**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 | 19503.2024 – 10.04.2024 |

Зміст

[Вступ 6](#_Toc162355345)

[1 Аналіз засобів та методів розробки 7](#_Toc162355346)

[2 Словесний опис алгоритмів для розв’язання задач 8](#_Toc162355347)

[2.1 Опис функції main() 8](#_Toc162355348)

[3 Результати 12](#_Toc162355349)

[4 Інструкція з використання програми 13](#_Toc162355350)

[5 Висновки з аналізом досягнутих результатів 14](#_Toc162355351)

[Список використаних джерел 15](#_Toc162355352)

[додатки 16](#_Toc162355353)

[Додаток А. Схеми алгоритмів функцій програмного проекту 16](#_Toc162355354)

[Додаток A.1 «Блок схема основного модуля» 16](#_Toc162355355)

[Додаток A.2 «Блок схема для коду з додатку Б.2» 18](#_Toc162355356)

[Додаток Б «Програмний код» 19](#_Toc162355357)

[Додаток Б.1 «Програмний код основного модуля» 19](#_Toc162355358)

[Додаток Б.2 «Програмний код функції для виведення вмісту файлу » 20](#_Toc162355359)

[Додаток Б.3 «Програмний код функції , що обчислює рядки та стовпчики» 20](#_Toc162355360)

[Додаток Б.4 «Програмний код функції, що створює динамічний двомірний масив» 21](#_Toc162355361)

[Додаток Б.5 «Програмний код функції, що виводить створену матрицю» 21](#_Toc162355362)

[Додаток Б.6 «Програмний код функції, що обчислює кількість елементів більших за 1» 22](#_Toc162355363)

[Додаток Б.7 «Програмний код функції , що обчислює кількість локальних мінімумів матриці» 22](#_Toc162355364)

[Додаток Б.8 «Програмний код функції для обчислення вектора за сумою від'ємних елементів стовпців» 23](#_Toc162355365)

[Додаток Б.9 «Програмний код функції, створює динамічний масив з ненульових елементів» 24](#_Toc162355366)

[Додаток Б.10 «Програмний код функції, що розташовує рядки матриці сортуючи за квадратом суми» 24](#_Toc162355367)

[Додаток Б.11 «Програмний код функції, що визначає, чи є у рядках цифри» 25](#_Toc162355368)

[Додаток Б.12 «Програмний код функції, що сортує слова за алфавітом в кожному рядку» 26](#_Toc162355369)

[Додаток Б.13 «Програмний код функції, що замінює введений рядок на введений текст» 27](#_Toc162355370)

[Додаток Б.14 «Програмний код функції, що замінює введений текст у рядку іншим введеним текстом» 28](#_Toc162355371)

[Додаток Б.15 «Програмний код функції, що видаляє введений рядок» 29](#_Toc162355372)

[Додаток Б.16 «Програмний код функції, що видаляє рядок з введеним текстом» 30](#_Toc162355373)

# 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 аргумента.

[Додаток Б.1 «Програмний код основного модуля»](#B1)

Підключення бібліотек для використання:

#include <iostream>

#include <cstring>

#include <map>

#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 масива 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.

# 3 Результати

# 4 Інструкція з використання програми

# 5 Висновки з аналізом досягнутих результатів

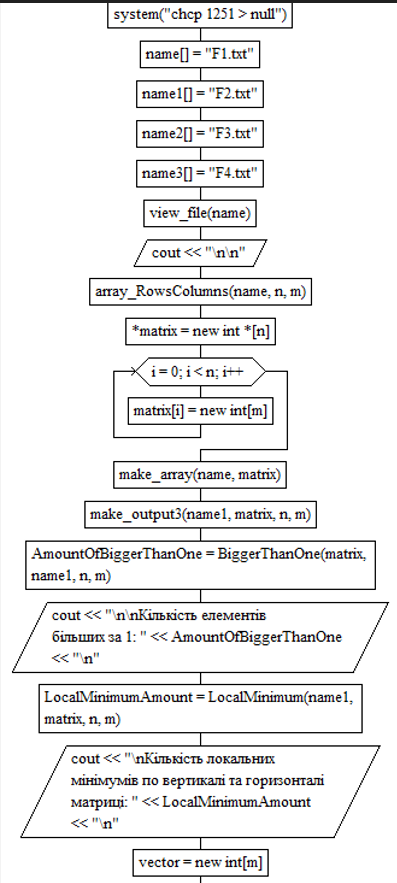
# Список використаних джерел

# додатки

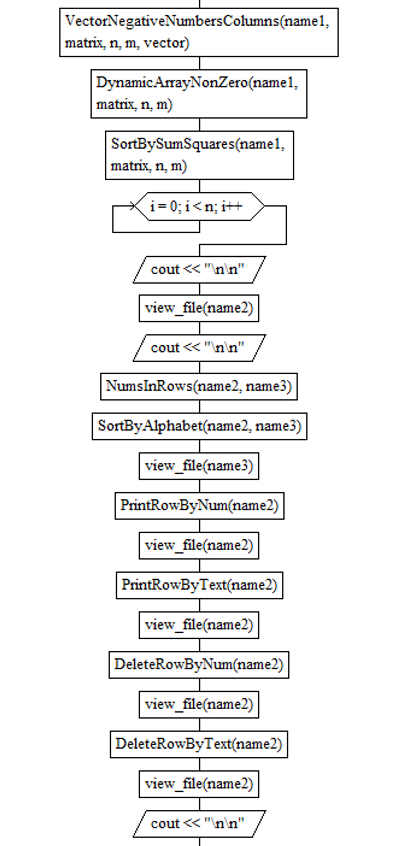
## Додаток А. Схеми алгоритмів функцій програмного проекту

### Додаток A.1 «Блок схема основного модуля»

Початок



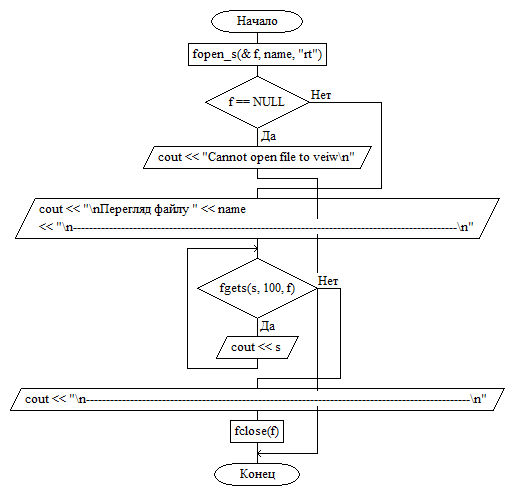
1



1

Кінець

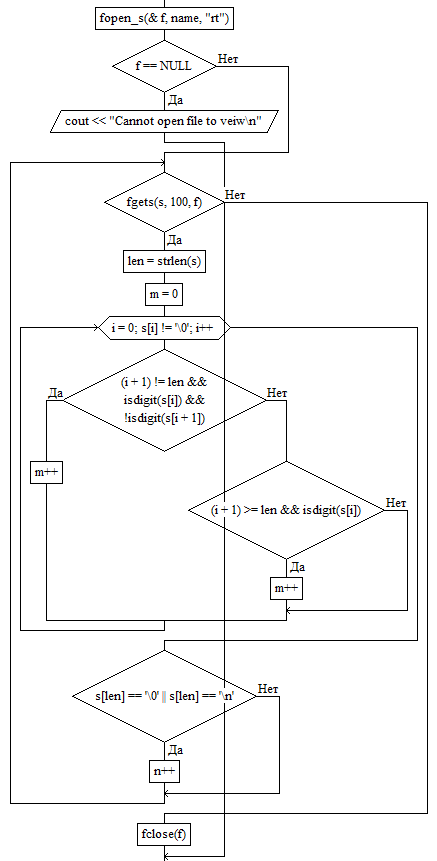
### Додаток A.2 «Блок схема для коду з додатку Б.2»



Початок

Кінець

### Додаток A.3 «Блок схема для коду з додатку Б.3»



Початок

Кінець

## Додаток Б «Програмний код»

### Додаток Б.1 «Програмний код основного модуля»

// CoursworkMain

#include <iostream>

#include <cstring>

#include <map>

#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]; // Створення динамічного

for (int i = 0; i < n; i++) // двомірного масиву

matrix[i] = new int[m]; // за допомогою new та (n,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]; // Створення динамічного масиву для вектора

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

delete[] vector; // Видалення динамічного масиву вектора для звільнення використанної пам'яті

DynamicArrayNonZero(name1, matrix, n, m);

SortBySumSquares(name1, matrix, n, m);

for (int i = 0; i < n; i++) // Видалення динамічного масиву матриці для звільнення використанної пам'яті

delete[] matrix[i];

delete[] matrix;

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 Б.2 «Програмний код функції для виведення вмісту файлу »

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

{

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);

}

### Додаток Б.3 «Програмний код функції , що обчислює рядки та стовпчики»

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

{

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++;

}

if (s[len] == '\0' || s[len] == '\n')

n++;

}

fclose(f);

}

### Додаток Б.4 «Програмний код функції, що створює динамічний двомірний масив»

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;

}

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);

}

### Додаток Б.5 «Програмний код функції, що виводить створену матрицю»

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;

}

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);

}

### Додаток Б.6 «Програмний код функції, що обчислює кількість елементів більших за 1»

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;

}

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;

}

### Додаток Б.7 «Програмний код функції , що обчислює кількість локальних мінімумів матриці»

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

{

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 локальних мінімумів: ");

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

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

{

if (i == 0 && j == 0)

{

if (matrix[i][j] < matrix[i + 1][j] && matrix[i][j] < matrix[i][j + 1])

amount++;

}

else if (i == 0 && j > 0 && j < m - 1)

{

if (matrix[i][j] < matrix[i + 1][j] && matrix[i][j] < matrix[i][j - 1] && matrix[i][j] < matrix[i][j + 1])

amount++;

}

else if (i == 0 && j == m - 1)

{

if (matrix[i][j] < matrix[i + 1][j] && matrix[i][j] < matrix[i][j - 1])

amount++;

}

else if (i > 0 && i < n - 1 && j == 0)

{

if (matrix[i][j] < matrix[i + 1][j] && matrix[i][j] < matrix[i - 1][j] && matrix[i][j] < matrix[i][j + 1])

amount++;

}

else if (i == n - 1 && j == 0)

{

if (matrix[i][j] < matrix[i - 1][j] && matrix[i][j] < matrix[i][j + 1])

amount++;

}

else if (i == n - 1 && j > 0 && j < m - 1)

{

if (matrix[i][j] < matrix[i][j - 1] && matrix[i][j] < matrix[i - 1][j] && matrix[i][j] < matrix[i][j + 1])

amount++;

}

else if (i == n - 1 && j == m - 1)

{

if (matrix[i][j] < matrix[i - 1][j] && matrix[i][j] < matrix[i][j - 1])

amount++;

}

else if (i > 0 && i < n - 1 && j == m - 1)

{

if (matrix[i][j] < matrix[i + 1][j] && matrix[i][j] < matrix[i - 1][j] && matrix[i][j] < matrix[i][j - 1])

amount++;

}

else if (matrix[i][j] < matrix[i - 1][j] && matrix[i][j] < matrix[i + 1][j] && matrix[i][j] < matrix[i][j - 1] && matrix[i][j] < matrix[i][j + 1])

amount++;

}

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

fclose(f);

return amount;

}

### Додаток Б.8 «Програмний код функції для обчислення вектора за сумою від'ємних елементів стовпців»

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;

}

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);

}

### Додаток Б.9 «Програмний код функції, створює динамічний масив з ненульових елементів»

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;

}

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);

}

### Додаток Б.10 «Програмний код функції, що розташовує рядки матриці сортуючи за квадратом суми»

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;

}

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 \*= sum;

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++;

}

fclose(f);

}

### Додаток Б.11 «Програмний код функції, що визначає, чи є у рядках цифри»

void NumsInRows(char\* name2, char\* name3) //функція визначає, чи є у рядках цифри, 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;

}

//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);

}

### Додаток Б.12 «Програмний код функції, що сортує слова за алфавітом в кожному рядку»

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;

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;

}

//cout << "Відсортовані рядки за алфавітом:\n\n";

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++)

{

//cout << it->first << ' ';

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

}

it++;

memset(word, 0, 50);

}

words.clear();

//cout << "\n";

fprintf(f2, "\n");

}

cout << "\n";

fclose(f1);

fclose(f2);

}

### 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");

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 InputText[200];

char SearchText[200];

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

cin.getline(SearchText, 200);

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

cin.getline(InputText, 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");

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;

int size = 0;

compared = true;

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

{

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

{

compared = false;

break;

}

n++;

size++;

}

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

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

if (compared)

{

int t;

if (j != 0)

{

for (n = 0, t = 0; n < j; n++)

{

tmpstart[n] = it->second[t++];

}

}

for (n = 0, t = j + size; it->second[t] != '\0'; n++)

{

tmpend[n] = it->second[t++];

}

//tmpend[n] = '\0';

strcat\_s(tmpstart, 200, InputText);

strcat\_s(tmpstart, 200, tmpend);

strcpy\_s(it->second, 200, tmpstart);

}

}

}

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

it++;

}

fclose(f2);

}

### 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");

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");

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);

}