**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 ІПЗ-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;

36

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 із 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.
* Відредагувати файл F3.txt - \*\* 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.
* Видалити з файлу F3.txt \*\* 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](#_Toc162859612)

[1 Аналіз засобів та методів розробки 7](#_Toc162859613)

[2 Словесний опис алгоритмів для розв’язання задач 8](#_Toc162859614)

[2.1 Опис функції main() 8](#_Toc162859615)

[2.2 Опис функції view\_file() 11](#_Toc162859616)

[2.3 Опис функції array\_RowsColumns() 12](#_Toc162859617)

[2.4 Опис функції make\_array 14](#_Toc162859618)

[2.5 Опис функції make\_output3() 15](#_Toc162859619)

[2.6 Опис функції BiggerThanOne() 16](#_Toc162859620)

[2.7 Опис функції LocalMinimum() 17](#_Toc162859621)

[2.8 Опис функції VectorNegativeNumbersColumns() 19](#_Toc162859622)

[2.9 Опис функції DynamicArrayNonZero() 20](#_Toc162859623)

[2.10 Опис функції SortBySumSquares() 21](#_Toc162859624)

[2.11 Опис функції NumsInRows() 23](#_Toc162859625)

[2.12 Опис функції SortByAlphabet() 24](#_Toc162859626)

[3 Результати 28](#_Toc162859627)

[4 Інструкція з використання програми 29](#_Toc162859628)

[5 Висновки з аналізом досягнутих результатів 30](#_Toc162859629)

[Список використаних джерел 31](#_Toc162859630)

[додатки 32](#_Toc162859631)

[Додаток А. Схеми алгоритмів функцій програмного проекту 32](#_Toc162859632)

[Додаток A.1 «Блок схема основного модуля main()» 32](#_Toc162859633)

[Додаток A.2 «Блок схема для коду view\_file()» 34](#_Toc162859634)

[Додаток A.3 «Блок схема для коду array\_RowsColumns()» 35](#_Toc162859635)

[Додаток A.4 «Блок схема для коду make\_array()» 36](#_Toc162859636)

[Додаток A.5 «Блок схема для коду make\_output3()» 37](#_Toc162859637)

[Додаток A.6 «Блок схема для коду BiggerThanOne()» 38](#_Toc162859638)

[Додаток A.7 «Блок схема для коду LocalMinimum()» 39](#_Toc162859639)

[Додаток A.8 «Блок схема для коду VectorNegativeNumbersColumns()» 40](#_Toc162859640)

[Додаток A.9 «Блок схема для коду DynamicArrayNonZero()» 41](#_Toc162859641)

[Додаток A.10 «Блок схема для коду SortBySumSquares()» 42](#_Toc162859642)

[Додаток A.11 «Блок схема для коду NumsInRows()» 43](#_Toc162859643)

[Додаток A.12 «Блок схема для коду SortByAlphabet()» 44](#_Toc162859644)

[Додаток A.13 «Блок схема для коду PrintRowByNum()» 46](#_Toc162859645)

[Додаток A.14 «Блок схема для коду PrintRowByText()» 47](#_Toc162859646)

[Додаток A.15 «Блок схема для коду DeleteRowByNum()» 48](#_Toc162859647)

[Додаток A.16 «Блок схема для коду DeleteRowByText()» 49](#_Toc162859648)

[Додаток Б «Програмний код» 51](#_Toc162859649)

[Додаток Б.13 «Програмний код функції, що замінює введений рядок на введений текст» 51](#_Toc162859650)

[Додаток Б.14 «Програмний код функції, що замінює введений текст у рядку іншим введеним текстом» 52](#_Toc162859651)

[Додаток Б.15 «Програмний код функції, що видаляє введений рядок» 52](#_Toc162859652)

[Додаток Б.16 «Програмний код функції, що видаляє рядок з введеним текстом» 53](#_Toc162859653)

# Вступ

Метою роботи є застосування набутих знань протягом навчального періоду, для розв'язання різноманітних завдань на різних рівнях складності.

Важливість цієї роботи є 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 проекту.

Актуальність опрацювання текстових даних наразі є дуже високою, такі завдання як пошук слів та заміна на введене слово, також видалення слів після їх пошуку є досить важливими. У цифровому світі сучасності неможливо уникнути обробки файлів, і це дуже очевидно навіть зараз, коли для написання цього документа використовуються програми, подібні до "Word", для створення та редагування документів різних форматів.

Об'єкти дослідження є не лише актуальними, але й необхідними в контексті сучасних вимог до обробки інформації.

Курсова робота складається із завдань, календарного плану, змісту, вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг курсової роботи: \* с., \* рис., \* табл., \* додатки, \* джерел.

# 1 Аналіз засобів та методів розробки

# 2 Словесний опис алгоритмів для розв’язання задач

## 2.1 Опис функції main()

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

Функція має тип int, повертає числове значення після закінчення роботи програми або у випадку 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

Команда system chcp 1251 призначена для зміни кодової сторінки (code page) у командному рядку.

view\_file(name);

Виклик функції з аргументом name – ім’я файлу , виводить в консоль вміст фалу.

Виводить в консоль два рази символ переходу на новий рядок

cout << "\n\n";

array\_RowsColumns(name, n, m);

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

Створення динамічного двомірного масиву для матриці з файлу:

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

створює одновимірний n-масив вказівників

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

matrix[i] = new int[m];

цикл, що ітерує по кожній клітинці масиву вказівників та створює динамічний масив в кожній ітерації циклу і записує вказівник на перший елемент в i-тий елемент масиву matrix.

make\_array(name, matrix);

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\_output3(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 виводить матрицю в консоль.

int AmountOfBiggerThanOne = BiggerThanOne(matrix, name1, 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.

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

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

int LocalMinimumAmount = LocalMinimum(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 чисельне значення кількості локальних мінімумів в матриці.

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

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

int\* vector = new int[m];

Створення динамічного масиву цілочислового типу з розмірністю в стовпці матриці.

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

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 вектора з сум негативних чисел стовпців матриці.

delete[] vector;

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

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 створює масив з ненульовими елементами матриці.

SortBySumSquares(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.

cout << "\n\n";

Виводить два символи переходу на новий рядок.

view\_file(name2);

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

cout << "\n\n";

Виводить два символи переходу на новий рядок.

NumsInRows(name2, name3);

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.

SortByAlphabet(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.

view\_file(name3);

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

PrintRowByNum(name2);

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

view\_file(name2);

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

PrintRowByText(name2);

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

view\_file(name2);

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

DeleteRowByNum(name2);

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

view\_file(name2);

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

DeleteRowByText(name2);

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

view\_file(name2);

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

cout << "\n\n";

Виводить два символи переходу на новий рядок.

Блок-схема цієї функції – Додаток А.1

## 2.2 Опис функції view\_file()

#include <iostream>

#include <cstring>

#include <map>

#include <string>

#include <fstream>

#include "MyFunctions.h"

Підключення необхідних бібліотек для виконання різних функцій.

void view\_file(char\* name) //функція для виведення вмісту файлу

{

char s[100];

Оголошення тимчасової змінної для зчитування рядків.

FILE\* f;

Оголошення змінної вказівника для файлу.

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

if (f == NULL)

{

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

return;

}

Відкриття текстового файлу у режимі читання перевірка чи знайшла його програма.

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

while (fgets(s, 100, f))

{

cout << s;

}

Цикл, що виводить зміст фалу 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.

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

fclose(f);

Закриваємо файл, що відкрили на початку програми.

}

Блок-схема цієї функції – Додаток А.2

## 2.3 Опис функції array\_RowsColumns()

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

void array\_RowsColumns(char\* name, int& n, int& m) //функція, що обчислює рядки та стовпчики у файлі F1 та передає їх за адресом

{

n = 0;

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

Оголошення змінної для читання рядків.

FILE\* f;

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

Відкриття текстового файлу у режимі читання.

if (f == NULL)

{

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

return;

}

Перевірка дійсності вказівника на файл.

while (fgets(s, 100, f))

{

Порядкове зчитування дан=их з фалу.

int len = strlen(s);

Обчислення довжини масиву з рядком.

m = 0;

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

{

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

m++;

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

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

m++;

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

}

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

n++;

}

fclose(f);

Закриваємо файл.

}

Блок-схема цієї функції – Додаток А.3

## 2.4 Опис функції make\_array

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

void make\_array(char\* name, int\*\* matrix) //функція, що створює динамічний двомірний масив з показників вимірювань у файлі F1

{

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;

}

Відкриваємо файл і перевіряємо його дійсність.

while (fgets(s, 100, f))

{

int column = 0;

int len = strlen(s);

Обчислюємо довжину тимчасового масиву рядка.

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

{

Цикл, що ітерує по рядку.

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

{

Якщо елемент не є пробілом, кінцем масиву, символом наст. рядка або табуляції виконується наступне

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

int j;

Оголошення тимчасового масиву для числа.

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

{

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

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

else break;

}

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

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

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

}

}

row++;

Збільшення числа для рядка після кожного прочитаного рядка у файлі.

}

fclose(f);

Закриття файлу.

}

Блок-схема цієї функції – Додаток А.4

## 2.5 Опис функції make\_output3()

Аргументи функції: ім’я файлу для записування, матриця та кількість рядків, стовпців матриці.

void make\_output3(char\* name1, int\*\* matrix, int& n, int& m) //функція, що виводить створену матрицю в консоль, а також записує в файл F2

{

FILE\* f;

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

if (f == NULL)

{

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

return;

}

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

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";

}

Двійний цикл для ітерування по матриці і записування її до файлу з виводом до консолі. Функція fprintf() записує дані до файлу, який йде до неї як перший аргумент.

fprintf(f, "\n");

cout << "\n";

}

fclose(f);

Закриваємо файл.

}

Блок-схема функції – Додаток 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.

int BiggerThanOne(int\*\* matrix, char\* name1, int& n, int& m) //функція, що обчислює кількість елементів більших за 1 в матриці та записує в файл F2

{

int amount = 0;

FILE\* f;

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

if (f == NULL)

{

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

return -1;

}

Відкриваємо текстовий файл для дописування даних та перевіряємо його дійсність.

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++;

}

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

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

fclose(f);

return amount;

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

}

Блок-схема функції – Додаток 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.

int LocalMinimum(char\* name1, int\*\* matrix, int& n, int& m) // функція повертає кількість локальних мінімумів по горизонталі та вертикалі та записує у файл F2

{

int amount = 0;

FILE\* f;

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

if (f == NULL) {

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

return -1;

}

Відкриваємо текстовий файл для дописування даних та перевіряємо його дійсність.

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

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

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

Масиви для перевірки умов локального мінімуму.

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

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

Двійний цикл для ітерації по матриці.

bool is\_min = true;

Оголошення булевої змінної для перевірки мінімальності елемента.

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

int ni = i + dx[k];

int nj = j + dy[k];

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

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

За умови, що сусідній елемент не виходить за межі масиву виконується наступне

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

is\_min = false;

break;

За умови, що елемент є більший за один сусідній елемент змінна для перевірки мінімальності набуває значення false.

}

}

}

if (is\_min) {

amount++;

За умови, що змінна is\_min є true кількість локальних мінімумів збільшується на один.

}

}

}

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

fclose(f);

return amount;

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

}

Блок-схема функції – Додаток А.7

## 2.8 Опис функції VectorNegativeNumbersColumns()

Аргументи функції: ім’я файлу для записування, матриця, її розмірність у двох змінних та вектор.

void VectorNegativeNumbersColumns(char\* name1, int\*\* matrix, int& n, int& m, int\* vector) //функція для обчислення вектора за сумою від'ємних елементів стовпців матриці та записує результат у файл F2

{

int sum = 0;;

FILE\* f;

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

if (f == NULL)

{

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

return;

}

Відкриваємо текстовий файл для дописування даних та перевіряємо його дійсність.

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];

}

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

vector[i] = sum;

sum = 0;

}

Записуємо суму елементів до вектору та записуємо 0 до неї.

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

{

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

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

}

cout << '\n';

fclose(f);

Записуємо вектор у файл за допомогою циклу, закриваємо файл.

}

Блок-схема функції – Додаток А.8

## 2.9 Опис функції DynamicArrayNonZero()

Аргументи функції: ім’я файлу для записування, матриця, її розмірність у двох змінних.

void DynamicArrayNonZero(char\* name1, int\*\* matrix, int& n, int& m) //функція створює динамічний масив з ненульових елементів матриці та записує результат у файл F2

{

int index = 0;

FILE\* f;

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

if (f == NULL)

{

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

return;

}

Відкриваємо текстовий файл для дописування даних та перевіряємо його дійсність.

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++;

Двійний цикл, що обчислює кількість ненульових елементів в матриці.

int\* vector = new int[index];

Створення динамічного одновимірного масиву.

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++;

}

Двійний цикл, що записує всі ненульові елементи матриці в вектор.

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

{

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

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

}

cout << '\n';

fclose(f);

Записуємо результат у файл за допомогою циклу і закриваємо файл.

}

Блок-схема функції – Додаток А.9

## 2.10 Опис функції SortBySumSquares()

Аргументи функції: ім’я файлу для записування, матриця, її розмірність у двох змінних.

void SortBySumSquares(char\* name1, int\*\* matrix, int& n, int& m) //функція, що розташовує рядки матриці сортуючи за квадратом суми та записує результат у файл F2

{

int sum = 0;

map < int, int> sums;

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

map <int, int> ::iterator it;

Створення ітератора для map sums

FILE\* f;

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

if (f == NULL)

{

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

return;

}

Відкриваємо текстовий файл для дописування даних та перевіряємо його дійсність.

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;

}

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

it = sums.begin();

Ітератор ставиться на початкову позицію словника sums.

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

{

int j = it->second;

Змінна для номеру рядка.

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

{

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

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

}

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

fprintf(f, "\n");

cout << "\n";

it++;

Збільшення ітератора на один – наступний елемент словника.

}

fclose(f);

Закриваємо файл.

}

Блок-схема функції – Додаток А.10

## 2.11 Опис функції NumsInRows()

Аргументи функції: ім’я файлу для читання, ім’я фалу для записування.

void NumsInRows(char\* name2, char\* name3) //функція визначає, чи є у рядках цифри файлу F3, виводить їх та записує результат у файл F4

{

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;

}

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

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

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

while (fgets(s, 200, f1))

{

Цикл для читання файлу по рядкам.

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

{

Цикл, що ітерує по масиву до символу закінчення.

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

{

Умова, що перевіряє чи є елемент цифрою.

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

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

k++;

Якщо елемент є цифрою він записується у файл та виводиться в консоль.

}

}

}

if (k == 0)

{

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

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

}

Перевірка умови, чи є k нульовою змінною.

else

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

fprintf(f2, "\n");

fclose(f1);

fclose(f2);

Закриваємо два файли.

}

Блок-схема функції – Додаток А.11

## 2.12 Опис функції SortByAlphabet()

Аргументи функції: ім’я файлу для читання, ім’я фалу для записування.

void SortByAlphabet(char\* name2, char\* name3) //функція, що сортує слова за алфавітом в кожному рядку та записує результат у файл F4

{

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

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

map <string, int> words;

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

map <string, int> ::iterator it;

Створення ітератору для словника words.

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;

}

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

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

while (fgets(s, 200, f1))

{

Цикл, що читає рядки з файлу f1.

memset(tmp, 0, 100);

Обнуління тимчасової змінної для рядку.

int amount = 0;

int size = strlen(s);

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

{

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

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')

{

Перевірка чи є елемент буквою алфавіту.

int k;

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

{

Цикл для записування слова у тимчасову змінну.

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

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

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.

else

{

break;

}

}

if (k > 0)

{

tmp[k] = '\0';

}

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

words[tmp]++;

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

}

}

amount = words.size();

Зміна, що приймає значення кількості елементів в словнику words.

it = words.begin();

Ітератор переміщується на початок словника.

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

{

Цикл, що проходить повністю по словнику відсортованих слів.

int tmp = it->second;

Тимчасова змінна приймає значення елемента словника.

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

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

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

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

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

{

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

}

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.

it++;

Ітератор збільшується на один для переходу до наступного елементу.

memset(word, 0, 50);

Скидання тимчасового масиву до 0.

}

words.clear();

Скидання повністю словника до 0 елментів.

fprintf(f2, "\n");

}

cout << "\n";

fclose(f1);

fclose(f2);

Закриття файлів.

}

Блок-схема функції – Додаток А.12

# 3 Результати

# 4 Інструкція з використання програми

# 5 Висновки з аналізом досягнутих результатів

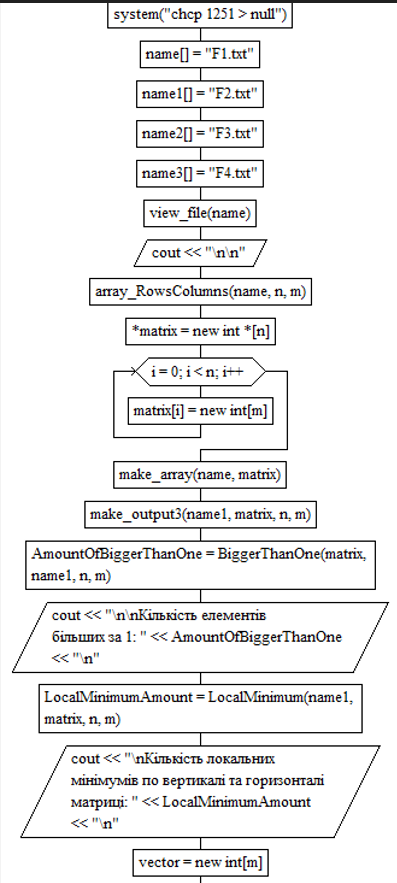
# Список використаних джерел

# додатки

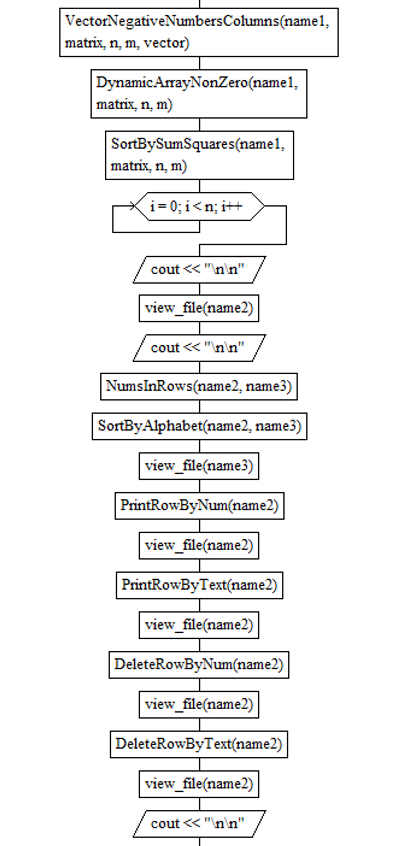
## Додаток А. Схеми алгоритмів функцій програмного проекту

### Додаток A.1 «Блок схема основного модуля main()»

Початок



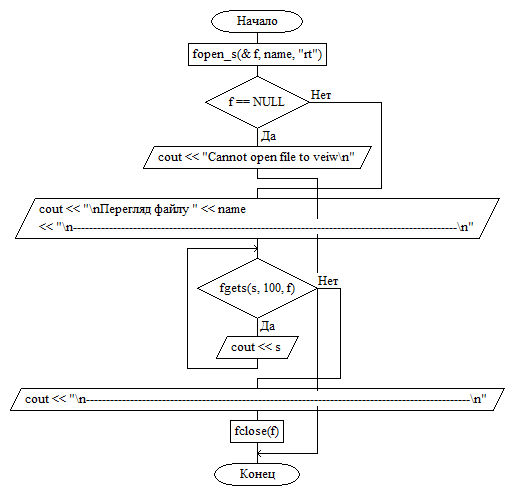
1



1

Кінець

### Додаток A.2 «Блок схема для коду view\_file()»



Початок

Кінець

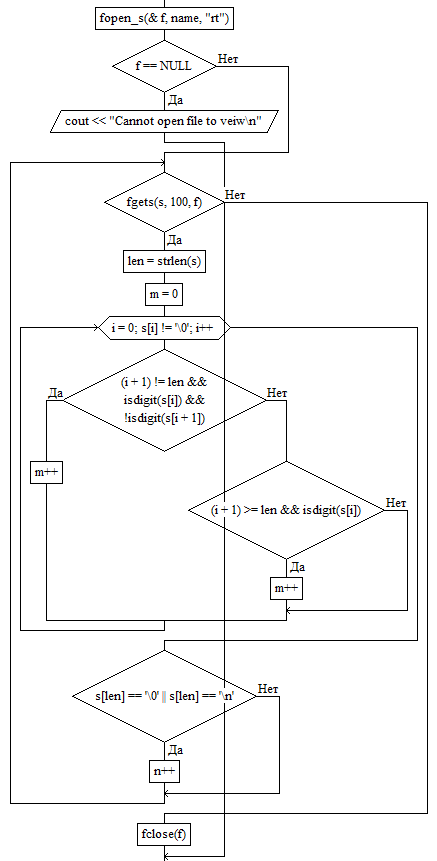
Ні

Ні

Так

Так

### Додаток A.3 «Блок схема для коду array\_RowsColumns()»



Початок

Кінець

Так

Ні

Так

Ні

Так

Ні

Так

Ні

Так

Ні

### Додаток A.4 «Блок схема для коду make\_array()»



Початок

Кінець

Так

Ні

Так

Ні

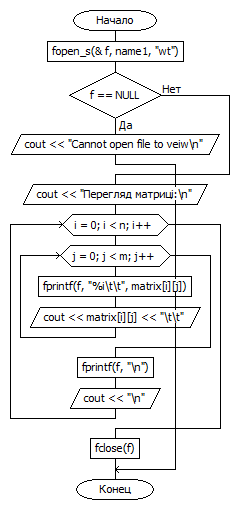
Так

Ні

Так

Ні

### Додаток A.5 «Блок схема для коду make\_output3()»



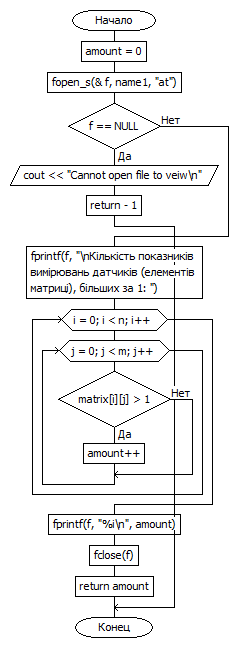
Початок

Кінець

Так

Ні

### Додаток A.6 «Блок схема для коду BiggerThanOne()»



Початок

Кінець

Так

Ні

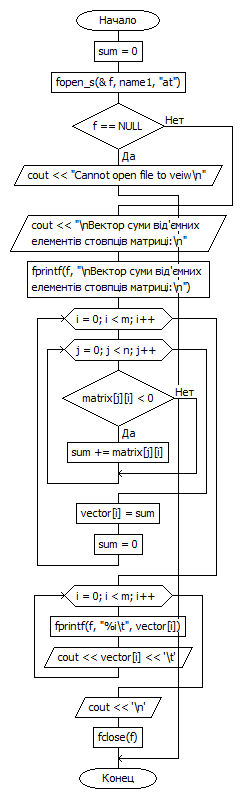
Так

Ні

### Додаток A.7 «Блок схема для коду LocalMinimum()»

|  |
| --- |
| Початок  Кінець  Так  Ні  Так  Ні  Так  Ні  Так  Ні |
|  |  |

### Додаток A.8 «Блок схема для коду VectorNegativeNumbersColumns()»



Початок

Кінець

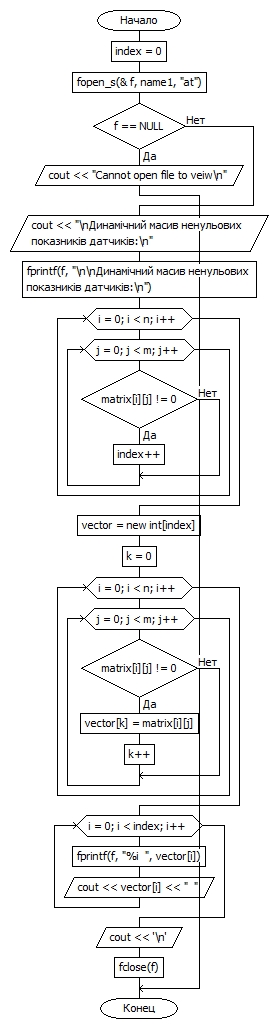
Так

Ні

Так

Ні

### Додаток A.9 «Блок схема для коду DynamicArrayNonZero()»



Початок

Кінець

Так

Ні

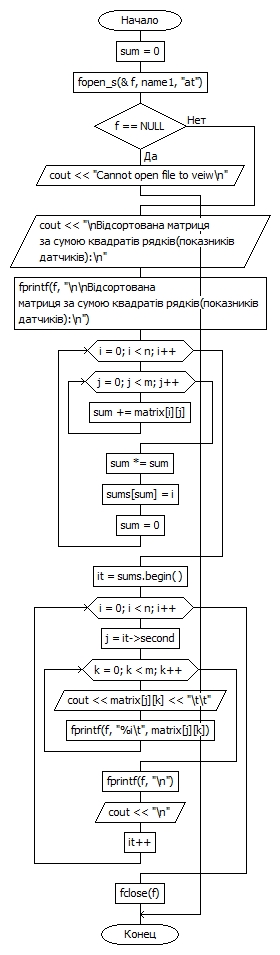
Так

Ні

Так

Ні

### Додаток A.10 «Блок схема для коду SortBySumSquares()»



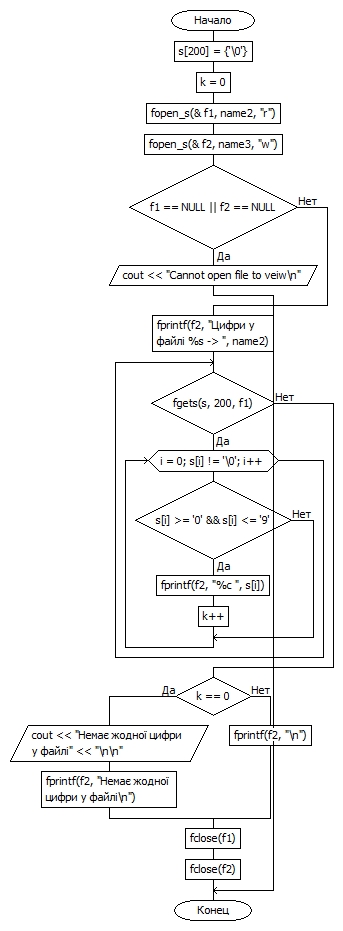
Початок

Кінець

Так

Ні

### Додаток A.11 «Блок схема для коду NumsInRows()»



Початок

Кінець

Так

Ні

Так

Так

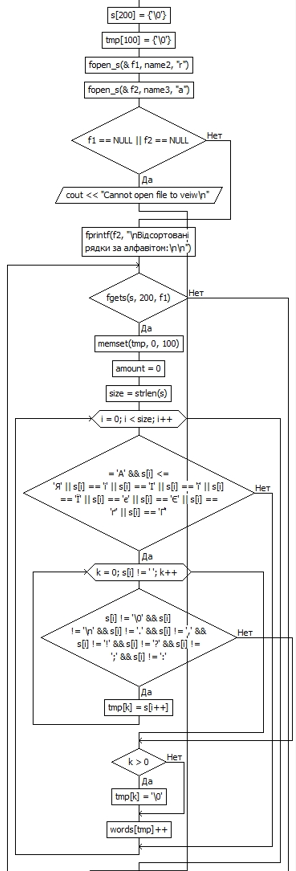
Так

Ні

Ні

Ні

### Додаток A.12 «Блок схема для коду SortByAlphabet()»



Початок

1

2

3

4

Так

Ні

Так

Ні

Так

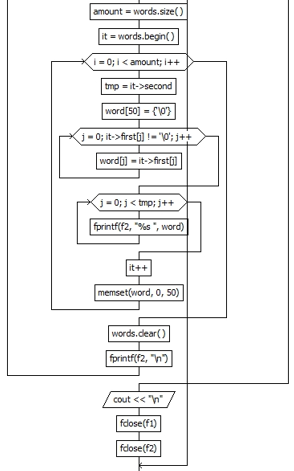
Ні

Ні

Так

Ні

Так



4

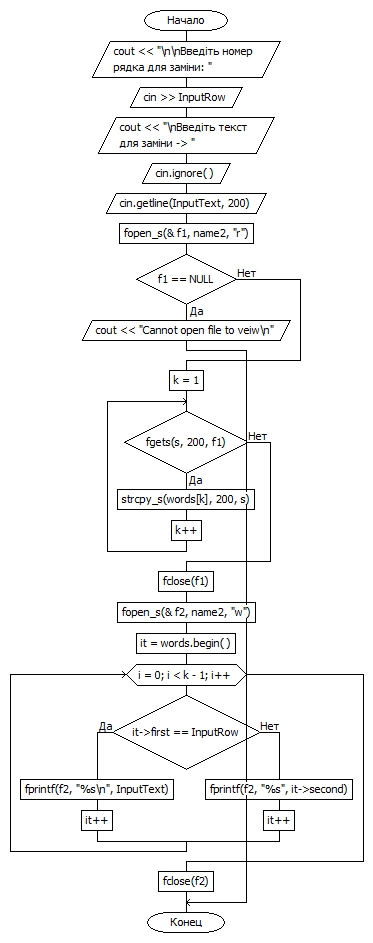
1

2

3

Кінець

### Додаток A.13 «Блок схема для коду PrintRowByNum()»



Початок

Кінець

Так

Ні

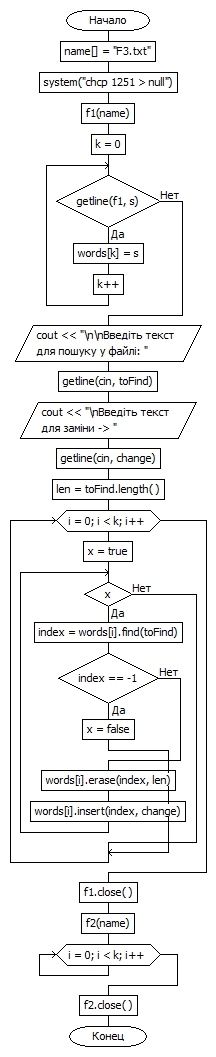
Так

Ні

Так

Ні

### Додаток A.14 «Блок схема для коду PrintRowByText()»



Кінець

Початок

Так

Ні

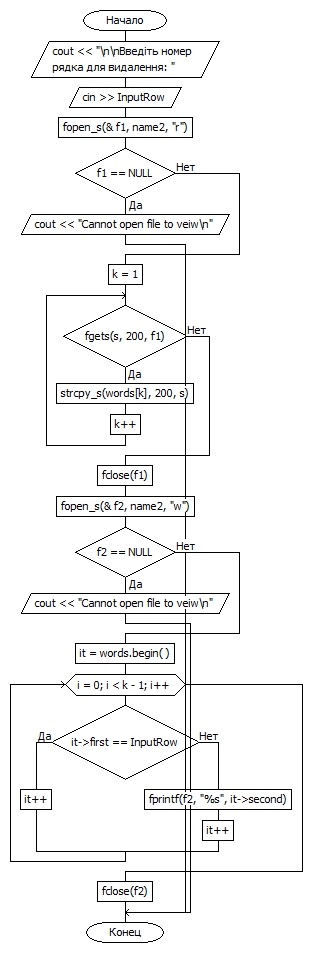
Так

Так

Ні

Ні

### Додаток A.15 «Блок схема для коду DeleteRowByNum()»



Початок

Кінець

Так

Ні

Так

Ні

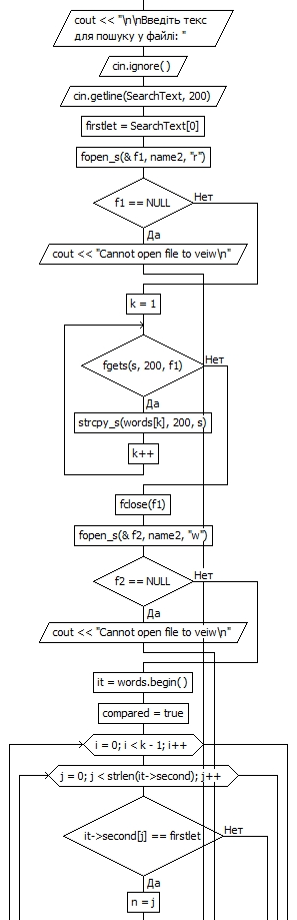
Ні

Так

Так

Ні

### Додаток A.16 «Блок схема для коду DeleteRowByText()»



Початок

1

2

3

4

5

7

6

8

Так

Так

Так

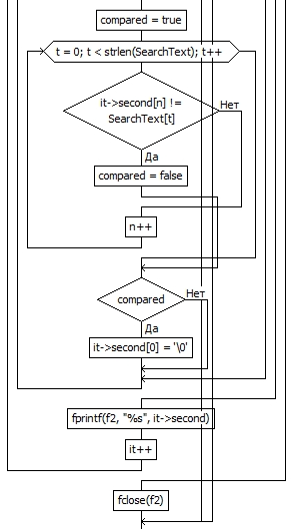
Так

Ні

Ні

Ні

Ні



1

2

3

4

5

6

8

7

Кінець

Так

Так

Ні

Ні

## Додаток Б «Програмний код»

### Додаток Б.13 «Програмний код функції, що замінює введений рядок на введений текст»

void PrintRowByNum(char\* name2) //функція, що замінює введений рядок на введений текст у файлі F3

{

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; //створення map для записування речень у значення та їх номери у ключі

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);

}

### Додаток Б.14 «Програмний код функції, що замінює введений текст у рядку іншим введеним текстом»

void PrintRowByText(char\* name2) //функція, що замінює введений текст у рядку іншим введеним текстом у файлі F3

{

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();

}

### Додаток Б.15 «Програмний код функції, що видаляє введений рядок»

void DeleteRowByNum(char\* name2) //функція, що видаляє введений рядок у файлі F3

{

int InputRow;

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

cin >> InputRow;

map <int, char[200]> words; //створення map для записування речень у значення та їх номери у ключі

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);

}

### Додаток Б.16 «Програмний код функції, що видаляє рядок з введеним текстом»

void DeleteRowByText(char\* name2) //функція, що видаляє рядок з введеним текстом у файлі F3

{

char SearchText[200];

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

cin.ignore();

cin.getline(SearchText, 200);

char firstlet = SearchText[0];

map <int, char[200]> words; //створення map для записування речень у значення та їх номери у ключі

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);

}