**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»**

**ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

Кафедра інформаційних технологій

**КУРСОВА РОБОТА  
з дисципліни «Алгоритмізація та програмування»**

**на тему: «Програмне опрацювання даних у файлах»**

студента 1 курсу групи ІПЗ-111

спеціальності 121 «Інженерія програмного забезпечення»  
Кожухаря Володимира Олександровича

Керівник доцент кафедри ІТ

к.т.н., доцент Трофименко О. Г.   
Національна шкала

Кількість балів \_\_\_\_\_ Оцінка ECTS

Члени комісії:

підпис прізвище та ініціали

підпис прізвище та ініціали

м. Одеса – 2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»

ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інформаційних технологій

ЗАВДАННЯ НА КУРСОВУ РОБОТУ

з дисципліни «Алгоритмізація та програмування»

студента Кожухаря Володимира Олександровича

спеціальності 121 Інженерія програмного забезпечення

першого курсу групи 111

Тема «Програмне опрацювання даних у файлах»

Варіант 8

Вхідні дані:

− текстовий файл F1.txt із числовими даними вимірювань датчиків у вигляді матриці розмірності M на N (M – кількість датчиків, N – кількість вимірювань);

− файл F2.txt з текстом вихідних результатів;

Курсова робота виконується в такому обсязі:

I. Розрахунково-пояснювальна записка:

− аналіз засобів та методів розробки;

− словесний опис алгоритмів основного модуля;

− опис функцій розробленої бібліотеки StaticLibForCoursework;

− аналіз досягнутих результатів;

− інструкція з використання програми;

II. Графічна частина:

− схеми алгоритмів основного модуля для виконання завдань;

− схеми алгоритмів функцій бібліотеки StaticLibForCoursework.

Індивідуальне завдання:

1. Розробити алгоритми та написати програмний код для формування з даних текстового файлу F1.txt числової матриці та розв’язання таких завдань:

* обчислити кількість показників вимірювань датчиків (елементів матриці), більших за 1;
* елемент матриці називається локальним мінімумом, якщо він строго менше всіх сусідів нього (ліворуч, праворуч, зверху і знизу), які є у нього. Підрахувати кількість локальних мінімумів заданої матриці;
* обчислити вектор як суми від’ємних елементів стовпців матриці;
* \*створити динамічний масив із ненульових показників усіх датчиків;
* \*\* характеристикою рядка матриці назвемо квадрат суми його елементів. Переставляючи рядки заданої матриці, розташувати їх відповідно до зростання характеристик.

2. Програмно записати результати обчислень у новий файл, наприклад, з ім’ям F2.txt.

3. Як вхідні дані для програмного опрацювання текстових даних створити ще один файл з ім’ям F3.txt та заповнити його коректними текстовими даними та розв’язати програмно такі завдання:

* програмно переглянути на екрані вміст текстового файлу F3.txt, ім’я якого передаватиметься як аргумент із функції main(). Визначити, чи є у рядках цифри. Якщо так, то вивести їх. Відсортувати слова за абеткою у кожному рядку. Програмно записати результати опрацювання текстових даних у файл F4. Програмно переглянути на екрані вміст текстового файлу з результатами опрацювання текстових даних;
* Відредагувати файл F3.txt:
  + рядок, заданий номером (номер ввести з клавіатури), після чого переглянути на екрані його вміст;
  + рядки, які містять заданий текст (текст ввести з клавіатури), після чого переглянути на екрані його вміст;
* Видалити з файлу F3.txt:
  + рядок, заданий номером, після чого переглянути на екрані його вміст;
  + рядки, які містять заданий текст, після чого переглянути на екрані його вміст.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| № | Завдання | Термін виконання |
| 1 | Ознайомлення з завданням до курсової роботи. Підготування вхідних файлів. Опрацювання матриць. | 12.03.2024 – 15.03.2024 |
| 2 | Опрацювання матриць. | 15.03.2024 – 16.03.2024 |
| 3 | Опрацювання динамічних масивів. | 17.03.2024 – 18.03.2024 |
| 4 | Опрацювання рядкових даних. | 21.03.2024 – 25.03.2024 |
| 5 | Опрацювання текстових файлів. | 26.03.2024 – 01.04.2024 |
| 6 | Створення бібліотеки функцій. | 01.04.2024 – 01.04.2024 |
| 7 | Оформлення пояснювальної записки | 19.03.2024 – 10.04.2024 |

Зміст

[Вступ 6](#_Toc164017195)

[1 Аналіз засобів та методів розробки 7](#_Toc164017196)

[1.1 Методи розробки програми, бібліотеки 7](#_Toc164017197)

[1.2 Середовище розробки 8](#_Toc164017198)

[2 Словесний опис алгоритмів для розв’язання задач 9](#_Toc164017199)

[2.1 Опис функції main() 9](#_Toc164017200)

[2.2 Опис функції view\_file() 12](#_Toc164017201)

[2.3 Опис функції array\_RowsColumns() 13](#_Toc164017202)

[2.4 Опис функції make\_array 14](#_Toc164017203)

[2.5 Опис функції make\_output3() 15](#_Toc164017204)

[2.6 Опис функції BiggerThanOne() 16](#_Toc164017205)

[2.7 Опис функції LocalMinimum() 17](#_Toc164017206)

[2.8 Опис функції VectorNegativeNumbersColumns() 19](#_Toc164017207)

[2.9 Опис функції DynamicArrayNonZero() 20](#_Toc164017208)

[2.10 Опис функції SortBySumSquares() 22](#_Toc164017209)

[2.11 Опис функції NumsInRows() 23](#_Toc164017210)

[2.12 Опис функції SortByAlphabet() 25](#_Toc164017211)

[2.13 Опис функції PrintRowByNum() 28](#_Toc164017212)

[2.14 Опис функції PrintRowByText() 30](#_Toc164017213)

[2.15 Опис функції DeleteRowByNum() 32](#_Toc164017214)

[2.16 Опис функції DeleteRowByText() 34](#_Toc164017215)

[3 Результати Виконання програми 37](#_Toc164017216)

[4 Інструкція з використання програми 41](#_Toc164017217)

[Висновки з аналізом досягнутих результатів 42](#_Toc164017218)

[Список використаних джерел 43](#_Toc164017219)

[Додатки 44](#_Toc164017220)

[Додаток А. Схеми алгоритмів функцій програмного проєкту 44](#_Toc164017221)

[Додаток Б. Програмний код головного файлу 62](#_Toc164017222)

[Додаток В. Програмний код бібліотеки функцій 63](#_Toc164017223)

# Вступ

Важливість цієї роботи полягає у перевірці знань та можливостей реалізації багатьох програмних завдань, зокрема використання мови програмування C++, обробки текстових файлів та даних у них, запису результатів у файли, створенні функцій та власних бібліотек функцій і їх підключення до проєкту.

Актуальність теми роботи: опрацювання текстових даних наразі є дуже високою, такі завдання як пошук слів та заміна на введене слово, видалення слів після їх пошуку. Всі ці функції можливо використати в майбутньому для створення свого редактора документів та для інших ідей з обробкою тексту.

У цифровому світі сучасності неможливо уникнути обробки файлів і це очевидно навіть зараз, коли для написання цього документа використовується програма Word, для створення та редагування документів різних форматів.

**Об'єкт дослідження –** базові технології програмування алгоритмічною мовою високого рівня і засоби створення програмних проєктів.

**Предмет дослідження** – програмні засоби мови C++ для обробки різних типів даних. Це включає створення файлів різних форматів, запис даних у них, виведення вмісту файлів та вибір даних за певними критеріями. Також можливе редагування даних та створення нових файлів з певними даними з основного файлу. Використовуються різні методи для опрацювання даних різних типів.

**Мета роботи** – розробка програми, яка надає користувачеві можливість працювати з різнорідними даними, збереженими у файлах, виконуючи такі операції як зчитування, запис, обробка та виведення даних на екран, застосування знань, набутих протягом навчального періоду, для розв'язання різноманітних завдань на різних рівнях складності.

Відповідно до поставленої мети **завданнями** роботи є:

– аналіз засобів та методів розробки програмного проєкту;

– аналіз засобів створення та використання бібліотеки функцій;

– словесний опис алгоритмів для розв’язання задач;

– аналіз результатів роботи програми.

Структура роботи: курсова робота складається із вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг курсової роботи: 71 с., 23 рис., 1 табл., 3 додатки, 9 джерел.

# 1 Аналіз засобів та методів розробки

## 1.1 Методи розробки програми та користувальницької бібліотеки функцій

У роботі використовувалися різні методи розробки програмного коду для вирішення поставленої задачі. Загалом я використовував стандартні бібліотеки, а також створив свою бібліотеку функцій.

iostream – бібліотека для введення та виведення даних [1], а також для роботи з файловими потоками. Вона підключається як до головного файлу, так і до файлу бібліотеки функцій.

string – бібліотека функцій та методів для опрацювання рядків класу даних string [2]. Підключається ця бібліотека до файлу бібліотеки функцій.

map – бібліотека для роботи з текстовими даними, яка дозволяє використовувати карту або словник зі значенням і ключем. Вона має декілька класів і методів, а також автоматичне сортування за ключами [3]. Використовуватиметься вона у файлі бібліотеки функцій.

fstream – стандартна бібліотека для потоків файлів, введення та виведення даних [4]. Бібліотека використовувалася для функції з альтернативним підходом для розв’язання задачі з текстовими даними класу string. Підключатиметься вона до бібліотеки функцій.

Бібліотека функцій, що були написанні по завданню цієї роботи і створена, як користувацька – MyFunctions.h. Усі функції бібліотеки використані у головному модулі коду.

Здебільшого в програмі використовуватимуться методи опрацювання текстових даних через стандартну бібліотеку у стилі С. Цей стиль полягає у використанні масивів типу char із записуванням по одному елементу рядка і опрацювання масиву як рядка. Методи є доволі надійними, хоча і складнішими в реалізації. У функції PrintRowByText() було використано стиль опрацювання С++, що полягає у використанні методів і класів string, а також використання інших функцій для запису та зчитування даних у файли. У завданні з двома зірочками (високий рівень складності) використовуватиметься динамічна пам’ять, робота з нею, її виділення та її звільнення.

Код має кодування символів сторінки 1251, що підходить для застосування українського алфавіту. Метод кодування виставляється за допомогою функції chcp [5] та у зв’язці з функцією system [6]. Суть полягає у тому, що без використання такого кодування, програма не змогла б виводити інформацію та коректно опрацьовувати текстові дані.

## 1.2 Середовище розробки

Як середовище розробки в цій роботи використовуватиметься IDE Visual Studio Community. Visual Studio — серія продуктів фірми Майкрософт, які містять інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів [7]. Ці продукти дають змогу розробляти як консольні програми, так і програми з графічним інтерфейсом.

Visual Studio має такі компоненти Visual Basic .NET, Visual C++, Visual C#, Visual F#, Visual Studio Debugger, Microsoft SQL Server, JavaScript, Python, TypeScript, XAML [8]. Це середовище розробки має численні зручні функції, що полегшують роботу програмістам, а також покращують написання коду і його дебагінг особливо. Це середовище розробки стало популярним через свої функції та можливості, а також через надання безкоштовної версії.

# 2 Словесний опис алгоритмів для розв’язання задач

## 2.1 Опис функції main()

Функція main() є головною функцією, в якій викликаються всі функція для опрацювання файлів *F1*.txt, *F2*.txt, *F3*.txt та *F4*.txt.

Функція має тип int, тобто повертає числове значення після закінчення роботи програми або у разі появи будь-якої помилки. Функція не має жодного аргументу.

Підключення бібліотек для використання:

#include <iostream>

#include "MyFunctions.h"

Використання простору імен std:

using namespace std;

Оголошення змінних:

n – змінна типу int для кількості рядків матриці

m – змінна типу int для кількості стовпців матриці

name – змінна типу char[] для імені першого фалу

name1 – змінна типу char[] для імені другого фалу

name2 – змінна типу char[] для імені третього фалу

name3 – змінна типу char[] для імені четвертого фалу

Виклик функції, що виводить в консоль вміст файлу:

view\_file(name);

Виклик функції з аргументами name – ім’я файлу, n – кількість рядків, m – кількість рядків. Обчислює кількість рядків та стовпців для матриці та повертає їх, як вказівники:

array\_RowsColumns(name, n, m);

Створення динамічного двомірного масиву для матриці з файлу.

Цикл, що створює динамічний масив в кожній ітерації циклу і записує вказівник на перший елемент в i-тий елемент масиву matrix.

int\*\* matrix = new int\* [n];

for (int i = 0; i < n; i++)

matrix[i] = new int[m];

Виклик функції з аргументами name – ім’я файлу, та matrix – вказівник на перший елемент динамічного масиву. Функція записує дані з файлу у числову матрицю.

make\_array(name, matrix);

Виклик функції з аргументами name1 – ім’я файлу для запису, matrix – вказівник на перший елемент динамічного масиву, n та m – розмірність матриці. Функція виводить матрицю в консоль.

make\_output3(name1, matrix, n, m);

Оголошення змінної типу int, значення якої це значення виклику функції з аргументами name1 – ім’я файлу для запису, matrix – вказівник на перший елемент динамічного масиву, n та m – розмірність матриці. Функція повертає значення кількості елементів більших за 1 в матриці.

int AmountOfBiggerThanOne = BiggerThanOne(matrix, name1, n, m);

Виведення в консоль кількості елементів більших за один з символами переходу на інший рядок перед та після виведення.

cout << "\n\nКількість елементів більших за 1: " << AmountOfBiggerThanOne << "\n";

Оголошення змінної типу int, значення якої це значення виклику функції з аргументами name1 – ім’я файлу для запису, matrix – вказівник на перший елемент динамічного масиву, n та m – розмірність матриці. Функція повертає чисельне значення кількості локальних мінімумів в матриці.

int LocalMinimumAmount = LocalMinimum(name1, matrix, n, m);

Виведення в консоль кількості елементів, що є локальними мінімумами з символами переходу на інший рядок перед та після виведення.

cout << "\nКількість локальних мінімумів по вертикалі та горизонталі матриці: " << LocalMinimumAmount << "\n";

Створення динамічного масиву цілочислового типу з розмірністю в стовпці матриці.

int\* vector = new int[m];

Виклик функції з аргументами name1 – ім’я файлу для запису, matrix – вказівник на перший елемент динамічного масиву, vector – вказівник на перший елемент динамічного масиву, n та m – розмірність матриці. Функція створення вектору з сум негативних чисел стовпців матриці.

VectorNegativeNumbersColumns(name1, matrix, n, m, vector);

Видалення динамічної пам’яті, що виділили під динамічний масив vector.

delete[] vector;

Виклик функції з аргументами name1 – ім’я файлу для запису, matrix – вказівник на перший елемент динамічного масиву, n та m – розмірність матриці. Функція створює масив з ненульовими елементами матриці.

DynamicArrayNonZero(name1, matrix, n, m);

Виклик функції з аргументами name1 – ім’я файлу для запису, matrix – вказівник на перший елемент динамічного масиву, n та m – розмірність матриці. Функція сортує матрицю за квадратами суми рядків.

SortBySumSquares(name1, matrix, n, m);

cout << "\n\n";

Виклик функції перегляду змісту файлу з аргументом name2 – ім’я файлу для виведення.

view\_file(name2);

cout << "\n\n";

Виклик функції з аргументами name2 і name3 – ім’я файлу для перегляду і ім’я файлу для запису. Функція шукає цифри у файлі і записує їх в другий файл.

NumsInRows(name2, name3);

Виклик функції з аргументами name2 і name3 – ім’я файлу для перегляду і ім’я файлу для запису. Функція сортує всі слова в кожному рядку за алфавітом.

SortByAlphabet(name2, name3);

Виклик функції перегляду змісту файлу з аргументом name3 – ім’я файлу для виведення.

view\_file(name3);

Виклик функції з аргументом name2 – ім’я файлу для читання. Функція замінює рядок за введеним номер на введений текст.

PrintRowByNum(name2);

view\_file(name2);

Виклик функції з аргументом name2 – ім’я файлу для читання. Функція шукає введений текст у файлі і замінює його на введений текст.

PrintRowByText(name2);

view\_file(name2);

Виклик функції з аргументом name2 – ім’я файлу для читання. Функція видаляє рядок за введеним номером.

DeleteRowByNum(name2);

view\_file(name2);

Виклик функції з аргументом name2 – ім’я файлу для читання. Функція видаляє рядок якщо в ньому є введений текст

DeleteRowByText(name2);

view\_file(name2);

cout << "\n\n";

Блок-схема цієї функції наведена у додатку А.1

## 2.2 Опис функції view\_file()

Функція для виведення вмісту файлу має прототип:

void view\_file(char\* name);

Оголошення тимчасової змінної для зчитування рядків.

char s[100];

Оголошення змінної вказівника для файлу.

FILE\* f;

Відкривання текстового файлу у режимі читання та перевірка наявності файлу.

fopen\_s(&f, name, "rt");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "\nПерегляд файлу " << name << "\n----------------\n";

Цикл, що виводить зміст файлу в консоль по рядку за кожну ітерацію. Функція fgets() зчитує по 100 байтів у змінну s кожну ітерацію, а при закінчення файлу виводить значення, що завершує дію циклу.

while (fgets(s, 100, f))

{

cout << s;

}

cout << "\n-------------------------------\n";

Закриваємо файл, що відкрили на початку програми.

fclose(f);

}

Блок-схема цієї функції наведена у додатку А.2

## 2.3 Опис функції array\_RowsColumns()

Аргументами функції є ім’я файлу та змінні для рядків та стовпчиків.

Функція, що обчислює рядки та стовпчики у файлі F1 та передає їх за адресом.

void array\_RowsColumns(char\* name, int& n, int& m)

{

n = 0;

Оголошення змінної для читання рядків.

char s[100] = { '0' };

FILE\* f;

Відкриття текстового файлу у режимі читання.

fopen\_s(&f, name, "rt");

Перевірка файлу на існування.

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

Порядкове зчитування даних з фалу.

while (fgets(s, 100, f))

{

Обчислення довжини масиву з рядком.

int len = strlen(s);

m = 0;

Цикл, що ітерує по масиву з рядком та записує кількість чисел у ньому у змінну з стовпчиками, а кількість ітерації записує в змінну рядків.

for (int i = 0; s[i] != '\0'; i++)

{

Перевірка таких умов: елемент не останній у масиві, елемент є цифрою, наступний елемент не є цифрою.

if ((i + 1) != len && isdigit(s[i]) && !isdigit(s[i + 1]))

m++;

Перевірка таких умов: елемент є останнім у масиві, елемент є цифрою.

else if ((i + 1) >= len && isdigit(s[i]))

m++;

}

n++;

}

Закриваємо файл.

fclose(f);

}

Блок-схема цієї функції наведена у додатку А.3

## 2.4 Опис функції make\_array

Аргументи функції: ім’я фалу та вказівник на динамічну матрицю.

Функція, що створює динамічний двомірний масив з показників вимірювань у файлі F1.

void make\_array(char\* name, int\*\* matrix)

{

int row = 0;

char s[100] = { '0' };

FILE\* f;

Відкриваємо файл і перевіряємо його на існування.

fopen\_s(&f, name, "rt");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

Цикл для читання даних файлу по рядках

while (fgets(s, 100, f))

{

int column = 0;

Обчислюємо довжину тимчасового масиву рядка.

int len = strlen(s);

Цикл, що ітерує по рядку.

for (int i = 0; i < len; i++)

{

Перевірка умови чи елемент не є пробілом, кінцем масиву, символом наст. рядка або табуляції.

if (s[i] != ' ' && s[i] != '\0' && s[i] != '\n' && s[i] != '\t')

{

Оголошення тимчасового масиву для числа.

char number[10] = { '\0' };

int j;

Цикл, що записує все що йде після першого числа в масив number, якщо це також є число.

for (j = 0; s[i] != ' '; j++)

{

if (s[i] != '\n' && s[i] != '\0' && s[i] != '\t')

number[j] = s[i++];

else break;

}

Масив з числом перетворюється на число завдяки функції atoi() та записується у відповідний рядок і стовпчик. Стовпчик одразу збільшується для наступного числа.

matrix[row][column++] = atoi(number);

}

}

Збільшення числа для рядка після кожного прочитаного рядка у файлі.

row++;

}

Закриття файлу.

fclose(f);

}

Блок-схема цієї функції наведена у додатку А.4

## 2.5 Опис функції make\_output3()

Аргументи функції: ім’я файлу для записування, матриця та кількість рядків, стовпців матриці. Функція, що виводить створену матрицю в консоль, а також записує в файл F2.

void make\_output3(char\* name1, int\*\* matrix, int& n, int& m)

{

FILE\* f;

Відкриття текстового файлу для записування даних та перевірка на його існування.

fopen\_s(&f, name1, "wt");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "Перегляд матриці:\n";

Вкладений цикл для записування матриці до файлу з виводом до консолі.

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

Функція fprintf() записує дані до файлу, який йде до неї як перший аргумент.

fprintf(f, "%i\t\t", matrix[i][j]);

cout << matrix[i][j] << "\t\t";

}

fprintf(f, "\n");

cout << "\n";

}

Закриваємо файл.

fclose(f);

}

Блок-схема цієї функції наведена у додатку А.5

## 2.6 Опис функції BiggerThanOne()

Аргументи функції: ім’я файлу для записування, матриця та її розмірність у двох змінних. Також ця функція повертає тип даних int. Функція, що обчислює кількість елементів більших за 1 в матриці та записує в файл F2.

int BiggerThanOne(int\*\* matrix, char\* name1, int& n, int& m)

{

int amount = 0;

FILE\* f;

Відкриваємо текстовий файл для дописування даних та перевіряємо його існування.

fopen\_s(&f, name1, "at");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return -1;

}

fprintf(f, "\nКількість показників вимірювань датчиків (елементів матриці), більших за 1: ");

Вкладений цикл, що рахує кількість елементів більших за один в матриці ітерючи по кожному елементу.

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

{

if (matrix[i][j] > 1)

amount++;

}

Записуємо результат у файл, закриваємо файл та повертаємо значення функції.

fprintf(f, "%i\n", amount);

fclose(f);

return amount;

}

Блок-схема цієї функції наведена у додатку А.6

## 2.7 Опис функції LocalMinimum()

Аргументи функції: ім’я файлу для записування, матриця та її розмірність у двох змінних. Також ця функція повертає тип даних int. Функція повертає кількість локальних мінімумів по горизонталі та вертикалі та записує у файл F2.

int LocalMinimum(char\* name1, int\*\* matrix, int& n, int& m)

{

int amount = 0;

FILE\* f;

Відкриваємо текстовий файл для дописування даних та перевіряємо його існування.

fopen\_s(&f, name1, "at");

if (f == NULL) {

cout << "Cannot open file to view\n";

return -1;

}

fprintf(f, "\nКількість локальних мінімумів: ");

Оголошення масивів для перевірки умов локального мінімуму.

int dx[] = { 0, 1, 0, -1 };

int dy[] = { 1, 0, -1, 0 };

Вкладений цикл для ітерації по кожному елементу матриці.

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

Оголошення булевої змінної для перевірки мінімальності елемента.

bool is\_min = true;

Цикл, що перевіряє один елемент порівнюючи його з кожним сусіднім.

for (int k = 0; k < 4; k++) {

int ni = i + dx[k];

int nj = j + dy[k];

Перевірка умови чи сусідній елемент не виходить за межі масиву

if (ni >= 0 && ni < n && nj >= 0 && nj < m) {

Перевірка умови чи елемент є менший за сусідній елемент. Змінна для перевірки мінімальності набуває значення false у випадку виконання умови.

if (matrix[ni][nj] <= matrix[i][j]) {

is\_min = false;

break;

}

}

}

Перевірка умови чи є змінна is\_min true. У випадку виконання умови кількість локальних мінімумів збільшується на один.

if (is\_min) {

amount++;

}

}

}

Записуємо результат у файл, закриваємо файл та повертаємо значення функції.

fprintf(f, "%i\n", amount);

fclose(f);

return amount;

}

Блок-схема цієї функції наведена у додатку А.7

## 2.8 Опис функції VectorNegativeNumbersColumns()

Аргументи функції: ім’я файлу для записування, матриця, її розмірність у двох змінних та вектор. Функція для обчислення вектору за сумою від'ємних елементів стовпців матриці та записує результат у файл F2.

void VectorNegativeNumbersColumns(char\* name1, int\*\* matrix, int& n, int& m, int\* vector)

{

int sum = 0;;

FILE\* f;

Відкриваємо текстовий файл для дописування даних та перевіряємо його існування.

fopen\_s(&f, name1, "at");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "\nВектор суми від'ємних елементів стовпців матриці:\n";

fprintf(f, "\nВектор суми від'ємних елементів стовпців матриці:\n");

Ітеруємо по елементам стовпців матриці, якщо елемент менший за нуль додаємо його до тимчасової змінної суми.

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

if (matrix[j][i] < 0)

sum += matrix[j][i];

}

Записуємо суму елементів до вектору та записуємо 0 до неї.

vector[i] = sum;

sum = 0;

}

Записуємо вектор у файл за допомогою циклу, закриваємо файл.

for (int i = 0; i < m; i++)

{

fprintf(f, "%i\t", vector[i]);

cout << vector[i] << '\t';

}

cout << '\n';

fclose(f);

}

Блок-схема цієї функції наведена у додатку А.8

## 2.9 Опис функції DynamicArrayNonZero()

Аргументи функції: ім’я файлу для записування, матриця, її розмірність у двох змінних. Функція створює динамічний масив з ненульових елементів матриці та записує результат у файл F2.

void DynamicArrayNonZero(char\* name1, int\*\* matrix, int& n, int& m)

{

int index = 0;

FILE\* f;

Відкриваємо текстовий файл для дописування даних та перевіряємо його існування.

fopen\_s(&f, name1, "at");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "\nДинамічний масив ненульових показників датчиків:\n";

fprintf(f, "\n\nДинамічний масив ненульових показників датчиків:\n");

Вкладений цикл, що обчислює кількість ненульових елементів в матриці.

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

if (matrix[i][j] != 0)

index++;

Створення динамічного одновимірного масиву.

int\* vector = new int[index];

int k = 0;

Вкладений цикл, що записує всі ненульові елементи матриці у вектор.

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

if (matrix[i][j] != 0)

{

vector[k] = matrix[i][j];

Збільшуємо індекс елементу вектору.

k++;

}

Цикл для записування даних у файл та виведення в консоль.

for (int i = 0; i < index; i++)

{

fprintf(f, "%i ", vector[i]);

cout << vector[i] << " ";

}

cout << '\n';

Закриваємо файл.

fclose(f);

}

Блок-схема цієї функції наведена у додатку А.9

## 2.10 Опис функції SortBySumSquares()

Аргументи функції: ім’я файлу для записування, матриця, її розмірність у двох змінних. Функція, що розташовує рядки матриці сортуючи за квадратом суми та записує результат у файл F2.

void SortBySumSquares(char\* name1, int\*\* matrix, int& n, int& m)

{

int sum = 0;

Створення map з ключем та значенням для записування квадратів сум елементів та номер рядка.

map < int, int> sums;

Створення ітератора для map sums.

map <int, int> ::iterator it;

FILE\* f;

Відкриваємо текстовий файл для дописування даних та перевіряємо його існування.

fopen\_s(&f, name1, "at");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "\nВідсортована матриця за сумою квадратів рядків(показників датчиків):\n";

fprintf(f, "\n\nВідсортована матриця за сумою квадратів рядків(показників датчиків):\n");

Вкладені цикли для ітерації по кожному рядку та обчислення їх квадратів суми. Після обчислення квадрати записуються у значення ключів sums, а номер рядка записується до значення.

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

sum += matrix[i][j];

}

Обчислення квадрату суми.

sum \*= sum;

sums[sum] = i;

sum = 0;

}

Ітератор ставиться на початкову позицію словника sums.

it = sums.begin();

Цикл, що записує у файл та виводить в консоль рядки за сортуванням у словнику sums.

for (int i = 0; i < n; i++)

{

Змінна для номеру рядка.

int j = it->second;

for (int k = 0; k < m; k++)

{

cout << matrix[j][k] << "\t\t";

fprintf(f, "%i\t", matrix[j][k]);

}

fprintf(f, "\n");

cout << "\n";

Збільшення ітератора на один – наступний елемент словника.

it++;

}

Закриваємо файл.

fclose(f);

}

Блок-схема цієї функції наведена у додатку А.10

## 2.11 Опис функції NumsInRows()

Аргументи функції: ім’я файлу для читання, ім’я фалу для записування. Функція визначає, чи є у рядках цифри файлу F3, виводить їх та записує результат у файл F4.

void NumsInRows(char\* name2, char\* name3)

{

char s[200] = { '\0' };

int k = 0;

FILE\* f1, \* f2;

Відкриваємо текстовий файл для читання даних та записування даних і перевіряємо їх існування.

fopen\_s(&f1, name2, "r");

fopen\_s(&f2, name3, "w");

if (f1 == NULL || f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "Цифри у файлі " << name2 << " -> ";

fprintf(f2, "Цифри у файлі %s -> ", name2);

Цикл для читання файлу по рядкам.

while (fgets(s, 200, f1))

{

Цикл, що ітерує по масиву до символу закінчення.

for (int i = 0; s[i] != '\0'; i++)

{

Перевірка умови чи є елемент цифрою.

if (s[i] >= '0' && s[i] <= '9')

{

Якщо елемент є цифрою він записується у файл.

fprintf(f2, "%c ", s[i]);

k++;

}

}

}

Перевірка умови, чи є k нульовою змінною.

if (k == 0)

{

Записування у файл тексту у випадку нульового k.

cout << "Немає жодної цифри у файлі" << "\n\n";

fprintf(f2, "Немає жодної цифри у файлі\n");

}

else

fprintf(f2, "\n");

Закриваємо два файли.

fclose(f1);

fclose(f2);

}

Блок-схема цієї функції наведена у додатку А.11

## 2.12 Опис функції SortByAlphabet()

Аргументи функції: ім’я файлу для читання, ім’я фалу для записування. Функція, що сортує слова за алфавітом в кожному рядку та записує результат у файл F4.

void SortByAlphabet(char\* name2, char\* name3)

{

char s[200] = { '\0' };

char tmp[100] = { '\0' };

Створення map та ітератора для записування слів у ключі та їх автоматичне сортування за алфавітом

map <string, int> words;

map <string, int> ::iterator it;

FILE\* f1, \* f2;

Відкриваємо текстовий файл для читання даних та записування даних і перевіряємо їх існування.

fopen\_s(&f1, name2, "r");

fopen\_s(&f2, name3, "a");

if (f1 == NULL || f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

fprintf(f2, "\nВідсортовані рядки за алфавітом:\n\n");

Цикл, що читає рядки з файлу f1.

while (fgets(s, 200, f1))

{

Скидання до 0 тимчасової змінної для рядку.

memset(tmp, 0, 100);

int amount = 0;

int size = strlen(s);

Цикл, що ітерує по масиву char-ів з рядком.

for (int i = 0; i < size; i++)

{

Перевірка умови чи є елемент буквою алфавіту.

if (s[i] >= 'а' && s[i] <= 'я' || s[i] >= 'А' && s[i] <= 'Я' || s[i] == 'і' || s[i] == 'І' || s[i] == 'ї' || s[i] == 'Ї' || s[i] == 'є' || s[i] == 'Є' || s[i] == 'ґ' || s[i] == 'Ґ')

{

int k;

Цикл для записування слова у тимчасову змінну.

for (k = 0; s[i] != ' '; k++)

{

Умова, що перевіряє чи не є змінна символами, що не є буквами. У випадку виконання умови елемент записується у тимчасовий маси і збільшується індекс елементу і.

if (s[i] != '\0' && s[i] != '\n' && s[i] != '.' && s[i] != ',' && s[i] != '!' && s[i] != '?' && s[i] != ';' && s[i] != ':')

tmp[k] = s[i++];

else

{

break;

}

}

Якщо індекс тимчасової змінної для слова не є більше за 0, останній елемент буде символом закінчення масиву.

if (k > 0)

{

tmp[k] = '\0';

}

У словник words у ключ записується слово з тимчасової змінної, і значення збільшується на один.

words[tmp]++;

}

}

Зміна, що приймає значення кількості елементів в словнику words.

amount = words.size();

Ітератор переміщується на початок словника.

it = words.begin();

Цикл, що проходить повністю по словнику відсортованих слів.

for (int i = 0; i < amount; i++)

{

Тимчасова змінна приймає значення елемента словника.

int tmp = it->second;

char word[50] = { '\0' };

Цикл, що копіює слово з словнику до тимчасової змінної word.

for (int j = 0; it->first[j] != '\0'; j++)

word[j] = it->first[j];

Цикл, що записує слова з словнику в відсортованому порядку до файлу. У випадку повторювання слова, воно буде записано стільки раз, скільки було в початковому файлі.

for (int j = 0; j < tmp; j++)

{

fprintf(f2, "%s ", word);

}

Ітератор збільшується на один для переходу до наступного елементу.

it++;

Скидання тимчасового масиву до 0.

memset(word, 0, 50);

}

Скидання повністю словника до 0 елементів.

words.clear();

fprintf(f2, "\n");

}

cout << "\n";

Закриття файлів.

fclose(f1);

fclose(f2);

}

Блок-схема цієї функції наведена у додатку А.12

## 2.13 Опис функції PrintRowByNum()

Аргументи функції: ім’я файлу для редагування. Функція, що замінює введений рядок на введений текст у файлі F3.

void PrintRowByNum(char\* name2)

{

int InputRow;

char InputText[200];

cout << "\n\nВведіть номер рядка для заміни: ";

cin >> InputRow;

cout << "\nВведіть текст для заміни -> ";

Рядок для ігнорування залишку від cin у консолі.

cin.ignore();

Введення рядка за допомогою функції getline в масив чарів.

cin.getline(InputText, 200);

Створення map та ітератора для записування речень у значення та їх номери у ключі.

map <int, char[200]> words;

map <int, char[200]> ::iterator it;

char s[200];

FILE\* f1, \* f2;

Відкриваємо файл у режимі читання і перевіряємо існування.

fopen\_s(&f1, name2, "r");

if (f1 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

int k = 1;

Копіюємо всі рядки у словник з ключами, що нумеруються за порядком записування.

while (fgets(s, 200, f1))

{

strcpy\_s(words[k], 200, s);

k++;

}

Закриваємо файл.

fclose(f1);

Відкриваємо файл у режимі записування даних.

fopen\_s(&f2, name2, "w");

Перевіряємо на існування файлу.

if (f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

Ітератор ставиться на початок словника.

it = words.begin();

Цикл для ітерації по словнику.

for (int i = 0; i < k - 1; i++)

{

Перевірка умови чи ключ дорівнює рядку, який треба видалити. У випадку виконання умови рядок замінюється і записується у файл.

if (it->first == InputRow)

{

fprintf(f2, "%s\n", InputText);

it++;

}

else

{

Якщо ключ не дорівнює рядку змінення, рядок записується у файл незмінно.

fprintf(f2, "%s", it->second);

it++;

}

}

Закриваємо файл.

fclose(f2);

}

Блок-схема цієї функції наведена у додатку А.13.

## 2.14 Опис функції PrintRowByText()

Аргументи функції: ім’я файлу для редагування. Функція, що шукає введений текст у рядку і замінює іншим введеним текстом у файлі F3.

void PrintRowByText(char\* name2)

{

char name[] = "F3.txt";

system("chcp 1251 > null");

Використання функції для виведення змісту файлу з бібліотеки <fstream>.

ifstream f1(name);

Оголошення текстових типів даних для шуканого тексту, тексту заміни, та для тимчасових рядків.

string toFind;

string change;

string s;

Використання словника для рядків слів.

map <int, string> words;

map <int, string> ::iterator it;

int k = 0;

Цикл, що копіює рядки у словник у поле для значень використовуючи функцію getline з двома аргументами.

while (getline(f1, s))

{

words[k] = s;

k++;

}

Вводимо тест для пошуку та для заміни.

cout << "\n\nВведіть текст для пошуку у файлі: ";

getline(cin, toFind);

cout << "\nВведіть текст для заміни -> ";

getline(cin, change);

int len = toFind.length();

Булева змінна для позначення знаходження слова у текстовій змінній.

bool x;

Цикл, що ітерує по рядкам в словнику.

for (int i = 0; i < k; i++)

{

Присвоєння значення true до змінної х.

x = true;

За умови х є дійсною виконується цикл

while (x)

{

Функція пошуку тексту у файлі з бібліотеки <string>. Вона повертає індекс першого входження до тексту або -1 у разі його відсутності.

int index = words[i].find(toFind);

Якщо в рядку немає шуканого тексту х стає хибою і оператор break виходить з циклу while.

if (index == -1)

{

x = false;

break;

}

Якщо в рядку є шуканий тест, він видаляється за допомогою метода erase, а після методом insert вставляється з того ж індексу новий текст.

words[i].erase(index, len);

words[i].insert(index, change);

}

}

Закриваємо файл для читання.

f1.close();

Відкриваємо файл для записування даних.

ofstream f2(name);

Записуємо змінені рядки словника по черзі.

for (int i = 0; i < k; i++)

{

f2 << words[i];

f2 << "\n";

}

Закриваємо файл.

f2.close();

}

Блок-схема цієї функції наведена у додатку А.14

## 2.15 Опис функції DeleteRowByNum()

Аргументи функції: ім’я фалу для редагування. Функція, що видаляє введений рядок у файлі F3.

void DeleteRowByNum(char\* name2)

{

int InputRow;

cout << "\n\nВведіть номер рядка для видалення: ";

cin >> InputRow;

Створення map для записування рядків у значення та їх номери у ключі.

map <int, char[200]> words;

map <int, char[200]> ::iterator it;

char s[200];

FILE\* f1, \* f2;

Відкриваємо файл для читання та перевіряємо на існування.

fopen\_s(&f1, name2, "r");

if (f1 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

int k = 1;

Копіюємо кожен ряжок до словника за допомогою циклу.

while (fgets(s, 200, f1))

{

Функція strcpy\_s, шо приймає 3 аргументи і копіює з 3 аргументу в перший кількість елементів позначених у 2 аргументі.

strcpy\_s(words[k], 200, s);

k++;

}

Закриваємо файл.

fclose(f1);

Відкриваємо файл для записування.

fopen\_s(&f2, name2, "w");

Перевіряємо на існування.

if (f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

Ітератор ставиться на початок словника.

it = words.begin();

Цикл для ітерування словником.

for (int i = 0; i < k - 1; i++)

{

За умови, що ключ елемента словника є введеним номером рядка, ітератор збільшується на один.

if (it->first == InputRow)

{

it++;

}

else

{

Якщо умова не є істинною виконується записування рядка до файлу.

fprintf(f2, "%s", it->second);

it++;

}

}

Закриваємо файл.

fclose(f2);

}

Блок-схема цієї функції наведена у додатку А.15

## 2.16 Опис функції DeleteRowByText()

Аргументи функції: ім’я файлу для редагування. Функція, що видаляє рядок з шуканим текстом у файлі F3.

void DeleteRowByText(char\* name2)

{

char SearchText[200];

cout << "\n\nВведіть текс для пошуку у файлі: ";

Ігнорування першого символу в консолі.

cin.ignore();

cin.getline(SearchText, 200);

Записування першої літери у окрему змінну.

char firstlet = SearchText[0];

Створення map для записування речень у значення та їх номери у ключі

map <int, char[200]> words;

map <int, char[200]> ::iterator it;

char s[200];

FILE\* f1, \* f2;

Відкриваємо файл для читання та перевіряємо існування.

fopen\_s(&f1, name2, "r");

if (f1 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

int k = 1;

Копіюємо повністю рядки до словника записуючи у значення рядок, а у ключі номера.

while (fgets(s, 200, f1))

{

strcpy\_s(words[k], 200, s);

k++;

}

Закриваємо файл.

fclose(f1);

Відкриваємо файл для записування та перевіряємо існування.

fopen\_s(&f2, name2, "w");

if (f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

Ітератор на початок словника.

it = words.begin();ї

Булева зміна для позначення шуканого слова.

bool compared = true;

Цикл для ітерування по словнику.

for (int i = 0; i < k - 1; i++)

{

Цикл для ітерування по рядку.

for (int j = 0; j < strlen(it->second); j++)

{

Умова перевірки чи є елемент першою літерою шуканого тексту у рядку.

if (it->second[j] == firstlet)

{

int n = j;

Змінна стає істинною.

compared = true;

Цикл для перевірки слова після першої літери на подібність.

for (int t = 0; t < strlen(SearchText); t++)

{

Якщо хоч один символ не співпадає, змінна набуває хибного значення і цикл завершується.

if (it->second[n] != SearchText[t])

{

compared = false;

break;

}

Збільшується індекс символу рядка.

n++;

}

Якщо текст був порівняний вірно, рядок скидається.

if (compared)

{

it->second[0] = '\0';

}

}

}

Рядок записується у файл та збільшується ітератор на один.

fprintf(f2, "%s", it->second);

it++;

}

Закриваємо файл.

fclose(f2);

}

Блок-схема цієї функції наведена у додатку А.16

# 3 Результати виконання програми

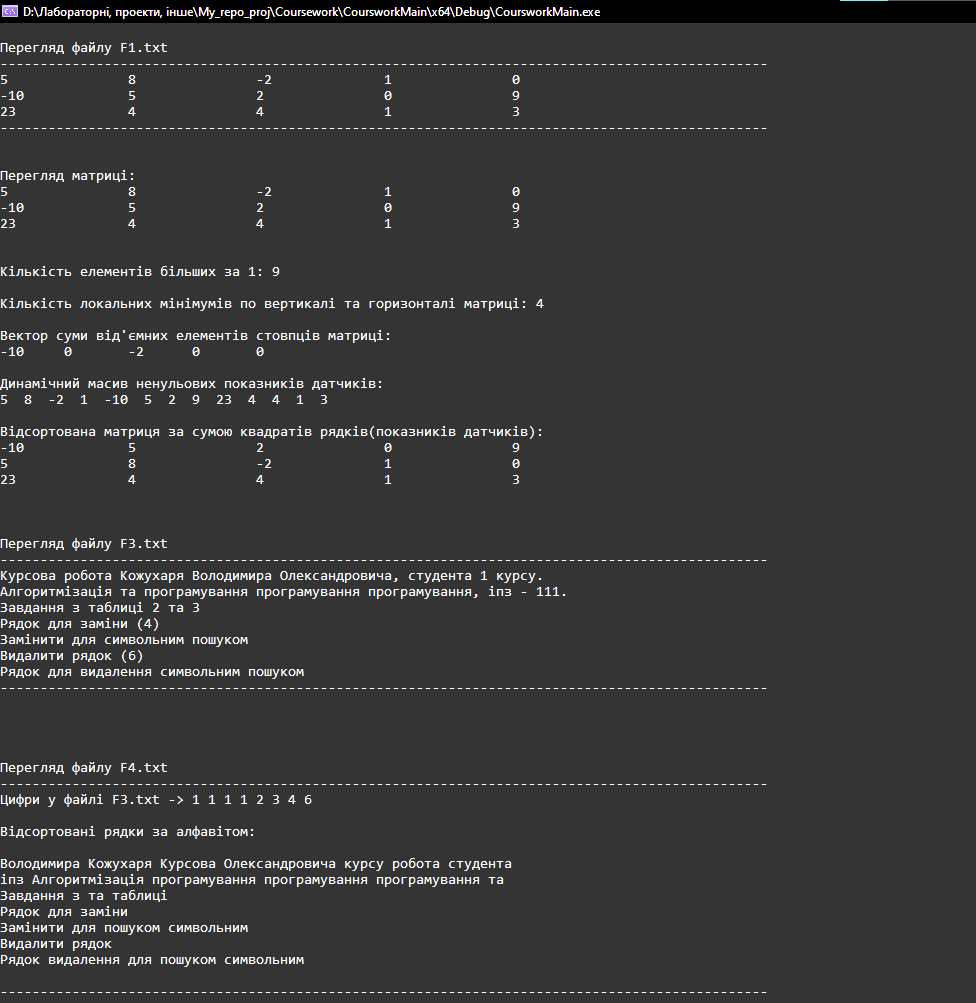


Рисунок 3.1 – Скріншот консольного вікна(перший) з результатами роботи проєкту

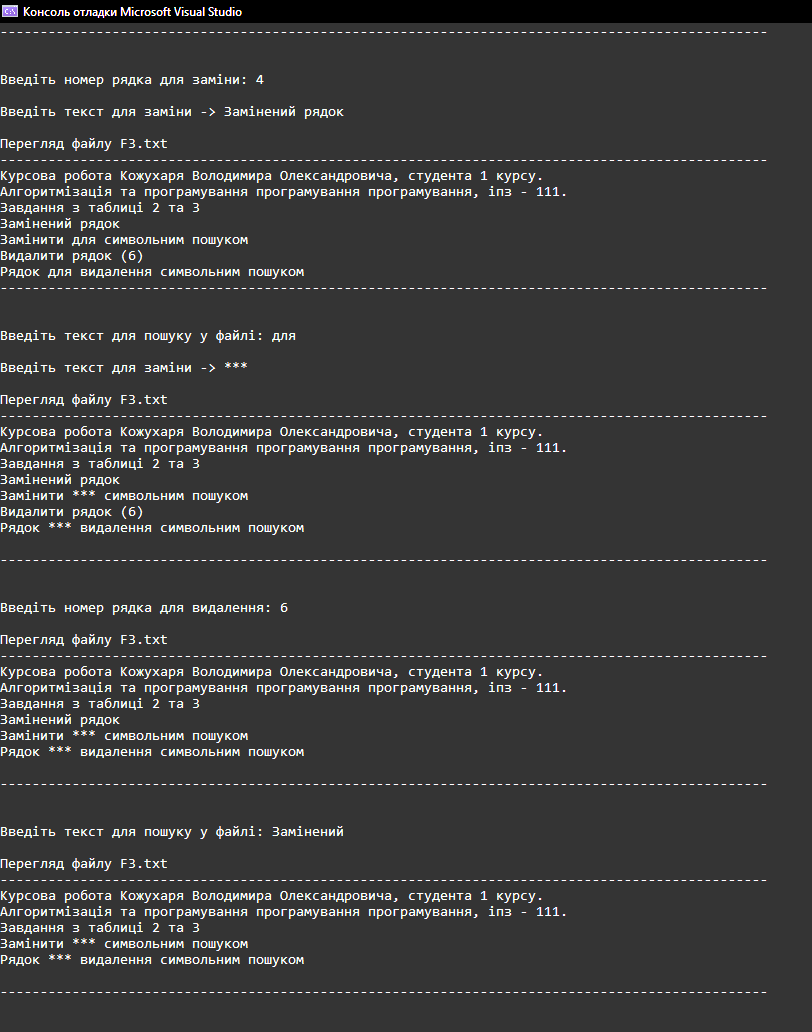


Рисунок 3.2 – Скріншот консольного вікна(другий) з результатами роботи проєкту

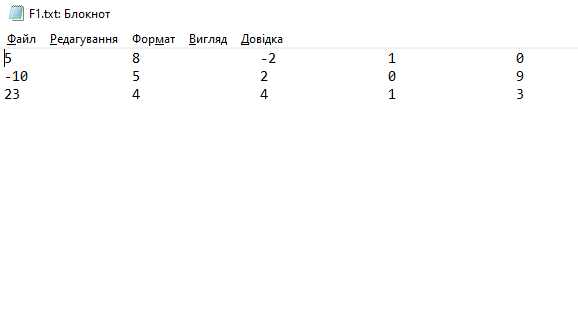


Рисунок 3.3 – Вигляд текстового файлу F1 з вхідною матрицею числових показників вимірювань

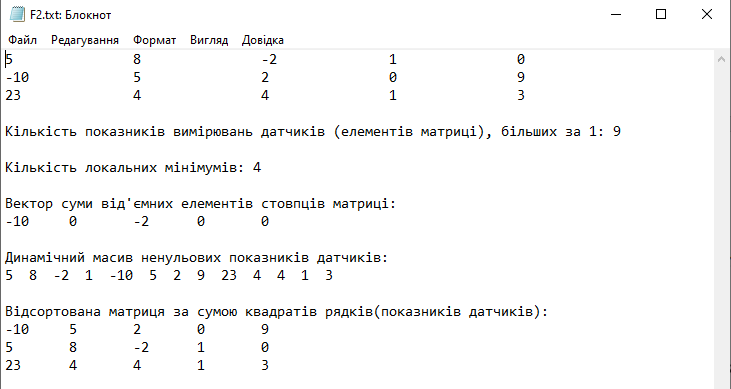


Рисунок 3.4 – Вигляд текстового файлу F2 з результатами опрацювання даних матриці

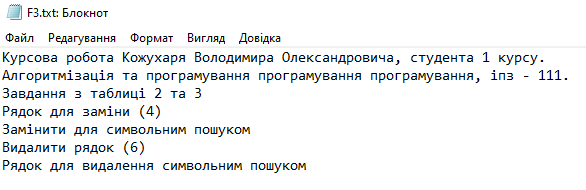


Рисунок 3.5 – Вигляд текстового файлу F3 з текстовими даними для опрацювання

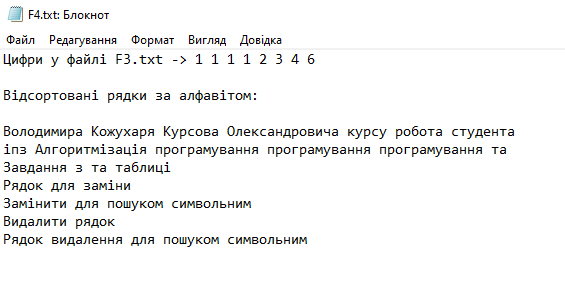


Рисунок 3.6 – Вигляд текстового файлу F4 з результатами опрацювання файлу F3

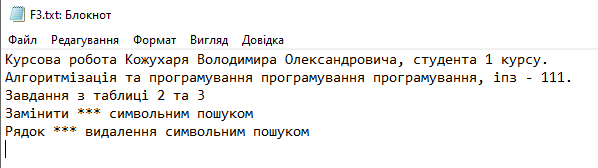


Рисунок 3.7 – Вигляд текстового файлу F3 після виконання програми

# 4 Інструкція з використання програми

Для використання цієї програми знадобиться середовище розробки Visual Studio Community [7] з встановленим розширенням для С++. До цієї програми входить статична бібліотека виконуваних функції з назвою StaticLibForCoursework, також входить головний файл проєкту CourseworkMain.

По-перше, до головного рішення проєкту, яке в свою чергу можна відкрити файлом з розширенням .sln, треба додати статичну бібліотеку натиснувши правою кнопкою по проєкту і функція додати->існуючий елемент. Вибрати папку бібліотеки і вона автоматично доєднається до рішення.

По-друге, у посиланнях проєкту треба додати посилання на бібліотеку, натиснувши додати посилання і вибрати StaticLibForCoursework. Потім натиснути властивості головного проєкта, на вкладці С/С++, у додаткових каталогах підключених файлів вказуємо шлях до папки з статичною бібліотекою.

По-третє, назначаємо файл головного проєкту як файл, що запускається. Після всіх дій необхідно додати два файли з даними до папки головної функції для їх опрацювання (“F1.txt” та “F3.txt”). Щоб запустити програму, треба відкрити файл CoursworkMain.cpp і запустити виконання програми комбінацією клавіш Ctrl + F5.

Після запуску програми перша частина виведеться в консоль з опрацюванням матриці даних датчиків. Після опрацювання матриці у консолі потрібно буде ввести дані, для продовження виконання програми, 4 рази. Після того як програма завершиться, нажміть будь-яку клавішу для закриття консолі. Під час виконання програми створяться два нових файлу з вихідними даними. Усі файли можна переглядати через будь-який текстовий редактор.

# Висновки з аналізом досягнутих результатів

Досягнуті результати цієї роботи:

* перетворення текстової матриці в числову;
* опрацювання числової матриці з різними умовами;
* опрацювання текстових даних в програмні рядки.
* пошук слів або тексту в рядку файлу.
* заміна тексту в рядку файлу.
* видалення рядків з файлу.

У роботі виконані всі завдання з різними рівнями складності, всі результати були досягнуті декількома методами опрацювання даних. При розробці цієї програми було використано мову С++, 16 різних алгоритмів було створено і реалізовано в функціях. Також під час розробки використовувалися такі бібліотеки: iostream, string, map, fstream, MyFunctions.h.

Написано алгоритм для змінення наявного файлу та редагування тексту у ньому. Також реалізовано алгоритм пошуку тексту у рядках файлу. Реалізовано введення шуканих даних користувачем, а також виведення результатів з інших файлів у консольному вікні. Передбачено міри безпеки щодо можливих помилок з відкриванням файлів на будь-якому етапі виконання програми.

Мета поставлена на початку роботи була виконана, знання набуті протягом навчального періоду були дуже важливі для реалізації проєкту. Також знання, що були набуті у самостійному пошуку, дали результати у вигляді нових алгоритмів, рішень задач та оптимізації.

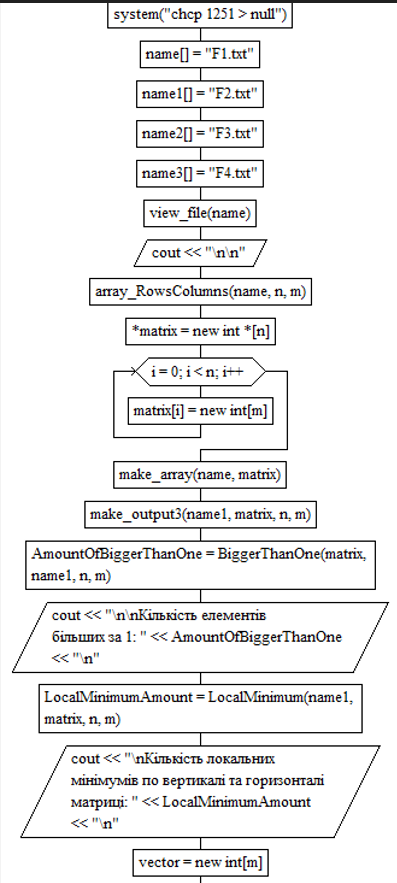
# Список використаних джерел

1. Standard Input / Output Streams Library documentation. Cplusplus.com, 2000-2023 - All rights reserved - v3.3.4s. URL: <https://cplusplus.com/reference/iostream>   
   (дата звернення: 07.04.2024).
2. Strings documentation.  Cplusplus.com, 2000-2023 - All rights reserved - v3.3.4s*.* URL: <https://cplusplus.com/reference/string> (дата звернення: 07.04.2024).
3. Map header documentation. Cplusplus.com, 2000-2023 - All rights reserved - v3.3.4s*.* URL: <https://cplusplus.com/reference/map> (дата звернення: 07.04.2024).
4. File streams documentation.  Cplusplus.com, 2000-2023 - All rights reserved - v3.3.4s*.* URL: <https://cplusplus.com/reference/fstream> (дата звернення: 07.04.2024).
5. Документація з команд операційної системи Віндоус. © Microsoft 2024. 1960 c. URL: <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/chcp> (дата звернення: 07.04.2024).
6. Function system documentation. Cplusplus.com, 2000-2023 - All rights reserved - v3.3.4s*.* URL: <https://cplusplus.com/reference/cstdlib/system> (дата звернення: 07.04.2024).
7. Документація з інтегрованого середовища розробки Visual Studio 2022. 2566 с. URL: <https://learn.microsoft.com/ru-ru/visualstudio/install/install-visual-studio?view=vs-2022> (дата звернення: 07.04.2024).
8. Microsoft Visual Studio, матеріал з вікіпедії URL: <https://uk.wikipedia.org/wiki/Microsoft_Visual_Studio> (дата звернення: 14.04.2024).
9. Трофименко О.Г., Прокоп Ю.В., Задерейко О.В. Алгоритмізація та програмування : навч.-метод. посібник. Одеса: Фенікс, 2020. 310 с URL: <https://dspace.onua.edu.ua/items/15f7a1eb-2d1f-4c3b-823c-89aba78c50b4> (дата звернення: 06.04.2024).

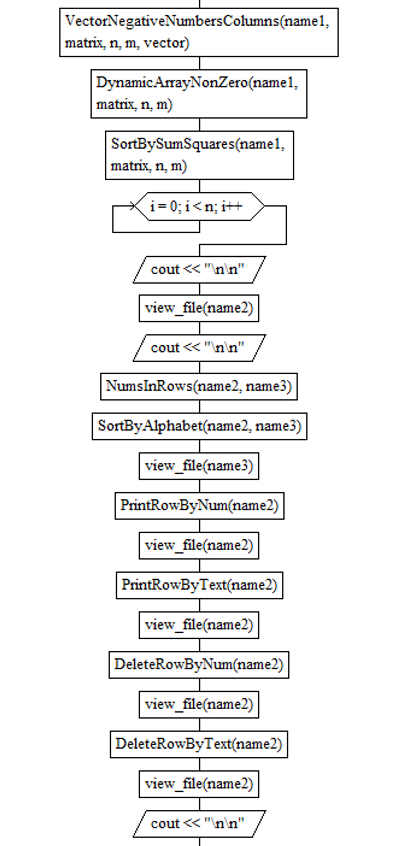
# Додатки

## Додаток А. Схеми алгоритмів функцій програмного проєкту

Початок



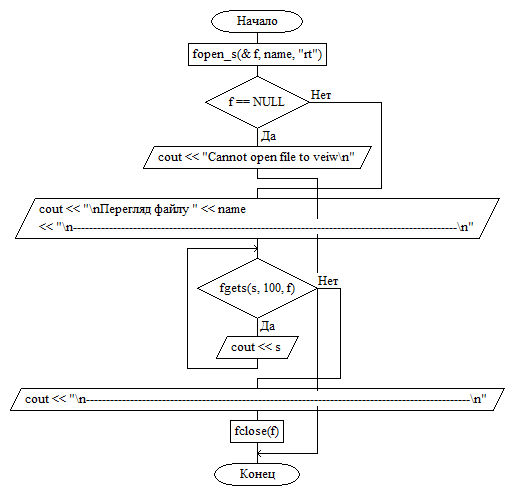
1



1

Кінець

Рисунок А.1 – Блок-схема основного модуля main()



Початок

Кінець

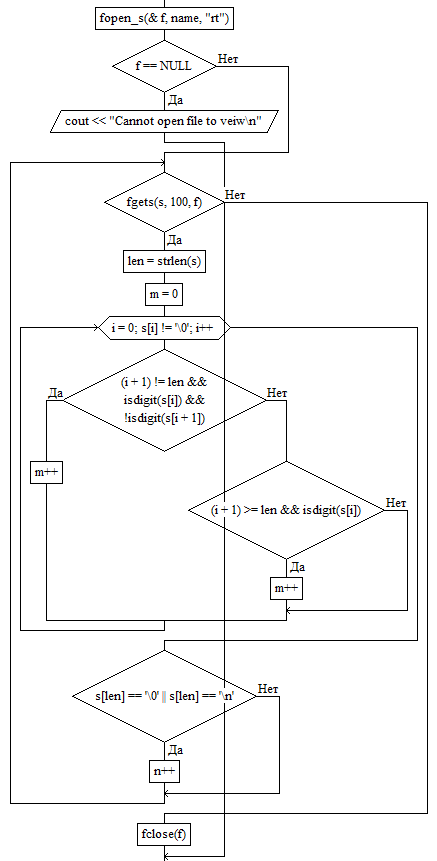
Ні

Ні

Так

Так

Рисунок A.2­­ – Блок-схема для коду view\_file()



Початок

Кінець

Так

Ні

Так

Ні

Так

Ні

Так

Ні

Так

Ні

Рисунок A.3 – Блок-схема для коду array\_RowsColumns()



Початок

Кінець

Так

Ні

Так

Ні

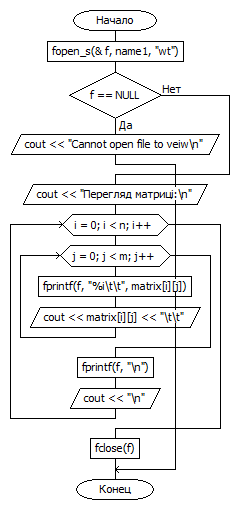
Так

Ні

Так

Ні

Рисунок A.4 – Блок-схема для коду make\_array()



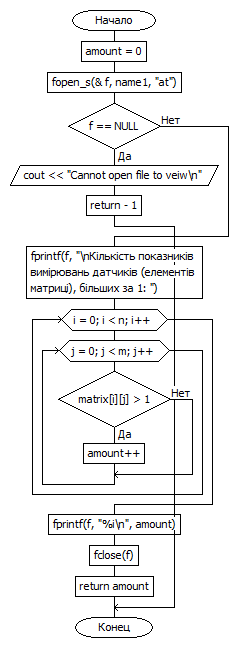
Початок

Кінець

Так

Ні

Рисунок A.5 – Блок-схема для коду make\_output3()



Початок

Кінець

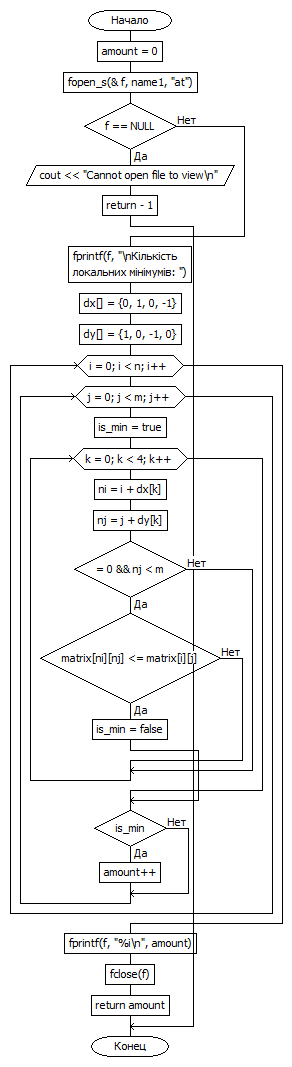
Так

Ні

Так

Ні

Рисунок A.6 – Блок-схема для коду BiggerThanOne()



Початок

Кінець

Так

Ні

Так

Ні

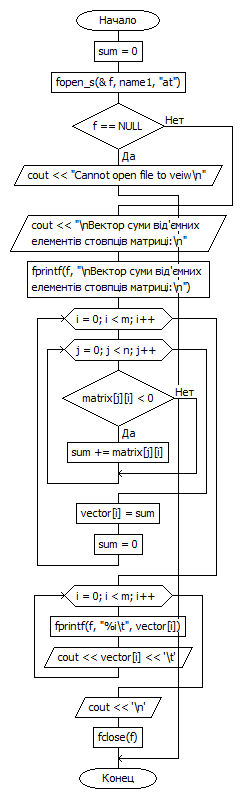
Так

Ні

Так

Ні

Рисунок A.7 – Блок-схема для коду LocalMinimum()



Початок

Кінець

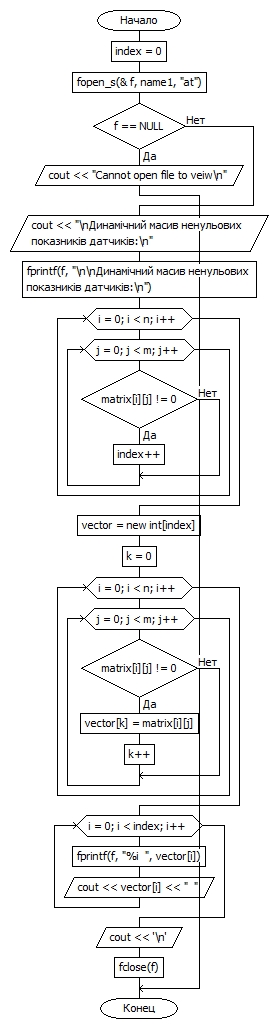
Так

Ні

Так

Ні

Рисунок A.8 – Блок-схема для коду VectorNegativeNumbersColumns()



Початок

Кінець

Так

Ні

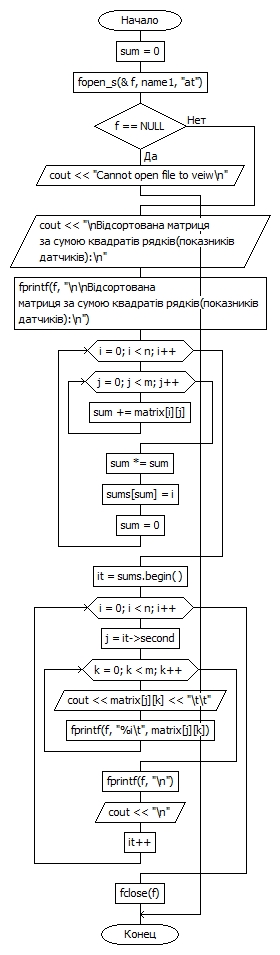
Так

Ні

Так

Ні

Рисунок A.9 – Блок-схема для коду DynamicArrayNonZero()



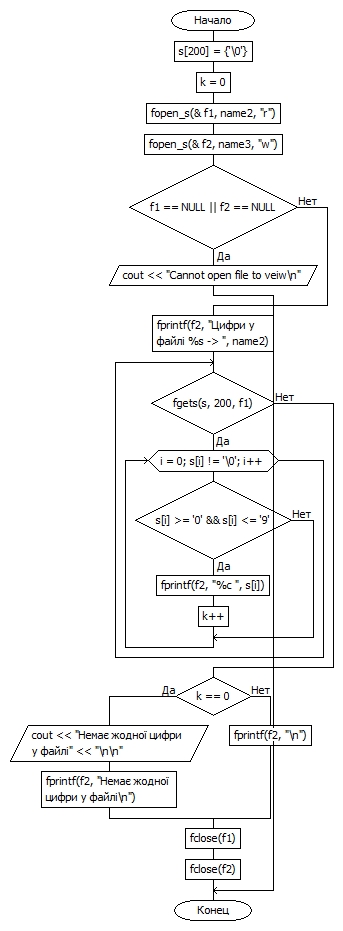
Початок

Кінець

Так

Ні

Рисунок A.10 – Блок-схема для коду SortBySumSquares()



Початок

Кінець

Так

Ні

Так

Так

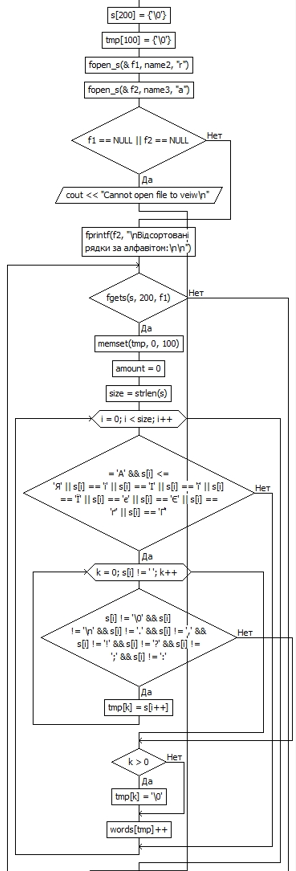
Так

Ні

Ні

Ні

Рисунок A.11 – Блок-схема для коду NumsInRows()



Початок

1

2

3

4

Так

Ні

Так

Ні

Так

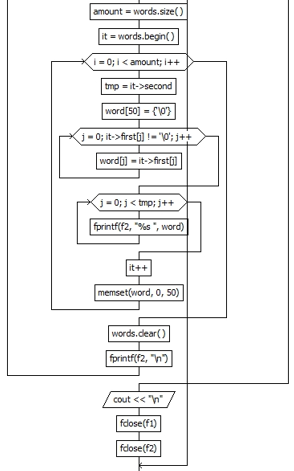
Ні

Ні

Так

Ні

Так



4

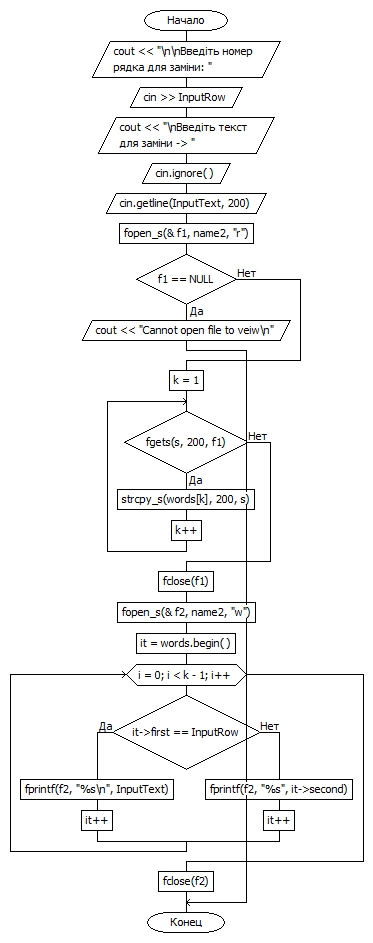
1

2

3

Кінець

Рисунок A.12 – Блок-схема для коду SortByAlphabet()



Початок

Кінець

Так

Ні

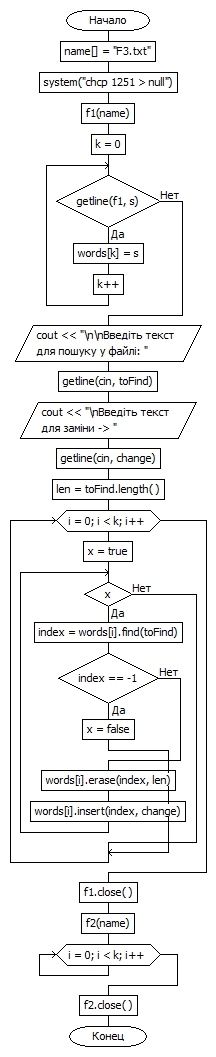
Так

Ні

Так

Ні

Рисунок A.13 – Блок-схема для коду PrintRowByNum()



Кінець

Початок

Так

Ні

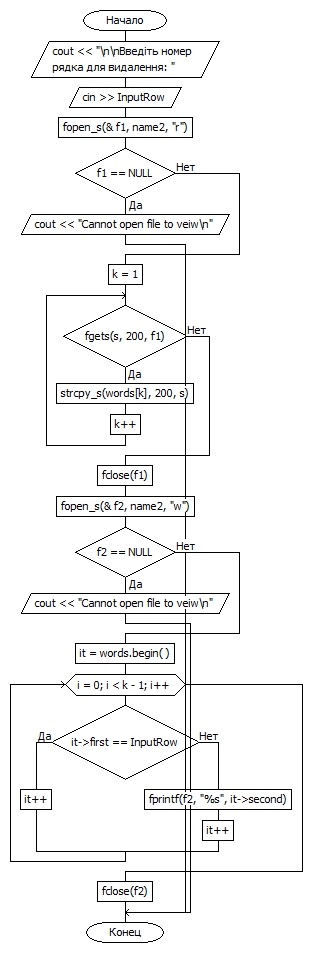
Так

Так

Ні

Ні

Рисунок A.14 – Блок-схема для коду PrintRowByText()



Початок

Кінець

Так

Ні

Так

Ні

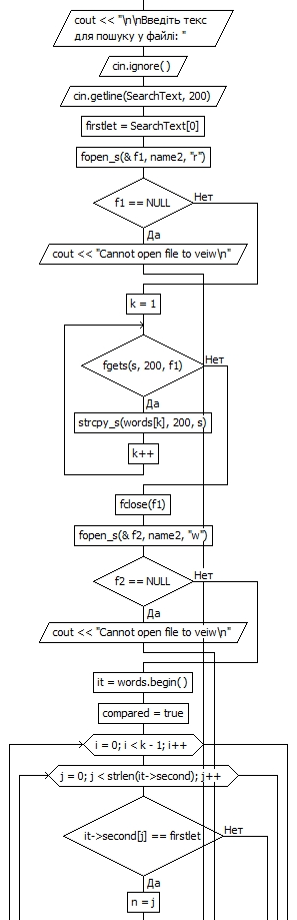
Ні

Так

Так

Ні

Рисунок A.15 – Блок-схема для коду DeleteRowByNum()



Початок

1

2

3

4

5

7

6

8

Так

Так

Так

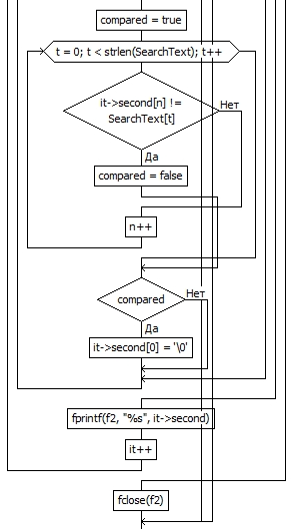
Так

Ні

Ні

Ні

Ні



1

2

3

4

5

6

8

7

Кінець

Так

Так

Ні

Ні

Рисунок A.16 – Блок-схема для коду DeleteRowByText()

## Додаток Б. Програмний код головного файлу

#include <iostream>

#include "MyFunctions.h"

using namespace std;

int main()

{

int n, m;

system("chcp 1251 > null");

char name[] = "F1.txt";

char name1[] = "F2.txt";

char name2[] = "F3.txt";

char name3[] = "F4.txt";

view\_file(name);

cout << "\n\n";

array\_RowsColumns(name, n, m);

int\*\* matrix = new int\* [n];

for (int i = 0; i < n; i++)

matrix[i] = new int[m];

make\_array(name, matrix);

make\_output3(name1, matrix, n, m);

int AmountOfBiggerThanOne = BiggerThanOne(matrix, name1, n, m);

cout << "\n\nКількість елементів більших за 1: " << AmountOfBiggerThanOne << "\n";

int LocalMinimumAmount = LocalMinimum(name1, matrix, n, m);

cout << "\nКількість локальних мінімумів по вертикалі та горизонталі матриці: " << LocalMinimumAmount << "\n";

int\* vector = new int[m];

VectorNegativeNumbersColumns(name1, matrix, n, m, vector);

delete[] vector;

DynamicArrayNonZero(name1, matrix, n, m);

SortBySumSquares(name1, matrix, n, m);

cout << "\n\n";

view\_file(name2);

cout << "\n\n";

NumsInRows(name2, name3);

SortByAlphabet(name2, name3);

view\_file(name3);

PrintRowByNum(name2);

view\_file(name2);

PrintRowByText(name2);

view\_file(name2);

DeleteRowByNum(name2);

view\_file(name2);

DeleteRowByText(name2);

view\_file(name2);

cout << "\n\n";

}

## Додаток В. Програмний код бібліотеки функцій

#include <iostream>

#include <map>

#include <string>

#include <fstream>

#include "MyFunctions.h"

using namespace std;

void view\_file(char\* name)

{

char s[100];

FILE\* f;

fopen\_s(&f, name, "rt");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "\nПерегляд файлу " << name << "\n-------\n";

while (fgets(s, 100, f))

{

cout << s;

}

cout << "\n----------------\n";

fclose(f);

}

void array\_RowsColumns(char\* name, int& n, int& m)

{

n = 0;

char s[100] = { '0' };

FILE\* f;

fopen\_s(&f, name, "rt");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

while (fgets(s, 100, f))

{

int len = strlen(s);

m = 0;

for (int i = 0; s[i] != '\0'; i++)

{

if ((i + 1) != len && isdigit(s[i]) && !isdigit(s[i + 1]))

m++;

else if ((i + 1) >= len && isdigit(s[i]))

m++;

}

n++;

}

fclose(f);

}

void make\_array(char\* name, int\*\* matrix)

{

int row = 0;

char s[100] = { '0' };

FILE\* f;

fopen\_s(&f, name, "rt");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

while (fgets(s, 100, f))

{

int column = 0;

int len = strlen(s);

for (int i = 0; i < len; i++)

{

if (s[i] != ' ' && s[i] != '\0' && s[i] != '\n' && s[i] != '\t')

{

char number[10] = { '\0' };

int j;

for (j = 0; s[i] != ' '; j++)

{

if (s[i] != '\n' && s[i] != '\0' && s[i] != '\t')

number[j] = s[i++];

else break;

}

matrix[row][column++] = atoi(number);

}

}

row++;

}

fclose(f);

}

void make\_output3(char\* name1, int\*\* matrix, int& n, int& m)

{

FILE\* f;

fopen\_s(&f, name1, "wt");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "Перегляд матриці:\n";

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

fprintf(f, "%i\t\t", matrix[i][j]);

cout << matrix[i][j] << "\t\t";

}

fprintf(f, "\n");

cout << "\n";

}

fclose(f);

int BiggerThanOne(int\*\* matrix, char\* name1, int& n, int& m)

{

int amount = 0;

FILE\* f;

fopen\_s(&f, name1, "at");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return -1;

}

fprintf(f, "\nКількість показників вимірювань датчиків (елементів матриці), більших за 1: ");

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

{

if (matrix[i][j] > 1)

amount++;

}

fprintf(f, "%i\n", amount);

fclose(f);

return amount;

}

int LocalMinimum(char\* name1, int\*\* matrix, int& n, int& m) {

int amount = 0;

FILE\* f;

fopen\_s(&f, name1, "at");

if (f == NULL) {

cout << "Cannot open file to view\n";

return -1;

}

fprintf(f, "\nКількість локальних мінімумів: ");

int dx[] = { 0, 1, 0, -1 };

int dy[] = { 1, 0, -1, 0 };

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

bool is\_min = true;

for (int k = 0; k < 4; k++) {

int ni = i + dx[k];

int nj = j + dy[k];

if (ni >= 0 && ni < n && nj >= 0 && nj < m) {

if (matrix[ni][nj] <= matrix[i][j]) {

is\_min = false;

break;

}

}

}

if (is\_min) {

amount++;

}

}

}

fprintf(f, "%i\n", amount);

fclose(f);

return amount;

}

void VectorNegativeNumbersColumns(char\* name1, int\*\* matrix, int& n, int& m, int\* vector)

{

int sum = 0;;

FILE\* f;

fopen\_s(&f, name1, "at");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "\nВектор суми від'ємних елементів стовпців матриці:\n";

fprintf(f, "\nВектор суми від'ємних елементів стовпців матриці:\n");

for (int i = 0; i < m; i++)

{

for (int j = 0; j < n; j++)

{

if (matrix[j][i] < 0)

sum += matrix[j][i];

}

vector[i] = sum;

sum = 0;

}

for (int i = 0; i < m; i++)

{

fprintf(f, "%i\t", vector[i]);

cout << vector[i] << '\t';

}

cout << '\n';

fclose(f);

}

void DynamicArrayNonZero(char\* name1, int\*\* matrix, int& n, int& m)

{

int index = 0;

FILE\* f;

fopen\_s(&f, name1, "at");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "\nДинамічний масив ненульових показників датчиків:\n";

fprintf(f, "\n\nДинамічний масив ненульових показників датчиків:\n");

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

if (matrix[i][j] != 0)

index++;

int\* vector = new int[index];

int k = 0;

for (int i = 0; i < n; i++)

for (int j = 0; j < m; j++)

if (matrix[i][j] != 0)

{

vector[k] = matrix[i][j];

k++;

}

for (int i = 0; i < index; i++)

{

fprintf(f, "%i ", vector[i]);

cout << vector[i] << " ";

}

cout << '\n';

fclose(f);

}

void SortBySumSquares(char\* name1, int\*\* matrix, int& n, int& m)

{

int sum = 0;

map < int, int> sums;

map <int, int> ::iterator it;

fopen\_s(&f, name1, "at");

if (f == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

cout << "\nВідсортована матриця за сумою квадратів рядків(показників датчиків):\n";

fprintf(f, "\n\nВідсортована матриця за сумою квадратів рядків(показників датчиків):\n");

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

sum += matrix[i][j];

}

sum \*= 2;

sums[sum] = i;

sum = 0;

}

it = sums.begin();

for (int i = 0; i < n; i++)

{

int j = it->second;

for (int k = 0; k < m; k++)

{

cout << matrix[j][k] << "\t\t";

fprintf(f, "%i\t", matrix[j][k]);

}

fprintf(f, "\n");

cout << "\n";

it++;

}

for (int i = 0; i < n; i++)

delete[] matrix[i];

delete[] matrix;

fclose(f);

}

void NumsInRows(char\* name2, char\* name3)

{

char s[200] = { '\0' };

int k = 0;

FILE\* f1, \* f2;

fopen\_s(&f1, name2, "r");

fopen\_s(&f2, name3, "w");

if (f1 == NULL || f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

//cout << "Цифри у файлі " << name2 << " -> ";

fprintf(f2, "Цифри у файлі %s -> ", name2);

while (fgets(s, 200, f1))

{

for (int i = 0; s[i] != '\0'; i++)

{

if (s[i] >= '0' && s[i] <= '9')

{

fprintf(f2, "%c ", s[i]);

//cout << s[i] << " ";

k++;

}

}

}

if (k == 0)

{

cout << "Немає жодної цифри у файлі" << "\n\n";

fprintf(f2, "Немає жодної цифри у файлі\n");

}

else

//cout << "\n\n";

fprintf(f2, "\n");

fclose(f1);

fclose(f2);

}

void SortByAlphabet(char\* name2, char\* name3)

{

char s[200] = { '\0' };

char tmp[100] = { '\0' };

map <string, int> words;

map <string, int> ::iterator it;

FILE\* f1, \* f2;

fopen\_s(&f1, name2, "r");

fopen\_s(&f2, name3, "a");

if (f1 == NULL || f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

fprintf(f2, "\nВідсортовані рядки за алфавітом:\n\n");

while (fgets(s, 200, f1))

{

memset(tmp, 0, 100);

int amount = 0;

int size = strlen(s);

for (int i = 0; i < size; i++)

{

if (s[i] >= 'а' && s[i] <= 'я' || s[i] >= 'А' && s[i] <= 'Я' || s[i] == 'і' || s[i] == 'І' || s[i] == 'ї' || s[i] == 'Ї' || s[i] == 'є' || s[i] == 'Є' || s[i] == 'ґ' || s[i] == 'Ґ')

{

int k;

for (k = 0; s[i] != ' '; k++)

{

if (s[i] != '\0' && s[i] != '\n' && s[i] != '.' && s[i] != ',' && s[i] != '!' && s[i] != '?' && s[i] != ';' && s[i] != ':')

tmp[k] = s[i++];

else

{

break;

}

}

if (k > 0)

{

tmp[k] = '\0';

}

words[tmp]++;

}

}

amount = words.size();

it = words.begin();

for (int i = 0; i < amount; i++)

{

int tmp = it->second;

char word[50] = { '\0' };

for (int j = 0; it->first[j] != '\0'; j++)

word[j] = it->first[j];

for (int j = 0; j < tmp; j++)

{

fprintf(f2, "%s ", word);

}

it++;

memset(word, 0, 50);

}

words.clear();

fprintf(f2, "\n");

}

cout << "\n";

fclose(f1);

fclose(f2);

}

void PrintRowByNum(char\* name2)

{

int InputRow;

char InputText[200];

cout << "\n\nВведіть номер рядка для заміни: ";

cin >> InputRow;

cout << "\nВведіть текст для заміни -> ";

cin.ignore();

cin.getline(InputText, 200);

map <int, char[200]> words;

map <int, char[200]> ::iterator it;

char s[200];

FILE\* f1, \* f2;

fopen\_s(&f1, name2, "r");

if (f1 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

int k = 1;

while (fgets(s, 200, f1))

{

strcpy\_s(words[k], 200, s);

k++;

}

fclose(f1);

fopen\_s(&f2, name2, "w");

if (f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

it = words.begin();

for (int i = 0; i < k - 1; i++)

{

if (it->first == InputRow)

{

fprintf(f2, "%s\n", InputText);

it++;

}

else

{

fprintf(f2, "%s", it->second);

it++;

}

}

fclose(f2);

}

void PrintRowByText(char\* name2)

{

char name[] = "F3.txt";

system("chcp 1251 > null");

ifstream f1(name);

string toFind;

string change;

string s;

map <int, string> words;

map <int, string> ::iterator it;

int k = 0;

while (getline(f1, s))

{

words[k] = s;

k++;

}

cout << "\n\nВведіть текст для пошуку у файлі: ";

getline(cin, toFind);

cout << "\nВведіть текст для заміни -> ";

getline(cin, change);

int len = toFind.length();

bool x;

for (int i = 0; i < k; i++)

{

x = true;

while (x)

{

int index = words[i].find(toFind);

if (index == -1)

{

x = false;

break;

}

words[i].erase(index, len);

words[i].insert(index, change);

}

}

f1.close();

ofstream f2(name);

for (int i = 0; i < k; i++)

{

f2 << words[i];

f2 << "\n";

}

f2.close();

}

void DeleteRowByNum(char\* name2)

{

int InputRow;

cout << "\n\nВведіть номер рядка для видалення: ";

cin >> InputRow;

map <int, char[200]> words;

map <int, char[200]> ::iterator it;

char s[200];

FILE\* f1, \* f2;

fopen\_s(&f1, name2, "r");

if (f1 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

int k = 1;

while (fgets(s, 200, f1))

{

strcpy\_s(words[k], 200, s);

k++;

}

fclose(f1);

fopen\_s(&f2, name2, "w");

if (f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

it = words.begin();

for (int i = 0; i < k - 1; i++)

{

if (it->first == InputRow)

{

it++;

}

else

{

fprintf(f2, "%s", it->second);

it++;

}

}

fclose(f2);

}

void DeleteRowByText(char\* name2)

{

char SearchText[200];

cout << "\n\nВведіть текст для пошуку у файлі: ";

cin.ignore();

cin.getline(SearchText, 200);

char firstlet = SearchText[0];

map <int, char[200]> words;

map <int, char[200]> ::iterator it;

char s[200];

FILE\* f1, \* f2;

fopen\_s(&f1, name2, "r");

if (f1 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

int k = 1;

while (fgets(s, 200, f1))

{

strcpy\_s(words[k], 200, s);

k++;

}

fclose(f1);

fopen\_s(&f2, name2, "w");

if (f2 == NULL)

{

cout << "Cannot open file to veiw\n";

return;

}

it = words.begin();

bool compared = true;

for (int i = 0; i < k - 1; i++)

{

for (int j = 0; j < strlen(it->second); j++)

{

if (it->second[j] == firstlet)

{

int n = j;

compared = true;

for (int t = 0; t < strlen(SearchText); t++)

{

if (it->second[n] != SearchText[t])

{

compared = false;

break;

}

n++;

}

if (compared)

{

it->second[0] = '\0';

}

}

}

fprintf(f2, "%s", it->second);

it++;

}

fclose(f2);

}