставити взворотному порядку елементи масиву, розташовані між елементами AK і AL, включаючи ці елементи.

Завдання 2. За допомогою текстового редактору створити текстовий файл«matr_in_n.txt» з елементами вихідного двовимірного масиву (n - номер варіанту). 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.

Matrix20. Дана матриця розміру M × N. Для кожного стовпця матриці знайти добуток його елементів.

Завдання 3. 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.

Варіант27. Двійкові вставки, Зростання, Цілий

ВИКОНАННЯ РОБОТИ

Завдання 1.

Вирішення задачі Array72.

Вхідні дані:

- Масив чисел (arr): Одновимірний масив цілих чисел, зчитується з файлу.
- Розмір масиву (n): Кількість елементів у масиві.
- Індекси для реверсу підмасиву (k, l): Початковий та кінцевий індекси підмасиву (користувацьке введення), які задовольняють умову: $1 \leq k < l \leq n$.

Вихідні дані:

- Початковий масив (arr): Масив до змін.
- Модифікований масив (modifiedArr): Масив після того, як підмасив між k та l переставлено у зворотному порядку.
- Повідомлення про помилки (errorMsg): Виводиться у випадку, якщо $k < 1$, $l > n$, або $k \geq l$.

Алгоритм:

1. Зчитування масиву з файлу.
2. Введення значень k і l.
3. Перевірка коректності: чи $1 \leq k < l \leq n$.
4. Якщо перевірка успішна, викликається функція reverseSubarray(arr, k, l), яка змінює порядок елементів між індексами k-1 та l-1.
5. Запис результату у файл та виведення зміненого масиву.

Завдання 2.

Вирішення задачі Matrix20.

Вхідні дані:

- Матриця (matrix): Двовимірний масив цілих чисел розміром M x N, зчитується з файлу.
- Розміри матриці (M, N): Кількість рядків (M) і стовпців (N).

  Вихідні дані:

- Початкова матриця: Матриця, яку користувач зчитав з файлу.

- Добутки стовпців: Одновимірний масив, який містить добуток усіх елементів кожного стовпця.
- Оновлений файл: До файлу додається результат у вигляді:

  Product of each column:

  <добуток_1> <добуток_2> ... <добуток_N>

  Алгоритм:

  1. Зчитати розміри матриці M і N, а також її елементи з файлу.
  2. Ініціалізувати змінні для збереження добутків кожного стовпця.
  3. Для кожного стовпця обчислити добуток відповідних елементів.
  4. Відкрити вхідний файл у режимі append та додати результат.
  5. Вивести повідомлення про успішне завершення або помилку, якщо файл не зчитано.

Завдання 3.

Виконання задачі 27

Вхідні дані:

- Масив чисел (arr): Одновимірний масив цілих чисел, зчитується з файлу.
- Розмір масиву (n): Кількість елементів у масиві.

Вихідні дані:

- Початковий масив (arr): Масив до сортування.
- Відсортований масив (sortedArr): Масив після бінарного сортування вставками.
- Оновлений файл: Вивід результату у новий файл або заміна вмісту старого.

Алгоритм:

1. Зчитати масив з файлу.

2. Для кожного елементу arr[i] (починаючи з індексу 1):
    - Знайти правильну позицію для вставки за допомогою бінарного пошуку.
    - Зсунути елементи правіше, щоб звільнити місце.
    - Вставити поточне значення у знайдену позицію.

3. Записати відсортований масив у файл.
4. Вивести повідомлення про завершення, або помилку, якщо файл порожній.

ВИСНОВОК

У ході роботи було вивчено алгоритми обробки масивів у мові C++ та бібліотеки для роботи з файлами. Реалізовано оголошення, зчитування з файлу, обробку та запис у файл як одновимірних, так і двовимірних масивів. Застосовано сортування, реверс підмасиву та обчислення добутку стовпців матриці. Усі завдання виконані у середовищі Visual Studio, що дозволило закріпити знання з програмування та роботи з файлами.

ДОДАТОК А

Лістинінг коду

//main.cpp

```
#include <iostream>  // Підключення бібліотеки для роботи з введенням/виведенням
#include "array_utils.h"  // Підключення заголовочного файлу для масивів
#include "matrix_utils.h" // Підключення заголовочного файлу для матриць

using namespace std; // Використання стандартного простору імен (щоб не писати std::)

void menu() {
   while (true) { // Нескінченний цикл для відображення меню, поки користувач не вибере вихід
      int choice;
      cout << "\nMenu:\n";
      cout << "1. Task 1: Reverse subarray between K and L\n"; // Опція 1 — переворот підмасиву
        cout << "2. Task 2: Calculate product of each column\n"; // Опція 2 — добуток елементів у кожному
стовпці матриці
      cout << "3. Task 3: Binary Insertion Sort\n"; // Опція 3 — бінарне сортування вставками
      cout << "4. Exit\n"; // Вихід з програми
      cout << "Choose an option: ";
      cin >> choice;

      switch (choice) { // Обробка вибору користувача
      case 1:
         task1(); // Виконати завдання 1
         break;
      case 2:
         task2(); // Виконати завдання 2
         break;
      case 3:
         task3(); // Виконати завдання 3
         break;
      case 4:
         return; // Вихід з меню
      default:
         cout << "Invalid choice, please try again.\n"; // Повідомлення про невірний вибір
      }
   }
}

int main() {
   menu(); // Запуск меню при старті програми
   return 0;
}
```

//array_utils.h

```
#pragma once
#ifndef ARRAY_UTILS_H
#define ARRAY_UTILS_H

#include <vector>
#include <string>
// Зчитування масиву з файлу
void inputArrayFromFile(std::vector<int>& arr, int& n, const std::string& filename);

// Запис масиву у файл
void outputArrayToFile(const std::vector<int>& arr, const std::string& filename);

// Реверс частини масиву між індексами k та l
void reverseSubarray(std::vector<int>& arr, int k, int l);

// Сортування масиву методом бінарної вставки
void binaryInsertionSort(std::vector<int>& arr);

// Завдання 1: реверс підмасиву
void task1();

// Завдання 3: сортування масиву
void task3();

#endif
```

//array_utils.cpp

```
#include "array_utils.h"
#include <iostream>
#include <fstream> // Для роботи з файлами
#include <algorithm> // Для std::reverse

using namespace std;

// Зчитування масиву з файлу
void inputArrayFromFile(vector<int>& arr, int& n, const string& filename) {
    ifstream infile(filename); // Відкриття файлу для читання
    if (!infile) {
        cerr << "Cannot open file: " << filename << endl;
        return;
    }

    int num;
    while (infile >> num) { // Зчитування чисел до кінця файлу
        arr.push_back(num);
    }
    n = arr.size(); // Встановлення розміру масиву
```

```
    infile.close(); // Закриття файлу
}

// Запис масиву у файл
void outputArrayToFile(const vector<int>& arr, const string& filename) {
    ofstream outfile(filename); // Відкриття файлу для запису
    for (int val : arr) {
        outfile << val << " "; // Запис елементів через пробіл
    }
    outfile << endl;
    outfile.close(); // Закриття файлу
}

// Реверс частини масиву від індексу (k-1) до (l-1)
void reverseSubarray(vector<int>& arr, int k, int l) {
    reverse(arr.begin() + (k - 1), arr.begin() + l);
}

// Бінарне сортування вставками
void binaryInsertionSort(vector<int>& arr) {
    for (int i = 1; i < arr.size(); ++i) {
        int key = arr[i];
        int left = 0, right = i;

        // Знаходимо позицію вставки за допомогою бінарного пошуку
        while (left < right) {
            int mid = (left + right) / 2;
            if (key < arr[mid])
                right = mid;
            else
                left = mid + 1;
        }

        // Зсуваємо елементи і вставляємо key у правильну позицію
        for (int j = i; j > left; --j) {
            arr[j] = arr[j - 1];
        }

        arr[left] = key;
    }
}

// Виконання завдання 1
void task1() {
    int n, k, l;
    vector<int> arr;
    string inputFileName, outputFileName;
```

```
    cout << "Enter input filename: ";
    cin >> inputFileName;
    cout << "Enter output filename: ";
    cin >> outputFileName;

    inputArrayFromFile(arr, n, inputFileName);
    if (arr.empty()) {
        cout << "Input array is empty or file could not be read." << endl;
        return;
    }

    cout << "Enter K and L (1 ≤ K < L ≤ N): ";
    cin >> k >> l;

    // Перевірка правильності значень
    if (k < 1 || l < 1 || k >= l || l > n) {
        cout << "Invalid values for K and L. Ensure that 1 ≤ K < L ≤ N." << endl;
        return;
    }

    reverseSubarray(arr, k, l);
    outputArrayToFile(arr, outputFileName);
    cout << "Modified array written to " << outputFileName << endl;
}

// Виконання завдання 3
void task3() {
    int n;
    vector<int> arr;
    string inputFileName, outputFileName;

    cout << "Enter input filename: ";
    cin >> inputFileName;
    cout << "Enter output filename: ";
    cin >> outputFileName;

    inputArrayFromFile(arr, n, inputFileName);
    if (arr.empty()) {
        cout << "Input array is empty or file could not be read." << endl;
        return;
    }

    binaryInsertionSort(arr);
    outputArrayToFile(arr, outputFileName);

    cout << "Sorted array written to " << outputFileName << endl;
}
```

//matrix_utils.h

```
#pragma once // Гарантія одноразового включення

#ifndef MATRIX_UTILS_H
#define MATRIX_UTILS_H

#include <vector>  // Для std::vector
#include <string>  // Для std::string

// Зчитування матриці з файлу
void inputMatrixFromFile(std::vector<std::vector<int>>& matrix, int& M, int& N, const std::string& filename);

// Запис добутку кожного стовпця в кінець файлу
void appendColumnProductsToFile(const std::vector<std::vector<int>>& matrix, int M, int N, const
std::string& filename);

// Завдання 2: розрахунок добутку кожного стовпця
void task2();

#endif
```

//matrix_untils.cpp

```
#include "matrix_utils.h"
#include <iostream>
#include <fstream> // Для роботи з файлами

using namespace std;

// Зчитування матриці з файлу
void inputMatrixFromFile(vector<vector<int>>& matrix, int& M, int& N, const string& filename) {
    ifstream infile(filename); // Відкриття файлу для читання
    if (!infile) {
        cerr << "Cannot open file: " << filename << endl;
        return;
    }

    infile >> M >> N; // Зчитування розмірів матриці
    matrix.resize(M, vector<int>(N)); // Зміна розміру матриці

    // Зчитування елементів матриці
    for (int i = 0; i < M; ++i)
        for (int j = 0; j < N; ++j)
            infile >> matrix[i][j];

    infile.close(); // Закриття файлу
}

// Додавання в кінець файлу добутку кожного стовпця матриці
```

```
void appendColumnProductsToFile(const vector<vector<int>>& matrix, int M, int N, const string& filename) {
    ofstream outfile(filename, ios::app); // Відкриття файлу для дозапису (append)
    outfile << "\nProduct of each column:\n";
    for (int j = 0; j < N; ++j) {
        int product = 1;
        for (int i = 0; i < M; ++i) {
            product *= matrix[i][j];
        }
        outfile << product << " ";
    }
    outfile << endl;
    outfile.close();
}

// Виконання завдання 2
void task2() {
    int M, N;
    vector<vector<int>> matrix;
    string filename;

    cout << "Enter matrix filename: ";
    cin >> filename;

    inputMatrixFromFile(matrix, M, N, filename);
    if (matrix.empty()) {
        cout << "Matrix could not be loaded." << endl;
        return;
    }

    appendColumnProductsToFile(matrix, M, N, filename);
    cout << "Column products appended to file: " << filename << endl;
}
```

ДОДАТОК Б

Скрін-шоти вікна виконання програми та діаграми

На рис.1 показано код виконання програми Лабораторна робота№8

```
Menu:
1. Task 1: Reverse subarray between K and L
2. Task 2: Calculate product of each column
3. Task 3: Binary Insertion Sort
4. Exit
Choose an option: 1
Enter input filename: array_in_15
Enter output filename: array_out_15
Cannot open file: array_in_15
Input array is empty or file could not be read.

Menu:
1. Task 1: Reverse subarray between K and L
2. Task 2: Calculate product of each column
3. Task 3: Binary Insertion Sort
4. Exit
Choose an option: 1
Enter input filename: array_in_15.txt
Enter output filename: array_out_15.txt
Enter K and L (1 ? K < L ? N): 3 5
Modified array written to array_out_15.txt

Menu:
1. Task 1: Reverse subarray between K and L
2. Task 2: Calculate product of each column
3. Task 3: Binary Insertion Sort
4. Exit
Choose an option: 2
Enter matrix filename: matr_in_15
Cannot open file: matr_in_15
Matrix could not be loaded.

Menu:
1. Task 1: Reverse subarray between K and L
2. Task 2: Calculate product of each column
3. Task 3: Binary Insertion Sort
4. Exit
Choose an option: 2
Enter matrix filename: matr_in_15.txt
Column products appended to file: matr_in_15.txt

Menu:
1. Task 1: Reverse subarray between K and L
2. Task 2: Calculate product of each column
3. Task 3: Binary Insertion Sort
4. Exit
Choose an option: 3
Enter input filename: array_sort_in.txt
Enter output filename: array_sort_out.txt
Sorted array written to array_sort_out.txt

Menu:
1. Task 1: Reverse subarray between K and L
2. Task 2: Calculate product of each column
3. Task 3: Binary Insertion Sort
4. Exit
Choose an option: 4
```

Рисунок 1 — виконання Лабораторної роботи №8

На рис.2 показано диаграму коду програми для завдання 3

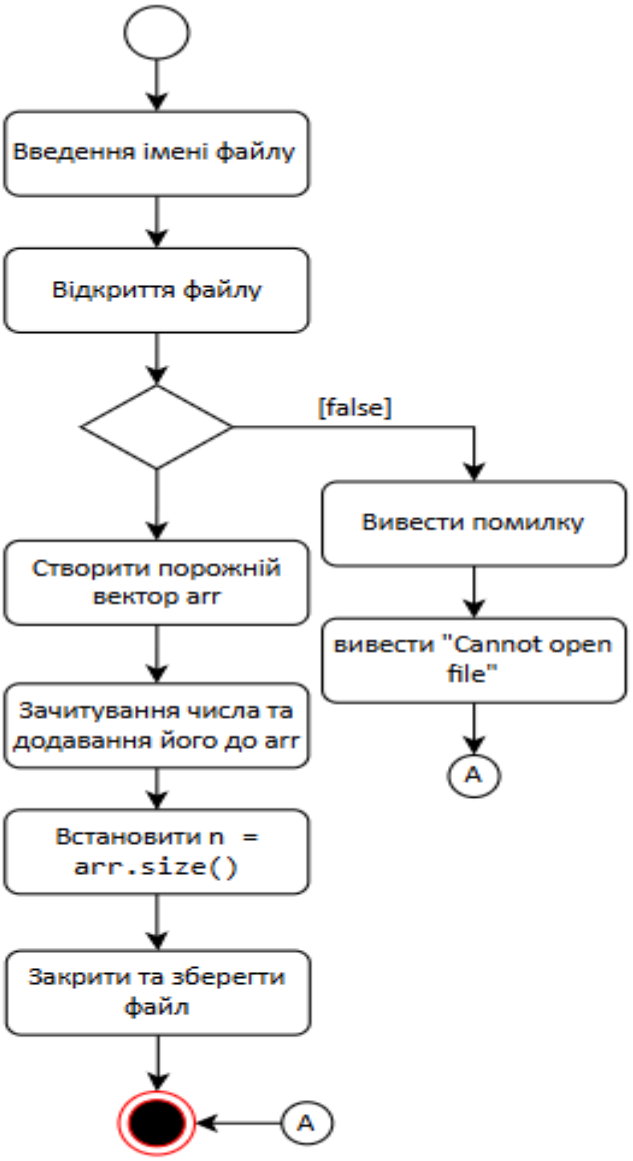

Рисунок 2 — Діаграма активності коду Варіанту 27 .

На рис.3 показано диаграму коду програми для завдання 3

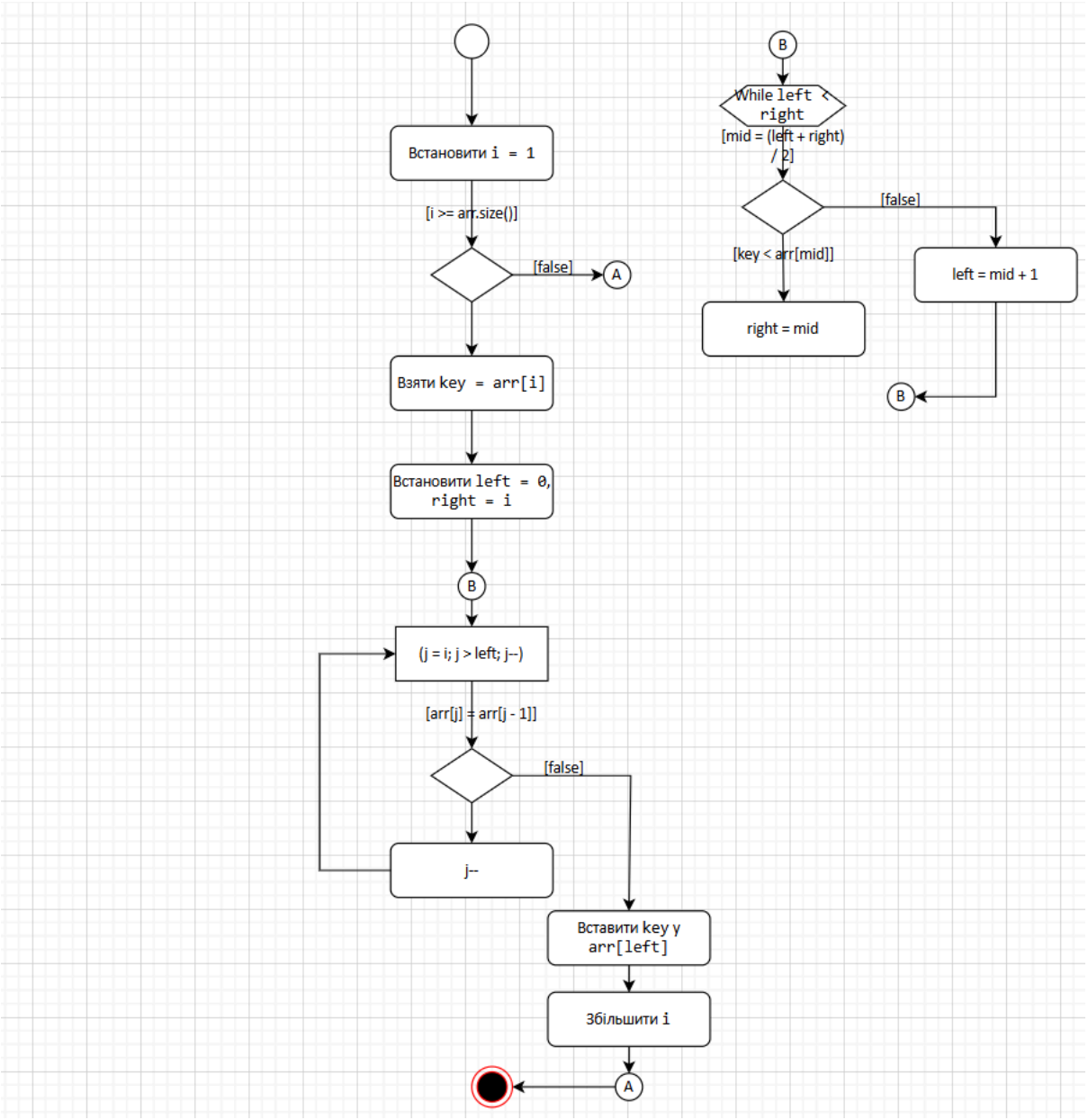

Рисунок 3 — Діаграма активності коду для Варіанту 27.