**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»**

студентки 1 курсу групи ІПЗ-111

спеціальності 121 «Інженерія програмного забезпечення»

Ярош Євгенії В’ячеславівни

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 – 2022 р.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»**

**ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра інформаційних технологій**

# ЗАВДАННЯ

з дисципліни «Алгоритмізація та програмування»

студентки Ярош Євгенії В’ячеславівни

спеціальності 121 Інженерія програмного забезпечення

першого курсу групи ІПЗ-111

Тема «Програмне опрацювання даних у файлах»

Варіант 12

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 F3.txt з коректними текстовими даними.

Курсова робота виконується в такому обсязі:

1. Розрахунково-пояснювальна записка:

− аналіз засобів та методів розробки;

− словесний опис алгоритмів основного модуля;

− аналіз досягнутих результатів;

− інструкція з використання програми.

1. Графічна частина:

− схеми алгоритмів основного модуля для виконання завдань.

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 додатних елементів і кількість нульових показників вимірювань датчиків (елементів матриці);
* обчислити середнє арифметичне вимірювання (стовпця матриці), який містить мінімальне значення матриці. У разі декількох однакових мінімальних значень, брати перший з них;
* обчислити елементи вектора як суми показників датчиків (елементів рядків матриці) з парними індексами вимірювань (стовпців);
* створити динамічний масив з чисел, розміщених після першого двоцифрового числа матриці.

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

1. Розробити алгоритми та написати програмний код для того, щоб вивести індекси всіх ком та крапок, які є у рядках і вивести слова файлу з довжиною більше семи символів та обчислити їхню кількість у файлі F3.txt. Програмно вивести кожний рядок з результатом на екран та записати у файл F2.txt.
2. Розробити алгоритми та написати програмний код для того, щоб відредагувати рядок заданий номером та рядки, які містять заданий текст у файлі F3.txt і переглянути його вміст.
3. Розробити алгоритми та написати програмний код для того, щоб видалити з файлу F3.txt рядок, заданий номером і рядки, що містять заданий текст та переглянути його вміст.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |
| --- | --- | --- |
| № | Завдання | Термін виконання |
| 1 | Ознайомлення з завданням до курсової роботи.  Підготування вхідних файлів | 21.02.2022-25.02.2022 |
| 2 | Частина 1. Опрацювання матриць | 01.03.2022-11.03.2022 |
| 3 | Частина 2. Опрацювання динамічних масивів | 14.03.2022-24.03.2022 |
| 4 | Частина 3. Опрацювання рядкових даних | 25.03.2022-04.04.2022 |
| 5 | Частина 4. Опрацювання текстових файлів | 06.04.2022-15.04.2022 |
| 6 | Оформлення пояснювальної записки | 16.04.2022-27.04.2022 |

# ЗМІСТ

[ЗАВДАННЯ 2](#_Toc100775971)

[ВСТУП 5](#_Toc100775973)

[1 АНАЛІЗ ЗАСОБІВ ТА МЕТОДІВ РОЗРОБКИ 7](#_Toc100775974)

[2 СЛОВЕСНИЙ ОПИС АЛГОРИТМІВ ДЛЯ РОЗВ’ЯЗУВАННЯ ЗАДАЧ 10](#_Toc100775975)

[2.1 Опис функції *view\_file()* 10](#_Toc100775976)

[2.2 Опис функції *create\_matrix()* 11](#_Toc100775977)

[2.3 Опис функції *output\_matrix()* 12](#_Toc100775978)

[2.4 Опис функції *amount()* 13](#_Toc100775979)

[2.5 Опис функції *obchusleny()* 14](#_Toc100775980)

[2.6 Опис функції *create\_vector()* 16](#_Toc100775981)

[2.7 Опис функції *create\_masiv()* 17](#_Toc100775982)

[2.8 Опис функції *indexsu()* 19](#_Toc100775983)

[2.9 Опис функції *slova()* 20](#_Toc100775984)

[2.10 Опис функції *redaguvany()* 22](#_Toc100775985)

[2.11 Опис функції *vudaleny()* 25](#_Toc100775986)

[2.12 Опис функції *main()* 27](#_Toc100775987)

[3 РЕЗУЛЬТАТИ 29](#_Toc100775988)

[4 ІНСТРУКЦІЯ З ВИКОРИСТАННЯ ПРОГРАМИ 34](#_Toc100775989)

[ВИСНОВКИ З АНАЛІЗОМ ДОСЯГНУТИХ РЕЗУЛЬТАТІВ 38](#_Toc100775990)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 39](#_Toc100775991)

[ДОДАТКИ 40](#_Toc100775992)

[Додаток А Схеми алгоритмів функцій програмного проєкту 40](#_Toc100775993)

[Додаток Б Програмний код 47](#_Toc100775994)

# ВСТУП

У цій роботі на конкретних прикладах розглядаються потужні можливості мови програмування високого рівня – С. Ця мова має широкий спектр засобів. Мова C пропонує широкий спектр засобів для створення бінарних (двійкових) і текстових файлів, а також опрацювання цих файлів.

Актуальність теми курсової роботи «Програмне опрацювання даних у файлах» спричинена широкою практичною затребуваністю вмінь і навичок з розробки програмних проєктів, які обробляють і зберігають дані у файлах. Інакше кажучи, програма має сама створювати файли різних форматів, записувати в них дані і мати здатність вивести, редагувати, видаляти всі або деякі дані за умовою або потреби, давати змогу створити нові файли з даними з головного файлу.

Об'єктом дослідження є стандартні технології програмування алгоритмічною мовою високого рівня і системи створення програмних проєктів.

Предмет дослідження – програмні засоби мови С для оброблення неоднорідних даних, створення різноформатних файлів, запису даних у ці файли, виведення їх вмісту, відбір даних за потребою, здатністю створення нових файлів з головного файлу.

Мета курсової роботи полягає у:

1. узагальненні, закріпленні та розширенні практичних і теоретичних знань, умінь і навичок за фахом та їх використанні при вирішенні конкретних завдань у галузі програмування;
2. покращенні умінь пошуку необхідної інформації, здобутті досвіду в дослідженні й експериментуванні під час розв’язування задач, які потрібно виконати в курсовій роботі;
3. закріпленні знань алгоритмізації та навичок програмування мовою високого рівня;
4. розкритті підготовки до самостійної роботи.

Відповідно до поставленої мети:

* розкрити сутність засобів мови С з опрацювання неоднорідних даних у файлах;
* опрацювати наявні сучасні програмні засоби, які дозволяють створювати програмні проєкти мовою C;
* вивчити засоби мови С з програмного створення різноформатних файлів, запис даних у них, перегляду вмісту цих файлів, створення динамічного масиву, редагування та видалення даних у файлах (для задач підвищеного рівня складності);
* розробити програмний проєкт для розв’язання завдання відповідно до індивідуального варіанта, який передбачає створення, заповнення даними, обчислення, виведення даних, створення динамічного масиву, редагування та видалення вмісту різноформатних файлів.

Курсова робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. Загальний обсяг курсової роботи: 54 с., 24 рис., 3 табл., 2 додатки, 7 джерел.

# 1 АНАЛІЗ ЗАСОБІВ ТА МЕТОДІВ РОЗРОБКИ

Для того, щоб мати змогу працювати з програмою необхідно підключити певні стандартні бібліотеки. Це можливо зробити за допомогою директиви #include.

1. iostream – бібліотека необхідна для введення і виведення даних в мові програмування С.
2. stdlib.h – стандартна бібліотека мови С, яка містить функції для виділення пам’яті, контролю процесу виконання програми та ін.
3. stdio.h – стандартна бібліотека мови С, яка містить визначення констант та оголошення функцій і типів, призначених для виконання вводу і виводу.
4. string.h – стандартна бібліотека мови С, яка містить функції призначені для роботи з пам’яттю і нуль-термінованими рядками [1].

Для того, щоб коректно відображалися слова українською мовою, потрібно додати бібліотеку windows.h – бібліотека використовується для підключення функціоналу в програмі, надані операційною системою (Windows 95, 98,NT, 2000, XP) [2].

Щоб не виникали помилки компіляції у Visual Studio, треба долучити директиву #pragma warning(disable : 4996) [3].

Якщо потрібно, щоб компілятор використовував все, що є в просторі імен std треба прописати такий рядок – using namespace std [4].

В програмі використовуємо:

* функція fopen() для того, щоб відкрити файл;
* умовний оператор if –задає умову і виконується, якщо ця умова істинна;
* функція puts() для виведення рядку;
* функція print() виводить на екран символи або значення;
* оператори циклу for, while;
* оператор return – вихід із функції або для повернення значень;
* функція fgets() для зчитування даних;
* функція fprintf() виводить в файл символи і значення;
* функція strlen() рахує кількість символів до термінованого нуля;
* функція strtok() виконує пошук лексем;
* функція atof() перетворює рядок в значення типу double;
* функція fclose() для того, щоб закрити файл;
* функція *fseek()* скидає прапорець кінця файлу, пов'язаний із зазначеним потоком;
* функція *getchar()* повертає наступний символ стандартного потоку введення (stdin);
* функція *strcpy()* для копіювання нуль-термінованого рядка (включно з нуль-термінатором) у заданий буфер;
* функція *gets\_s()* зчитує символи з stdin і поміщає в масив символів, який вказує str;
* оператор *cout* – для виведення даних на екран;
* оператор *cin* – для введення даних;
* інверсія(НЕ – !), кон’юкція(І – &&), диз’юнкція(АБО – ||);
* ескейп-послідовності – спеціальна комбінація символів, яка розпочинається зі зворотної скісної риси і записується в одинарних лапках (табл. 1.1), наприклад: '\0', '\n'. Кожна з наведених у таблиці комбінацій символів вважається за один символ [5].

Таблиця 1.1 – Деякі поширені у застосуванні ескейп-послідовності

|  |  |
| --- | --- |
| Символьне подання | Опис |
| \r | повернення курсора на початок рядка |
| \n | новий рядок |
| \t | горизонтальна табуляція |
| \v | вертикальна табуляція |
| \” | подвійні лапки |
| \’ | апостроф |
| \\ | зворотний слеш |
| \0 | нульовий символ |
| \? | знак питання |

Одним із аргументів функції fopen() є специфікатор режиму відкривання файлу. В таблиці 1.2 наведені такі специфікатори та їх опис [6].

Таблиця 1.2 – Специфікатори режиму відкривання файлів

|  |  |
| --- | --- |
| Параметр | Опис |
| r | відкрити файл лише для зчитування даних |
| w | створити файл для записування даних |
| a | відкрити чи створити файл для записування даних у кінець файлу |
| r+ | відкрити файл для зчитування й записування даних |

Закінчення табл. 1.2

|  |  |
| --- | --- |
| Параметр | Опис |
| w+ | створити файл для зчитування і записування даних |
| a+ | відкрити чи створити файл для зчитування чи то записування даних у кінець файлу |
| rt | відкрити текстовий файл для зчитування даних |
| wt | створити текстовий файл для записування даних |
| at | відкрити чи створити текстовий файл для зчитування чи то записування даних у кінець файлу |

Рядок форматування параметра format будується з послідовності символів-специфікаторів типів читаних даних, найпоширеніші з яких наведено у таблиці 1.3 [7].

Таблиця 1.3 – Специфікатори параметру format

|  |  |  |
| --- | --- | --- |
| Символ | Значення, що вводиться | Тип аргументу |
| i, l | десяткове, вісімкове чи шістнадцяткове ціле число | int, long |
| d, D | десяткове ціле число | int, long |
| e, E | дійсне число з рухомою крапкою | float |
| f, F | дійсне число з фіксованою крапкою | float |
| s | рядок символів | char s[] |
| c | символ | char |

# 2 СЛОВЕСНИЙ ОПИС АЛГОРИТМІВ ДЛЯ РОЗВ’ЯЗУВАННЯ ЗАДАЧ

Підключаємо стандартні бібліотеки:

#include <iostream>

#include < stdlib.h >

#include <string.h>

#include <windows.h>

Директива для відключення помилки:

#pragma warning(disable : 4996)

Using-директива:

using namespace std;

Розмірність масиву задаємо константами:

const int M = 5, N = 8;

## 2.1 Опис функції *view\_file()*

Функція *view\_file()* здійснює перегляд вказаного файлу. У програмному коді вона використовується тричі, але з різними аргументами.

Аргументом функції є рядкова змінна *name* – назва файлу з текстовими даними, створеного раніше у Блокноті для подальшої роботи.

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *s* – рядок, що має довжину до 100 символів для зчитування тексту із файлу;
* *f* – файлова змінна, яка здійснює доступ до файлу.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f = fopen(name, "rt");*

Функція *fopen()* використовується для відкриття файлу. Перший аргумент *name* – ім’я файлу, який потрібно відкрити; другий аргумент *"rt"* – специфікатор режиму відкривання файлу.

*"rt"* – відкривається лише для зчитування даних.

1. Перевірка чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f == NULL) { puts("Cannot open file to view"); return; }*

1. Виведення повідомлення про перегляд файлу на екран:

*printf("\nПерегляд файлу: %s\n", name);*

1. За допомогою циклу *while* послідовно зчитувати рядки з файлу у s, до того моменту, поки вони є у файлі(до кінця файлу):

*while (fgets(s, 100, f))*

Черговий рядок зчитується з файлу за допомогою функції *fgets(s, 100, f)*

Функція має три аргументи:

* *s* – рядкова змінна, в яку записується рядок, зчитаний із файлу;
* *100* – максимальна кількість символів, які можуть бути зчитані за один раз;
* *f* – файл, з якого зчитуються дані.

1. Зчитувати рядок s доки він не закінчиться за допомогою функції strlen та видалити останній символ (інакше після кожного рядка буде виведено порожній):

*s[strlen(s) – 1] = '\0';*

1. Виводиться результат – рядок *s* на екран:

*puts(s);*

1. Закрити файл після закінчення роботи з ним:

*fclose(f);*

## 2.2 Опис функції *create\_matrix()*

Функція *create\_matrix()* формує числову матрицю зчитуванням її значень із вказаного файлу.

Аргументом функції є матриця *matr[M][N]* та рядкова змінна *name*.

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *f* – файлова змінна, яка здійснює доступ до файлу;
* *s* – рядок, що має довжину до 100 символів для зчитування тексту із файлу;
* *t* – рядок, що має довжину до 100 символів для заповнення матриці;
* *i* – індекс кількості рядків матриці;
* *j* – індекс кількості стовпців матриці.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f = fopen(name, "rt");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name* – ім’я файлу, який потрібно відкрити; другий аргумент *"rt"* – специфікатор режиму відкривання файлу.

*"rt"* – файл відкривається лише для зчитування даних.

1. Перевірка чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f == NULL) { puts("Cannot open file to view"); return; }*

1. За допомогою циклу *while* послідовно зчитувати з файлу рядки у *s*, до того моменту, поки вони є у файлі(до кінця файлу):

*while (fgets(s, 100, f))*

Черговий рядок зчитується з файлу за допомогою функції *fgets(s, 100, f)*

Функція має три аргументи:

* *s* – рядкова змінна, в яку записується рядок, зчитаний із файлу;
* *100* – максимальна кількість символів, які можуть бути зчитані за один раз;
* *f* – файл, з якого зчитуються дані.

1. Для заповнення матриці обнуляємо індекс *j*:

*j = 0;*

1. Розділити рядок *s* на лексеми і записати його у рядок *t*:

*t = strtok(s, "\t");*

1. За допомогою циклу *while*, зчитувати кожний рядок доки не закінчилися дані:

*while (t != NULL)*

1. Присвоїти кожному значенню елемента матриці значення типу *double* за допомогою функції *atof()*:

*matr[i][j] = atof(t)*;

1. Використати функцію strtok() для розміщення елементів:

*t = strtok(NULL, "\t");*

1. Збільшити індекс на одиницю для внутрішнього циклу: *j++;*
2. Збільшити індекс на одиницю для зовнішнього циклу: *i++;*
3. Закрити файл після закінчення роботи з ним:

*fclose(f);*

## 2.3 Опис функції *output\_matrix()*

Функція *output\_matrix()* виводить матрицю на екран та у файл.

Аргументом функції є матриця *matr[M][N]* та рядкова зміна *name2.*

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *f2* – файлова зміна, яка здійснює доступ до файлу;
* *i* – індекс кількості рядків матриці;
* *j* – індекс кількості стовпців матриці.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f2 = fopen(name2, "wt");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name2* – ім’я файлу, який потрібно відкрити; другий аргумент *"wt"* – специфікатор режиму відкривання файлу.

*"wt"* – створення файлу для записування даних.

1. Перевірка чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f2 == NULL) { puts("Cannot open file"); return; }*

1. Виведення повідомлення про перегляд файлу на екран та у файл:

*printf("\n\nПерегляд матрицi:\n");*

*fprintf(f2, "Перегляд матрицi:\n");*

1. За допомогою вкладених циклів *for* вивести елементи матриці на екран і записати їх у файл:

*for (i = 0; i < M; i++)*

*for (j = 0; j < N; j++)*

*printf("%*7.1*f\t", matr[i][j]);*

*fprintf(f2, "%7.1f\t", matr[i][j]);*

1. Після виведення всіх елементів поточного рядка перехід на наступний рядок (тобто після внутрішнього циклу) на екрані і у файлі:

*puts("");*

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

1. Закрити файл після закінчення роботи з ним:

*fclose(f2);*

## 2.4 Опис функції *amount()*

Функція *amount()* обчислює кількість додатніх і кількість нульових елементів матриці і записує результат у файл *F2.txt* та виводить на екран.

Аргументом функції є матриця *matr[M][N]* та рядкова зміна *name2.*

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *f2* – файлова зміна, яка здійснює доступ до файлу;
* *i* – індекс кількості рядків матриці;
* *j* – індекс кількості стовпців матриці;
* *d=0*, *n=0* – змінні для вирішення задачі.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f2 = fopen(name2, "at");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name2* – ім’я файлу, який потрібно відкрити; другий аргумент *"at"* – специфікатор режиму відкривання файлу.

*"at"* – файл відкривається для зчитування і записування даних у кінець файлу.

1. Перевірка чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f2 == NULL) { puts("Cannot open file to append data\n"); return; }*

1. За допомогою вкладених циклів *for* створюємо масив:

*for (i = 0; i < M; i++)*

*for (j = 0; j < N; j++)*

1. За допомогою команти *if* перевіряємо чи є елемент додатнім:

*if (matr[i][j] >= 0)*

1. Якщо так, то збільшуємо змінну *d* на 1:

*d++;*

1. За допомогою команди *if* перевіряємо чи дорівнює елемент нулю:

*if (matr[i][j] == 0)*

1. Якщо так, то збільшуємо змінну *n* на 1:

*n++;*

1. Виведення результатів на екран та у файл:

*printf("\nКількість додатніх елементів: %i\t", d);*

*fprintf(f2, "\nКількість додатніх елементів: %i\n", d);*

*printf("\nКількість 0 показників: %i\t", n);*

*fprintf(f2, "\nКількість 0 показників: %i\n", n);*

1. Закрити файл після закінчення роботи з ним:

*fclose(f2);*

## 2.5 Опис функції *obchusleny()*

Функція *obchusleny()* обчислює середнє арифметичне стовпця матриці, який містить мінімальне значення матриці і записує результат у файл *F2.txt* та виводить на екран.

Аргументом функції є матриця *matr[M][N]* та рядкова зміна *name2.*

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *f2* – файлова зміна, яка здійснює доступ до файлу;
* *i* – індекс кількості рядків матриці;
* *j* – індекс кількості стовпців матриці;
* *sum=0, min=0, ser=0, kol=0, h=0*– змінні для вирішення задачі.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f2 = fopen(name2, "at");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name2* – ім’я файлу, який потрібно відкрити; другий аргумент *"at"* – специфікатор режиму відкривання файлу.

*"at"* – файл відкривається для зчитування і записування даних у кінець файлу.

1. Перевірка чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f2 == NULL) { puts("Cannot open file to append data\n"); return; }*

1. Виведення повідомлення про середнє арифметичне на екран та у файл:

*puts("\nСер.арифметичне:");*

*fprintf(f2, "\nСер.арифметичне:\n");*

1. За допомогою вкладених циклів *for* створюємо масив:

*for (i = 0; i < M; i++)*

*for (j = 0; j < N; j++)*

1. За допомогою команти *if* перевіряємо чи менший елемент за мінімальне значення:

*if (matr[i][j] < min)*

1. Якщо так, то присвоюємо мінімальному значенню, значення елемента матриці. Також присвоюємо змінній *h* значення стовбця матриці в якому знаходиться мінімальний елемент:

*min=matr[i][j];*

*h = j;*

1. За допомогою циклу *for*:

*for (i = 0; i < M; i++)*

1. Рахуємо кількість елементів у стовбці (збільшуємо *kol* на 1), де міститься мінімальний елемент, та рахуємо їх суму:

*kol++;*

*sum += matr[i][h];*

1. Знаходимо середнє арифметичне стовбця матриці з мінімальним елементом, шляхом ділення суми на кількість елементів:

*ser = sum/kol;*

1. Виведення результатів на екран та у файл:

*printf("%7.1f\t", ser);*

*fprintf(f2, "%7.1f\t", ser);*

1. Закрити файл після закінчення роботи з ним:

*fclose(f2);*

## 2.6 Опис функції *create\_vector()*

Функція *create\_vector()* обчислює елементи вектора як суми елементів рядків матриці з парними індексами стовпців і записує результат у файл *F2.txt* та на екран.

Аргументами функції є матриця *matr[M][N]*, вектор *vekt[M]* і рядкова змінна *name2*.

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *f2* – файлова змінна, яка здійснює доступ до файлу;
* *i* – індекс кількості рядків матриці;
* *j* – індекс кількості стовпців матриці;
* *sum=0* – змінна для вирішення задачі.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f2 = fopen(name2, "at");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name2* – ім’я файлу, який потрібно відкрити; другий аргумент *"at"* – специфікатор режиму відкривання файлу.

*"at"* – файл відкривається для зчитування і записування даних у кінець файлу.

1. Перевірка чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f2 == NULL) { puts("Cannot open file to append data\n"); return; }*

1. Виведення повідомлення про характер розв’язуваної задачі на екран та у файл:

*puts("\nСтворений вектор:");*

*fprintf(f2, "\nСтворений вектор:\n");*

1. За допомогою зовнішнього циклу *for*:

*for (int i = 0; i < M; i++)*

1. Обнуляємо суму:

*sum = 0;*

1. За допомогою внутрішнього циклу *for:*

*for (int j = 1; j < N; j += 2)*

1. Рахуємо суму елементів рядків матриці з парними індексами стовбців матриці:

*sum += matr[i][j];*

1. Створюємо вектор з суми елементів рядків матриці з парними індексами стовбців матриці:

*vekt[i] = matr[i][j];*

1. Виведення результатів на екран та у файл:

*printf("%7.1f\t", vekt[i]);*

*fprintf(f2, "%7.1f\t", vekt[i]);*

1. Закрити файл після закінчення роботи з ним.

*fclose(f2);*

## 2.7 Опис функції *create\_masiv()*

Функція *create\_masiv()* створює динамічний масив з чисел, розміщених після першого двоцифрового числа і записує результат у файл F2.txt та виводить на екран.

Аргументами функції є матриця *matr[M][N]* і рядкова змінна *name2*.

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *f2* – файлова змінна, яка здійснює доступ до файлу;
* *col=0*, *k, h, g=0* – змінні для вирішення задачі;
* *bool p = false* – змінна для перевірки істинності;
* *i* – індекс кількості рядків матриці;
* *j* – індекс кількості стовпців матриці.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f2 = fopen(name2, "at");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name2* – ім’я файлу, який потрібно відкрити; другий аргумент *"at"* – специфікатор режиму відкривання файлу.

*"at"* – файл відкривається для зчитування і записування даних у кінець файлу.

1. Перевірка чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f2 == NULL) { puts("Cannot open file to append data\n"); return; }*

1. Виведення повідомлення про характер розв’язуваної задачі на екран та у файл:

*puts("\nСтворений динамiчний масив:");*

*fprintf(f2, "\nСтворений динамiчний масив:\n");*

1. За допомогою вкладених циклів *for* створюємо двовимірний масив:

*for (i = 0; i < M; i++)*

*for (j = 0; j < N; j++)*

1. За допомогою оператору умови *if* перевіряємо чи є елемент двоцифровим і чи виконується умова НЕ *p*:

*if (abs(matr[i][j]) > 9 && abs(matr[i][j]) < 100 && !p)*

1. Якщо так, то умова істинна, присвоюємо змінній *k* індекс рядка матриці, а змінній *h* присвоюємо індекс стовпця матриці:

*p = true;*

*k = i;*

*h = j;*

1. Якщо умова *p*, то збільшуємо змінну *count* на 1:

*if (p)*

*count++;*

1. Оголошуємо динамічний масив:

*double\* arr = new double[count];*

1. За допомогою зовнішнього циклу *for*:

*for (i = k, g = 0; i < M; i++)*

1. Перевіряємо, чи рядок дорівнює 0:

*if (g == 0)*

1. Якщо так, то присвоюємо стовпцю значення *h*, інакше 0:

*j = h;*

*else*

*j = 0;*

1. За допомогою внутрішнього циклу *for:*

*for (j; j < N; j++)*

1. Присвоюємо масиву значення елемента матриці і збільшуємо змінну *g*:

*arr[g++] = matr[i][j];*

1. За допомогою циклу *for* створюємо одновимірний масив для виводу результату:

*for (int i = 0; i < count; i++)*

1. Виведення результатів на екран та у файл:

*printf("%7.1f\t", arr[i]);*

*fprintf(f2, "%7.1f\t", arr[i]);*

1. Закрити файл після закінчення роботи з ним.

*fclose(f2);*

## 2.8 Опис функції *indexsu()*

Функція *indexsu()* виводить індекси всіх ком та крапок, які є у рядках у файлі з ім’ям *F3.txt* на екран та у файл з ім’ям *F2.txt.*

Аргументами функції є рядкова змінна *name3* – назва файлу з текстовими даними; *name2* – ім’я файлу з результатами.

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *s* – рядок, що має довжину до 100 символів для зчитування тексту із файлу;
* *f3*, *f2* – файлові змінні, за допомогою яких здійснюється доступ до файлів;
* *sum* – змінна для вирішення задачі;
* *i* – змінна, організована для формування циклу *for*.

Словесний алгоритм роботи функції:

1. Відкрити файли:

*f3 = fopen(name3, "rt");*

*f2 = fopen(name2, "at");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name3* або *name2* – ім’я файлу, який потрібно відкрити; другий аргумент *"rt"* або *"at"* – специфікатор режиму відкривання файлу.

*"rt"* – файл відкривається для зчитування даних;

*"at"* – файл відкривається для зчитування і записування даних у кінець файлу.

1. Перевірити чи відкриваються файли. Якщо файли не відкриваються, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f3 == NULL) { puts("Cannot open file to view"); return; }*

*if (f2 == NULL) { puts("Cannot open file to append\n"); return; }*

1. Виведення повідомлення про характер задачі на екран та у файл:

*puts("\nІндекси всіх ком та крапок:");*

*fputs("\nІндекси всіх ком та крапок:\n", f2);*

1. За допомогою циклу *while* послідовно зчитувати з файлу рядки у *s*, до того моменту, доки вони є у файлі(до кінця файлу):

*while (fgets(s, 100, f3))*

Черговий рядок зчитується з файлу за допомогою функції *fgets(s, 100, f3)*

Функція має три аргументи:

* *s* – рядкова змінна, в яку записується рядок, зчитаний із файлу;
* *100* – максимальна кількість символів, які можуть бути зчитані за раз;
* *f* – файл, з якого зчитуються дані.

1. Формуємо цикл *for* для зчитування даних:

*for (int i = 0; i < strlen(s); i++)*

1. Перевіряємо рядок на наявність ком або крапок:

*if (s[i] == ',' || s[i] == '.')*

1. Якщо якийсь зі знаків присутній, то виводимо його індекс, та додаємо суму символів попереднього рядка:

*printf("%i\t", i+sum);*

*fprintf(f2, "%i\t", i+sum);*

1. Додаємо до суми, довжину цього рядка:

*sum += strlen(s);*

1. Закриваємо файли після закінчення роботи з ними:

*fclose(f2);*

*fclose(f3);*

## 2.9 Опис функції *slova()*

Функція *slova* виводить на екран слова файлу з ім’ям *F3.txt* з довжиною більше семи символів та обчислює їхню кількість та записує їх у файл з ім’ям *F2.txt.*

Аргументами функції є рядкова змінна *name3* – назва файлу з текстовими даними; *name2* – ім’я файлу з результатами.

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *s* – рядок, що має довжину до 100 символів для зчитування тексту із файлу;
* *f2*, *f3* – файлові змінні, за допомогою яких здійснюється доступ до файлів;
* *k* – змінна для вирішення задачі;
* *t* – рядок довжиною до 100 символів для визначення лексем.

Словесний алгоритм роботи функції:

1. Відкрити файли:

*f3 = fopen(name3, "rt");*

*f2 = fopen(name2, "at");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name3* або *name2* – ім’я файлу, який потрібно відкрити; другий аргумент *"rt"* або *"at"* – специфікатор режиму відкривання файлу.

*"rt"* – файл відкривається для зчитування даних;

*"at"* – файл відкривається для зчитування і записування даних у кінець файлу.

1. Перевірка чи відкриваються файли. Якщо файли не відкриваються, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f3 == NULL) { puts("Cannot open file to view"); return; }*

*if (f2 == NULL) { puts("Cannot open file to append\n"); return; }*

1. Створити динамічний масив:

*\*t = new char [100];*

1. Виведення повідомлення про характер задачі на екран та у файл:

*puts("\nСлова з довжиною > 7 символів:");*

*fputs("\nСлова з довжиною > 7 символів:\n", f2);*

1. За допомогою циклу *while* послідовно зчитувати з файлу рядки у *s*, до того моменту, доки вони є у файлі(до кінця файлу):

*while (fgets(s, 100, f3))*

Черговий рядок зчитується з файлу за допомогою функції *fgets(s, 100, f3)*

Функція має три аргументи:

* *s* – рядкова змінна, в яку записується рядок, зчитаний із файлу;
* *100* – максимальна кількість символів, які можуть бути зчитані за один раз;
* *f* – файл, з якого зчитуються дані.

1. Розбити файл на лексеми. Розділювачі слів зазначено другим аргументом функції strtok:

*t = strtok (s, " ,.;:-^=!?");*

1. За допомогою циклу *while*, зчитувати кожний рядок доки не закінчилися дані:

*while (t != NULL)*

1. Перевіряємо чи довжина лексеми більше 7 символів:

*if (strlen(t) > 7)*

1. Якщо так, виконуємо наступні дії:

*t = strtok (s, " ,.;:-^=!?");*

*k++;*

1. Виведення на екран та у файл слів, довжина яких більше семи символів:

*puts(t);*

*fprintf(f2, "%s\n", t);*

1. Використати функцію *strok()* для розміщення слів:

*t = strtok(NULL, "\t");*

1. Виведення повідомлення про характер задачі на екран та у файл:

*puts("\nЇх кількість:");*

*fputs("\nЇх кількість:\n", f2);*

1. Виведення результатів на екран та у файл:

*printf(f2, "%i\t", k);*

*fprintf(f2, "%i\t", k);*

1. Закриваємо файли після закінчення роботи з ними:

*fclose(f2);*

*fclose(f3);*

## 2.10 Опис функції *redaguvany()*

Функція *redaguvany()* редагує текстовий файл з ім’ям *F3.txt* за номером рядка або по тексту.

Аргументами функції є рядкова змінна *name3* – назва файлу з текстовими даними.

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *s* – рядок, що має довжину до 100 символів для зчитування тексту із файлу;
* *s\_text* – рядок, що має довжину до 100 символів для копіювання тексту;
* *text* – рядок, що має довжину до 100 символів для введеня тексту;
* *f3* – файлова змінна, за допомогою якої здійснюється доступ до файлу;
* *choice = 0, num = 0, p = 0, n = 0* – змінні для вирішення задачі;
* *i* – змінна, організована для формування циклу *for*.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f3 = fopen(name3, "rt");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name3* – ім’я файлу, який потрібно відкрити; другий аргумент *"rt"* – специфікатор режиму відкривання файлу.

*"rt"* – файл відкривається для зчитування даних.

1. Перевірити чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f3 == NULL) { puts("Cannot open file to view"); return; }*

1. За допомогою циклу *while* послідовно зчитувати з файлу рядки у *s*, до того моменту, доки вони є у файлі(до кінця файлу) та збільшуємо змінну *p* на одиницю:

*while (fgets(s, 100, f3)) p++;*

Черговий рядок зчитується з файлу за допомогою функції *fgets(s, 100, f3)*

Функція має три аргументи:

* *s* – рядкова змінна, в яку записується рядок, зчитаний із файлу;
* *100* – максимальна кількість символів, які можуть бути зчитані за раз;
* *f* – файл, з якого зчитуються дані.

1. Створюємо динамічний масив:

*char\*\* temp = new char\*[p];*

1. Формуємо цикл *for* для зчитування даних:

*for (int i = 0; i < p; i++)*

1. Записуємо у тимчасовий масив:

*temp[i] = new char [100];*

1. Переміститися на початок файлу:

*fseek(f3,0,SEEK\_SET);*

1. За допомогою циклу *while* послідовно зчитувати з файлу рядки у *s*, до того моменту, доки вони є у файлі(до кінця файлу) та поки змінна *p>n*:

*while (fgets(s, 100, f3) && p>n)*

Черговий рядок зчитується з файлу за допомогою функції *fgets(s, 100, f3)*

Функція має три аргументи:

* *s* – рядкова змінна, в яку записується рядок, зчитаний із файлу;
* *100* – максимальна кількість символів, які можуть бути зчитані за раз;
* *f* – файл, з якого зчитуються дані.

1. Копіюємо рядки:

*s[strlen(s) – 1] = "\0";*

*strcpy(temp[n], s);*

*n++;*

1. Закриваємо файл після закінчення роботи з ним:

*fclose(f3);*

1. Відкрити файл:

*f3 = fopen(name3, "wt");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name3* – ім’я файлу, який потрібно відкрити; другий аргумент *"wt"* – специфікатор режиму відкривання файлу.

*"wt"* – файл відкривається для редагування даних.

1. Вибираємо, як саме будемо редагувати:

*cout << "\nДля редагування по номеру рядка, введіть: 1\n" << "Для редагування по тексту які вони містять, введіть: 2\n";*

*cin >> choice;*

1. Перевіряємо за допомогою *if* чи дорівнює змінна *choice* 1:

*if (choice == 1)*

1. Якщо так, то вводимо текст на який змінимо файл та номер рядка який потрібно редагувати:

*cout << "\nВведіть текст:\n";*

*getchar();*

*gets\_s(text,100);*

*cout << "\nВведіть номер рядка, який потрібно редагувати:\n";*

*cin >> num;*

1. Формуємо цикл *for* для зчитування даних:

*for (int i = 0; i < p; i++)*

1. Перевіряємо чи дорівнює номер рядка тому, який ми хочемо редагувати. Якщо ні, то виводимо скопійовані дані, якщо так то виводимо змінений текст:

*if ( i+1 != num)*

*fprintf (f3, "%s\n", temp [i]);*

*else fprintf (f3, "%s\n", text);*

1. Закриваємо файл після закінчення роботи з ним:

*fclose(f3);*

1. Якщо *choice* не дорівнює одиниці, то вводимо текст на який змінимо файл та текст для пошуку рядків з його вмістом для редагування:

*else*

*cout << "\nВведіть текст:\n";*

*getchar();*

*gets\_s(text,100);*

*cout << "\nВведіть текст для пошуків рядків з його вмістом для редагування:\n";*

*gets\_s(s\_text, 100);*

1. Формуємо цикл *for* для зчитування даних:

*for (int i = 0; i < p; i++)*

1. Перевіряємо чи не зустрівся нам рядок із заданим текстом, якщо ні то виводимо скопійовані дані, якщо так то виводимо нові записані дані:

*if (!strstr(temp[i], s\_text)) fprint (f3, "%s\n", temp[i]);*

*else fprintf (f3, "%s\n", text);*

1. Закриваємо файл після закінчення роботи з ним:

*fclose(f3);*

## 2.11 Опис функції *vudaleny()*

Функція *vudaleny()* видаляє з текстового файлу з ім’ям *F3.txt* за номером рядка або по тексту.

Аргументами функції є рядкова змінна *name3* – назва файлу з текстовими даними.

Функція має тип *void* – не повертає жодного значення засобами *return*.

Локальні змінні:

* *s* – рядок, що має довжину до 100 символів для зчитування тексту із файлу;
* *text* – рядок, що має довжину до 100 символів для введеня тексту;
* *f3* – файлова змінна, за допомогою якої здійснюється доступ до файлу;
* *choice = 0, num = 0, p = 0, n =0* – змінна для вирішення задачі;
* *i* – змінна, організована для формування циклу *for*.

Словесний алгоритм роботи функції:

1. Відкрити файл:

*f3 = fopen(name3, "rt");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name3* – ім’я файлу, який потрібно відкрити; другий аргумент *"rt"* – специфікатор режиму відкривання файлу.

*"rt"* – файл відкривається для зчитування даних.

1. Перевірити чи відкривається файл. Якщо файл не відкривається, то виводиться відповідне повідомлення і виконання програми переривається:

*if (f3 == NULL) { puts("Cannot open file to view"); return; }*

1. За допомогою циклу *while* послідовно зчитувати з файлу рядки у *s*, до того моменту, доки вони є у файлі(до кінця файлу) та збільшуємо змінну *p* на одиницю:

*while (fgets(s, 100, f3)) p++;*

Черговий рядок зчитується з файлу за допомогою функції *fgets(s, 100, f3)*

Функція має три аргументи:

* *s* – рядкова змінна, в яку записується рядок, зчитаний із файлу;
* *100* – максимальна кількість символів, які можуть бути зчитані за раз;
* *f* – файл, з якого зчитуються дані.

1. Створюємо динамічний масив:

*char\*\* temp = new char\*[p];*

1. Формуємо цикл *for* для зчитування даних:

*for (int i = 0; i < p; i++)*

1. Записуємо у тимчасовий масив:

*temp[i] = new char [100];*

1. Переміститися на початок файлу:

*fseek(f3,0,SEEK\_SET);*

1. За допомогою циклу *while* послідовно зчитувати з файлу рядки у *s*, до того моменту, доки вони є у файлі(до кінця файлу) та поки змінна *p>n*:

*while (fgets(s, 100, f3) && p>n)*

Черговий рядок зчитується з файлу за допомогою функції *fgets(s, 100, f3)*

Функція має три аргументи:

* *s* – рядкова змінна, в яку записується рядок, зчитаний із файлу;
* *100* – максимальна кількість символів, які можуть бути зчитані за раз;
* *f* – файл, з якого зчитуються дані.

1. Копіюємо рядки:

*s[strlen(s) – 1] = "\0";*

*strcpy(temp[n], s);*

*n++;*

1. Закриваємо файл після закінчення роботи з ним:

*fclose(f3);*

1. Відкрити файл:

*f3 = fopen(name3, "wt");*

Функція *fopen()* використовується для відкривання файлу. Перший аргумент *name3* – ім’я файлу, який потрібно відкрити; другий аргумент *"wt"* – специфікатор режиму відкривання файлу.

*"wt"* – файл відкривається для редагування даних.

1. Вибираємо, як саме будемо видаляти:

*cout << "\nДля видалення по номеру рядка, введіть: 1\n" << "Для видалення по тексту які вони містять, введіть: 2\n";*

*cin >> choice;*

1. Перевіряємо за допомогою *if* чи дорівнює змінна *choice* 1:

*if (choice == 1)*

1. Якщо так, то вводимо номер рядка який потрібно видалити:

*cin >> num;*

1. Формуємо цикл *for* для зчитування даних:

*for (int i = 0; i < p; i++)*

1. Перевіряємо чи дорівнює номер рядка тому, який ми хочемо видалити. Якщо ні, то виводимо скопійовані дані, якщо так то пропускаємо цей рядок:

*if ( i+1 != num)*

*fprintf( f3, "%s\n", temp [i]);*

1. Закриваємо файл після закінчення роботи з ним:

*fclose(f3);*

1. Якщо *choice* не дорівнює одиниці, то вводимо текст для пошуку рядків з його вмістом для видалення:

*else*

*cout << "\nВведіть текст для пошуку рядків з його вмістом для видалення:\n";*

*getchar();*

*gets\_s(text,100);*

1. Формуємо цикл *for* для зчитування даних:

*for (int i = 0; i < p; i++)*

1. Перевіряємо чи не зустрівся нам рядок із заданим текстом, якщо ні то виводимо скопійовані дані:

*if (!strstr(temp[i], s\_text)) fprint (f3, "%s\n", temp[i]);*

1. Закриваємо файл після закінчення роботи з ним:

*fclose(f3);*

## 2.12 Опис функції *main()*

int main()

{

//Використовується для підключення кирилиці

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

double A[M][N], B[N];

//Оголошення імен файлів

char name1[] = "E:\\F1.txt";

char name2[] = "E:\\F2.txt";

char name3[] = "E:\\F3.txt";

view\_file(name1); //Перегляд файлу

create\_matrix(A, name1); //Заповнити матрицю числами із файлу

output\_matrix(A, name2); //Вивести матрицю на екран та у файл

/\*Обчислити кількість додатніх елементів і кількість нульових показників вимірювань датчиків (елементів матриці)\*/

amount(A, name2);

/\*Обчислити середнє арифметичне вимірювання (стовпця матриці),який містить мінімальне значення матриці\*/

obchusleny(A, name2);

/\*Обчислити елементи вектора як суми елементів рядків матриці з парними індексами стовпців\*/

create\_vector(A, B, name2);

/\*Динамічний масив з чисел, розміщених після першого двоцифрового числа матриці\*/

create\_masiv(A, name2);

view\_file(name3); //Переглянути файл

//Вивести індекси всіх ком та крапок, які є у рядках

indexsu(name3, name2);

/\*Вивести слова файлу з довжиною більше 7 символів та обчислити їхню кількість\*/

slova(name3, name2);

view\_file(name2); //Переглянути файл

redaguvany(name3); //Редагувати файл F3.txt з текстовими даними

view\_file(name3); //Переглянути файл

vudaleny(name3); //Видалення рядків з файлу

view\_file(name3); //Переглянути файл

}

# 3 РЕЗУЛЬТАТИ

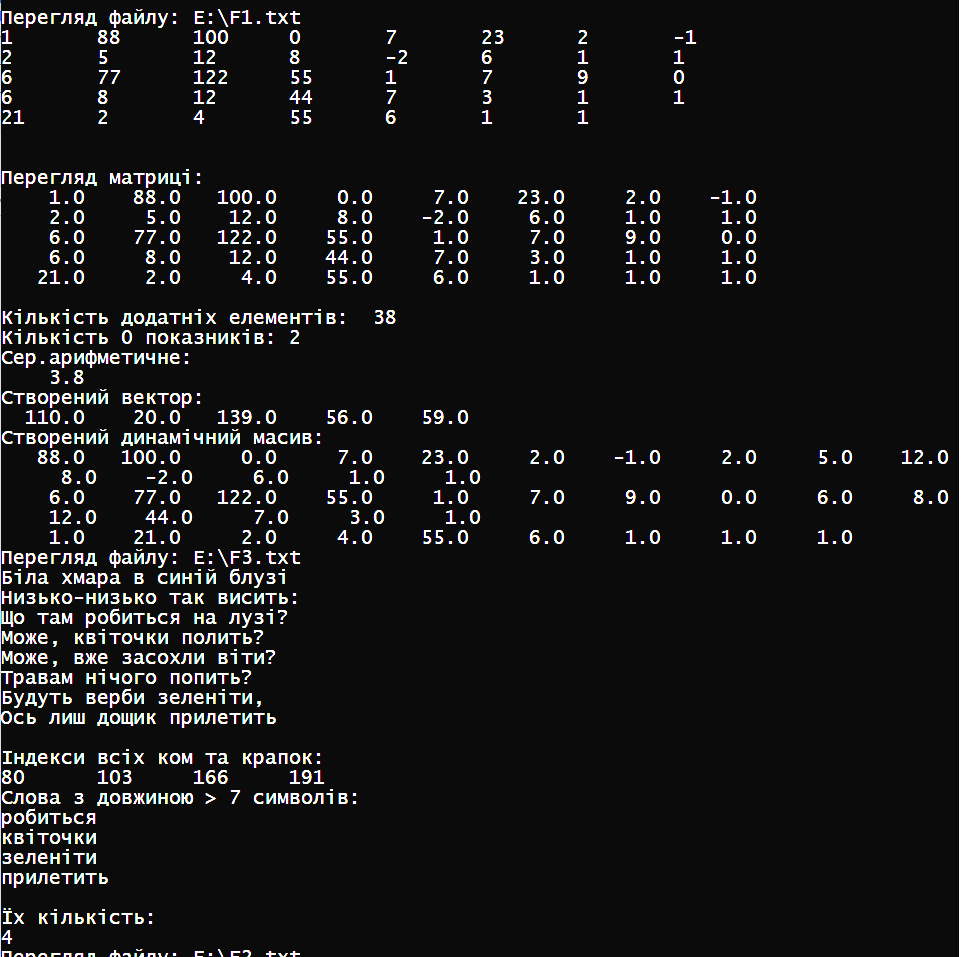


Рисунок 3.1 – Скріншот консольного вікна з результатами роботи

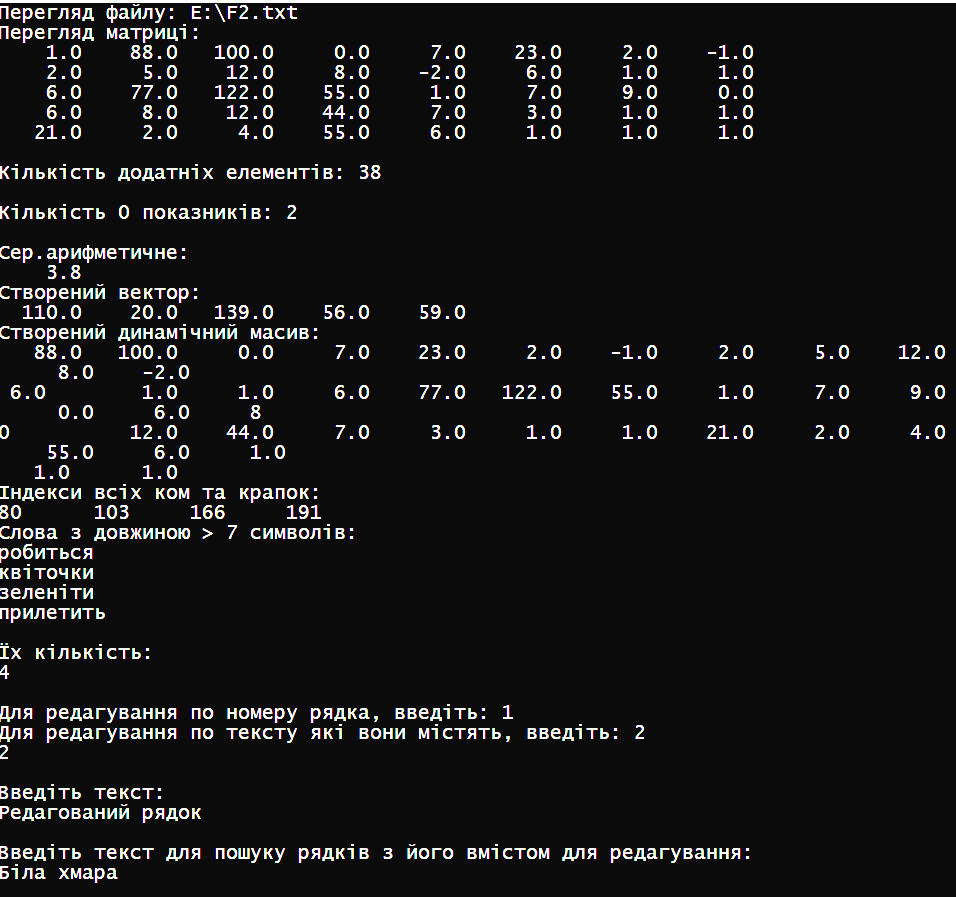


Рисунок 3.2 – Скріншот консольного вікна з результатами роботи

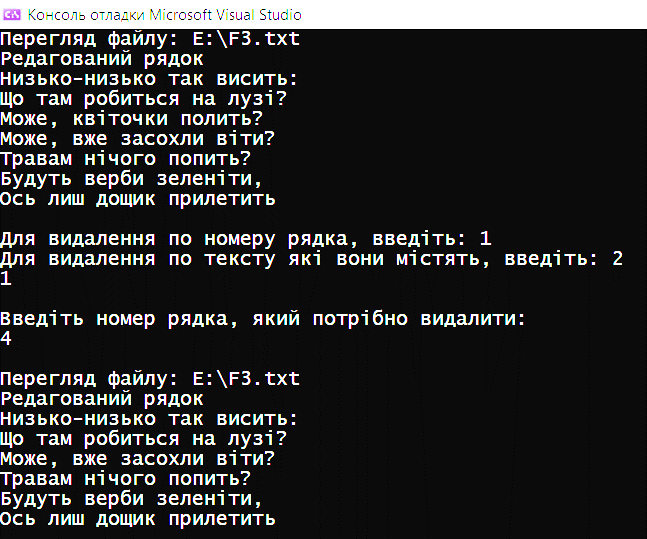


Рисунок 3.3 – Скріншот консольного вікна з результатами роботи

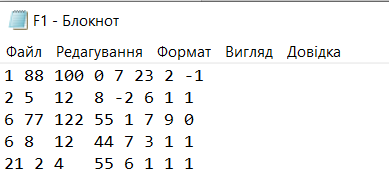


Рисунок 3.4 – Вигляд текстового файлу з вхідною матрицею

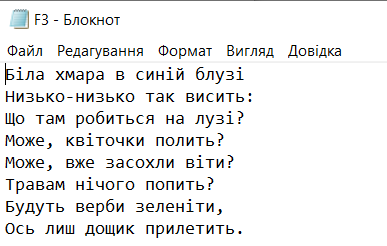


Рисунок 3.5 – Вигляд вхідного текстового файлу з текстовими даними

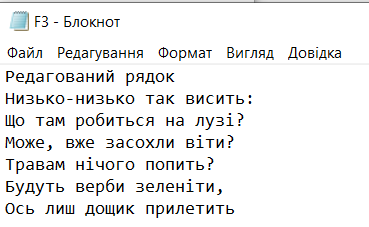


Рисунок 3.6 – Вигляд текстового файлу після опрацювання

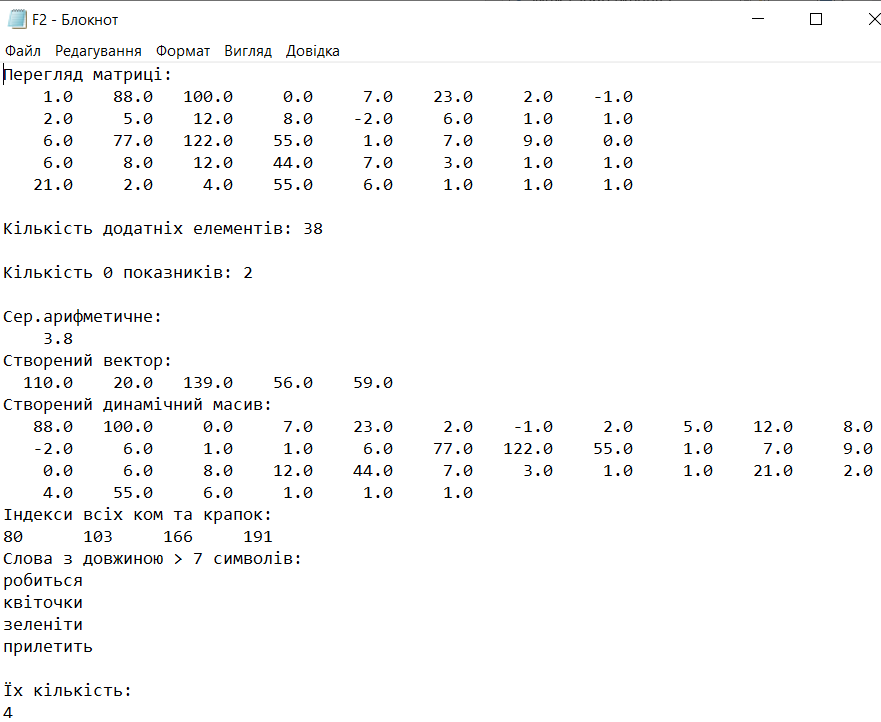


Рисунок 3.7 – Вигляд звітного текстового файлу з результатами програмного опрацювання

# 4 ІНСТРУКЦІЯ З ВИКОРИСТАННЯ ПРОГРАМИ

Для того щоб мати можливість працювати з програмою, потрібно створити текстовий файл і заповнити його числовими значеннями (тобто створити матрицю). Після кожного елементу матриці має бути табуляція (Tab) або пробіл, а також перехід на наступний рядок (Enter). Також потрібно створити ще один текстовий файл і заповнити його коректними текстовими даними.

Створюємо файл з ім’ям F1, як показано на рис. 4.1.

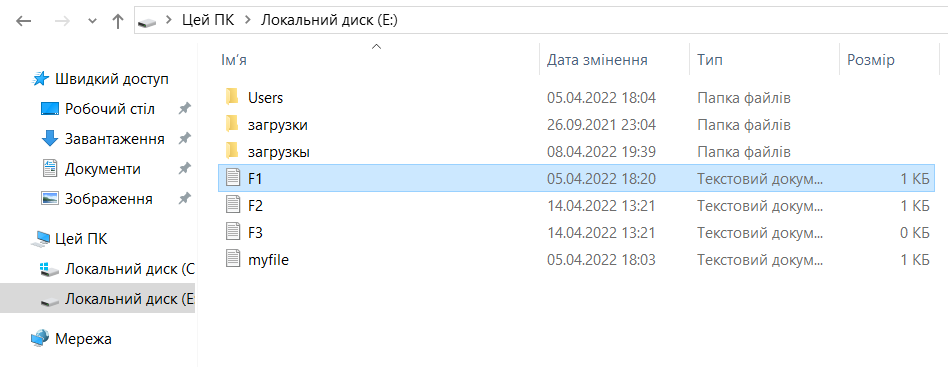


Рисунок 4.1 – Файл F1 у папці

Відкриваємо цей файл та заповнюємо значеннями (рис. 3.4).

Також створюємо файл з ім’ям F3 та вставляємо текст який буде використаний в програмі, як показано на рис. 4.2 та 3.5.

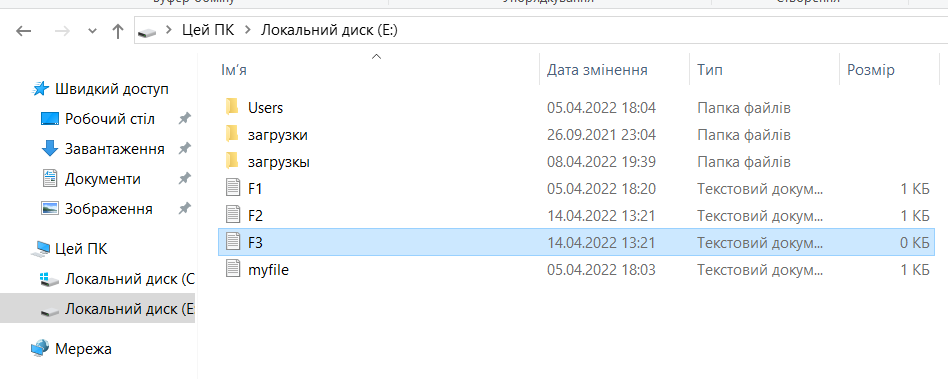


Рисунок 4.2 – Файл F3 у папці

Потім потрібно відкрити програму з ім’ям Kursova.sln та запустити її (рис. 4.3).

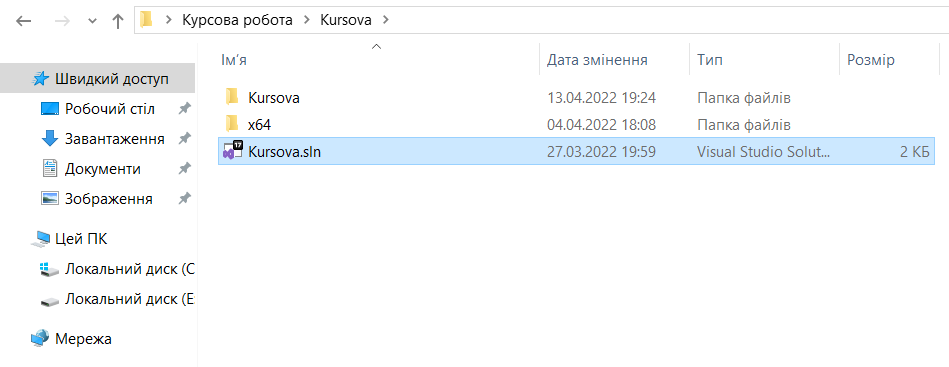


Рисунок 4.3 – Файл Kursova.sln у папці

Щоб отримати результати програми, потрібно відкрити файл Debug.exe (в нашому випадку Kursova) і після цього відкриється консоль з виконаною задачею (рис. 4.4).

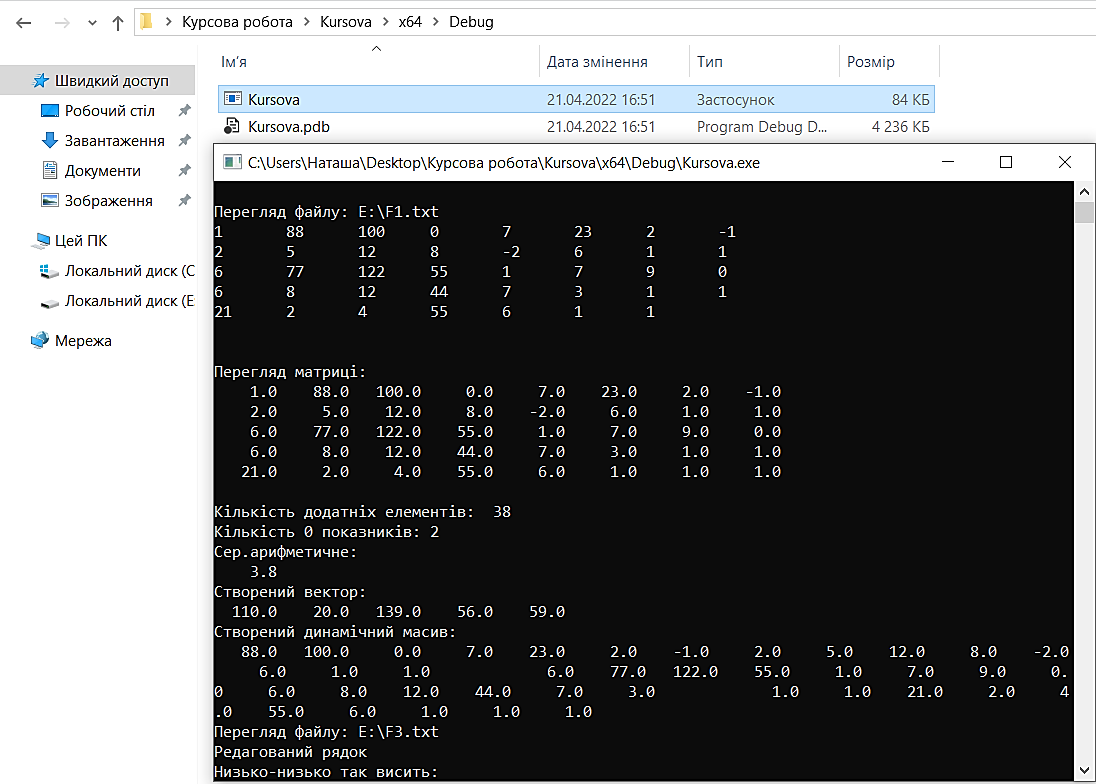


Рисунок 4.4 – Файл Debug і консоль

Результат роботи програми записується у файл з ім’ям F2, як показано на рис. 4.5.

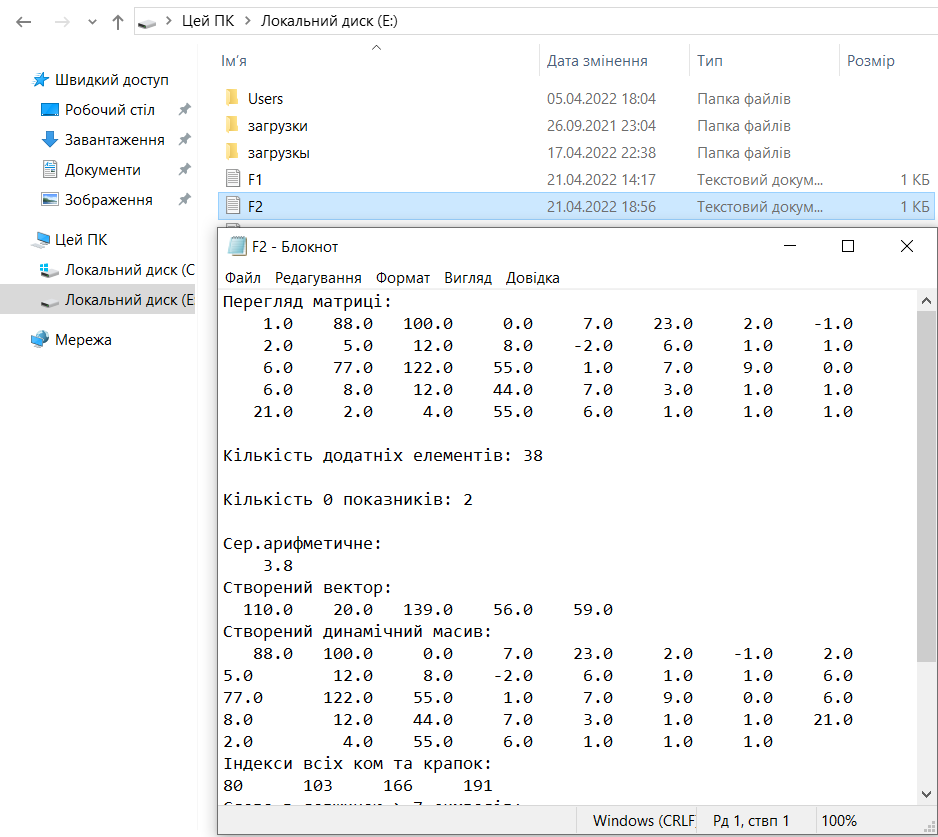


Рисунок 4.5 – Вигляд файлу F2

# ВИСНОВКИ З АНАЛІЗОМ ДОСЯГНУТИХ РЕЗУЛЬТАТІВ

Як висновок, курсова робота допомогла узагальнити, закріпити та розширити практичні і теоретичні знання, уміння і навички програмування мовою високого рівня С. Вона дала змогу укріпити вміння створення схем алгоритмів функцій в проєкті.

У курсовій роботі було розглянуто функції створення, заповнення даними, обчислення, виведення даних, створення динамічного масиву, редагування та видалення даних у файлах. Крім цього, було створено схеми алгоритмів функцій згідно з поставленим завданням.

Курсова робота розкрила сутність технологій мови С з опрацюванням неоднорідних даних у файлах. Було виконано розроблення програмного проєкту для розв’язання завдання відповідно до індивідуального варіанта, який передбачає створення, заповнення даними, редагування та видалення вмісту файлів різних форматів.

Під час роботи над проєктом я покращила уміння пошуку необхідної інформації, здобула досвід в дослідженні й експериментуванні під час розв’язування індивідуальних задач. Закріпила знання алгоритмізації та навичок програмування мовою високого рівня.

Отже, розроблення програмного проєкту підтвердило знання, уміння і навички, здобуті протягом всього навчального року.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Файлы заголовков стандартной библиотеки С и C++. URL: https://docs.microsoft.com/ru-ru/cpp/standard-library/cpp-standard-library-header-files?view=msvc-170 (дата звернення 09.04.2022).
2. Windows.h. URL: <https://rsdn.org/forum/cpp.applied/1284681.hot> (дата звернення 09.04.2022).
3. В чем смысл предупреждения "C4996: '...': This function or variable may be unsafe"? URL: https://ru.stackoverflow.com/questions/26196/В-чем-смысл-предупреждения-c4996-this-function-or-variable-may-be-unsaf (дата звернення 09.04.2022).
4. Ravesli using-стейтменты в С++ / Ravesli. URL: https://ravesli.com/urok-54-using-statements/ (дата звернення 09.04.2022).
5. Управляющие символы в программировании на C++. URL: http://cppstudio.com/post/256/ (дата звернення 09.04.2022).
6. fopen. Программирование на С и С++. URL: http://www.c-cpp.ru/content/fopen (дата звернення 09.04.2022).
7. Синтаксис спецификации формата: 'printf' и 'wprintf'. URL: https://docs.microsoft.com/ru-ru/cpp/c-runtime-library/format-specification-syntax-printf-and-wprintf-functions?view=msvc-170 (дата звернення 09.04.2022).

# ДОДАТКИ

## Додаток А Схеми алгоритмів функцій програмного проєкту

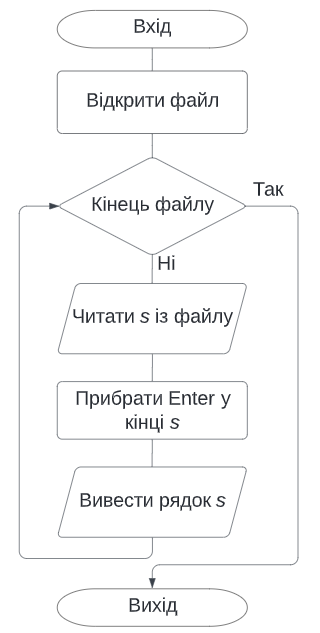
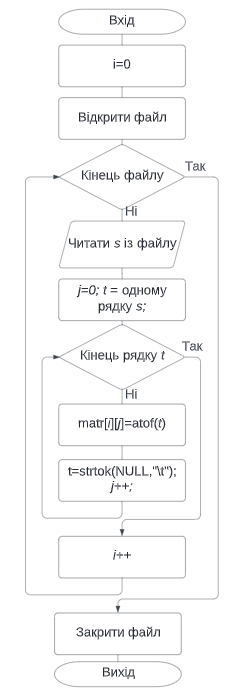
 

Рисунок А.1 – Схема алгоритмуРисунок А.2 – Схема алгоритму

функції view\_file функції create\_matrix

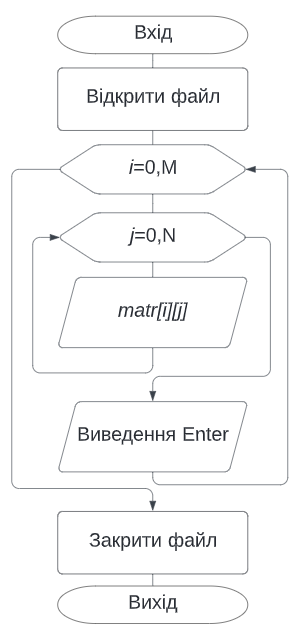
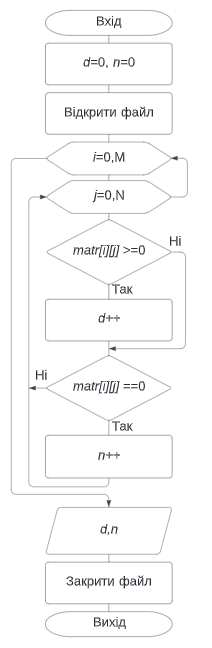
 

Рисунок А.3 – Схема алгоритму Рисунок А.4 – Схема алгоритму

функції *output\_matrix* функції *amount*

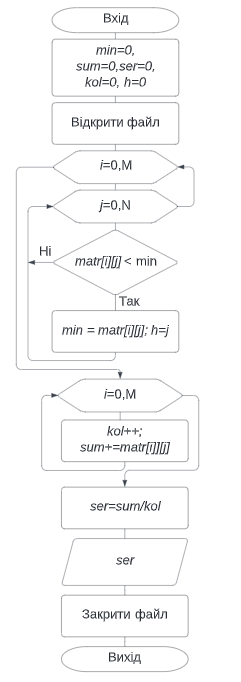
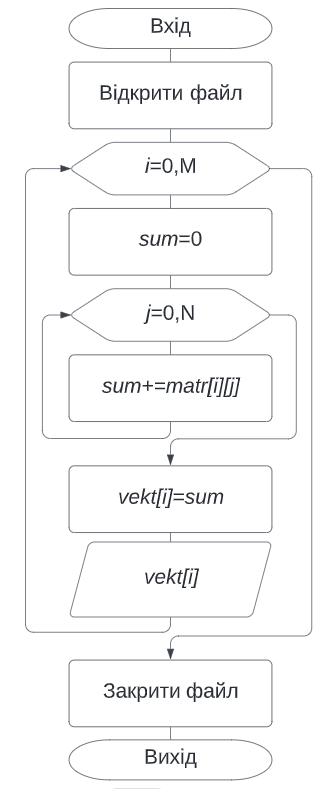
 

Рисунок А.5 – Схема алгоритму Рисунок А.6 – Схема алгоритму

функції *obchusleny* функції *create\_vector*

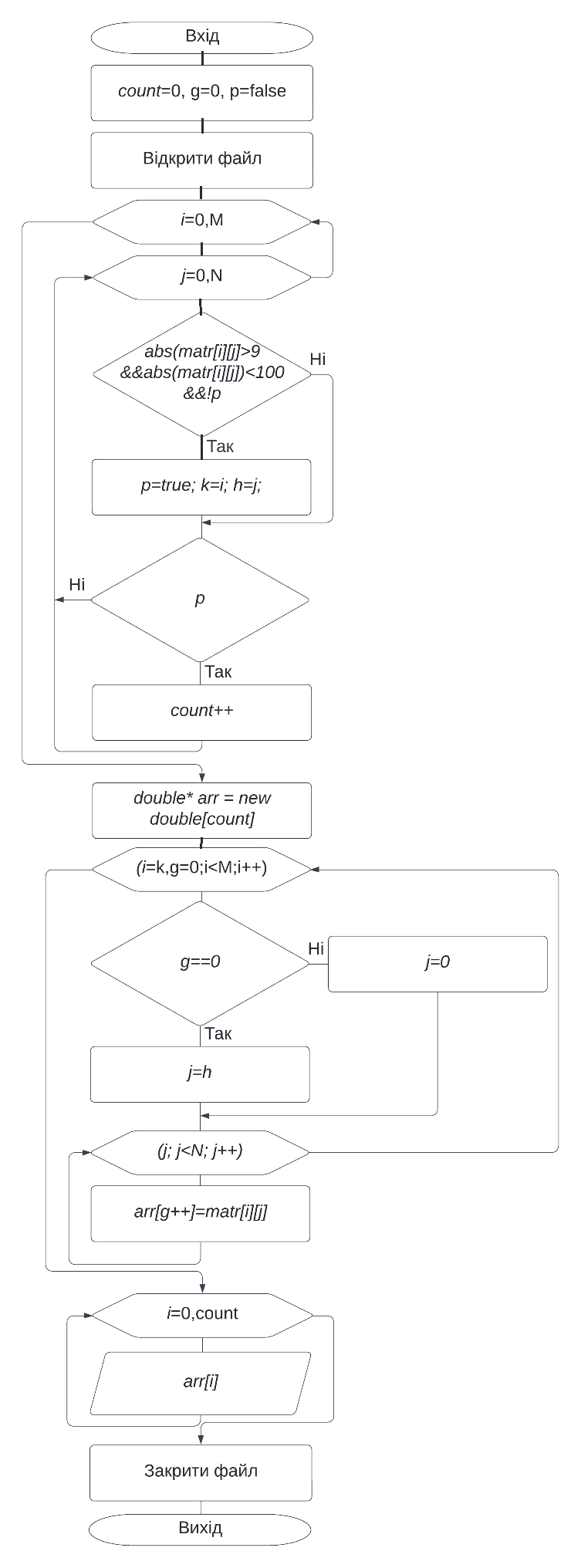


Рисунок А.7 – Схема алгоритму

функції *create\_masiv*

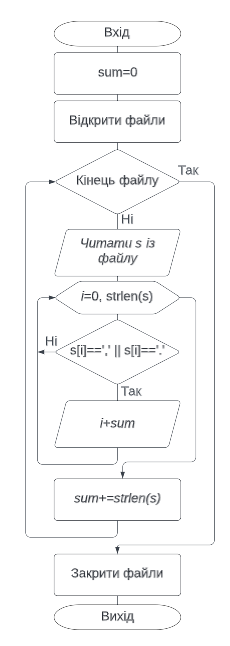
 

Рисунок А.8 – Схема алгоритму Рисунок А.9 – Схема алгоритму

функції *indexsu* функції *slova*

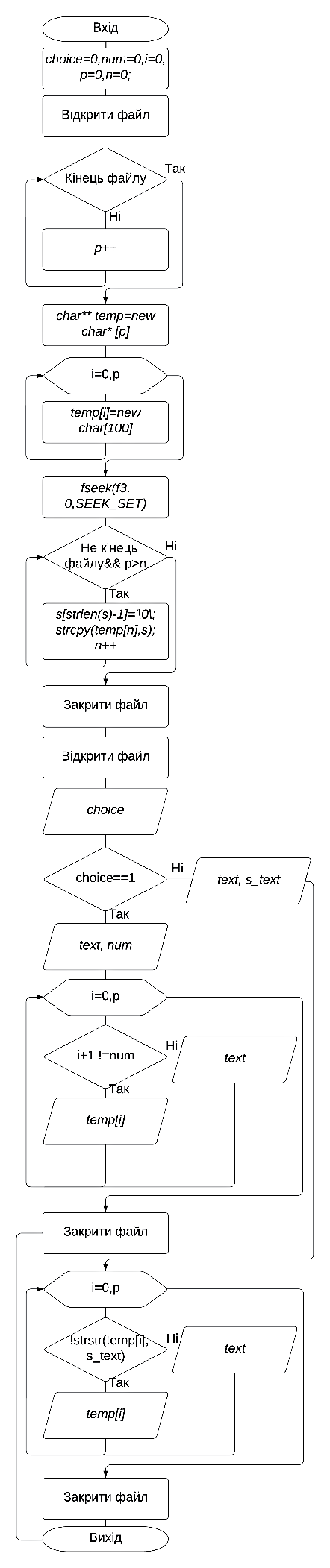
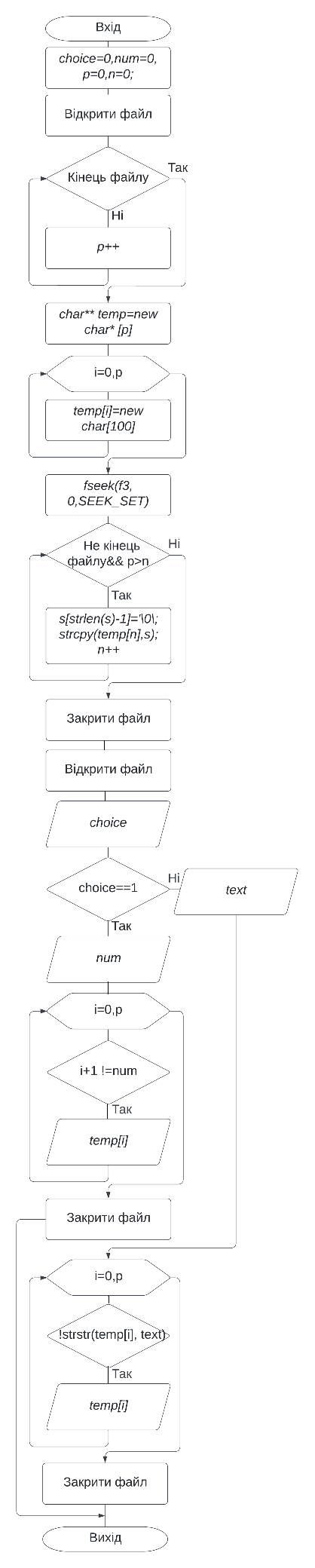
* *

Рисунок А.10 – Схема алгоритму Рисунок А.11 – Схема алгоритму

функції *redaguvany* функції *vudaleny*

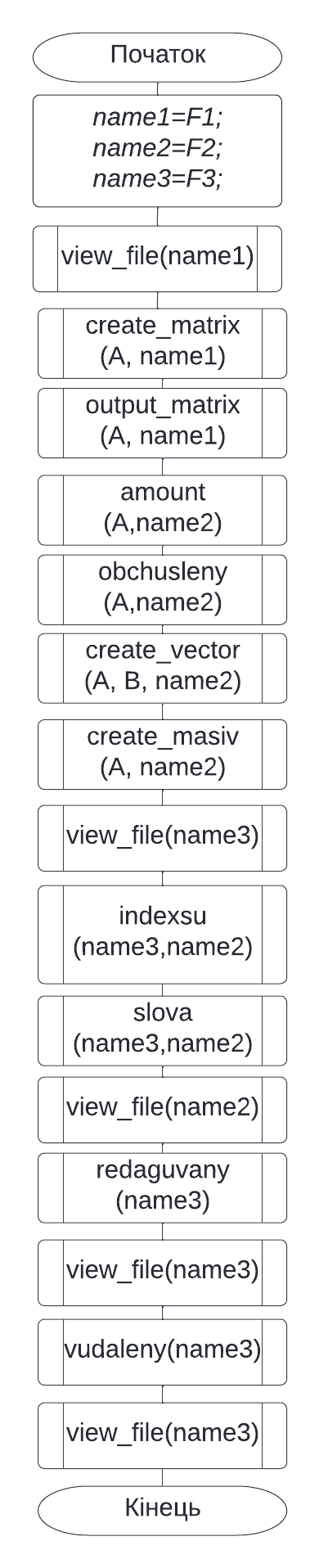
**

Рисунок А.12 – Схема алгоритму

функції *main*

## Додаток Б Програмний код

#include <iostream>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#pragma warning(disable : 4996)

using namespace std;

const int M = 5, N = 8;

**// Переглянути файл**

**void view\_file(char\* name)**

{

char s[100]; FILE\* f;

f = fopen(name, "rt");

if (f == NULL) { puts("Cannot open file to view"); return; }

printf("\nПерегляд файлу: %s\n", name);

while (fgets(s, 100, f))

{

s[strlen(s) - 1] = '\0';

puts(s);

}

fclose(f);

}

**// Заповнити матрицю значеннями із файлу**

**void create\_matrix(double matr[M][N], char\* name)**

{

FILE\* f;

char s[100], \* t;

int i = 0, j;

f = fopen(name, "rt");

if (f == NULL) { puts("Cannot open file to view"); return; }

while (fgets(s, 100, f))

{

j = 0;

t = strtok(s, " \t");

while (t != NULL)

{

matr[i][j] = atof(t);

t = strtok(NULL, " \t");

j++;

}

i++;

}

fclose(f);

return;

}

**// Вивести матрицю на екран та у файл**

**void output\_matrix(double matr[M][N], char\* name2)**

{

FILE\* f2;

int i, j;

f2 = fopen(name2, "wt");

if (f2 == NULL) { puts("Cannot open file"); return; }

printf("\n\nПерегляд матрицi:\n");

fprintf(f2, "Перегляд матрицi:\n");

for (i = 0; i < M; i++)

{

for (j = 0; j < N; j++)

{ //Вивести елемент матриці на екран

printf("%7.1f\t", matr[i][j]);

//Записати елемент у файл

fprintf(f2, "%7.1f\t", matr[i][j]);

}

puts("");

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

}

fclose(f2);

return;

}

**/\*Обчислити кількість додатніх елементів і кількістьть 0 показників вимірювань датчиків (елементів матриці)\*/**

**void amount(double matr[M][N], char\* name2)**

{

FILE\* f2;

int i, j;

f2 = fopen(name2, "at");

if (f2 == NULL)

{

puts("Cannot open file to append data\n"); return;

}

int d = 0, n = 0;

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

{

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

{

if (matr[i][j] >= 0) d++;

if (matr[i][j] == 0) n++;

}

}

printf("\nКількість додатніх елементів: %i\t", d);

fprintf(f2, "\nКількість додатніх елементів: %i\n", d);

printf("\nКількість 0 показників: %i\t", n);

fprintf(f2, "\nКількість 0 показників: %i\n", n);

fclose(f2);

}

**/\*Обчислити середнє арифметичне вимірювання (стовпця матриці),який містить мінімальне значення матриці\*/**

**void obchusleny(double matr[M][N], char\* name2)**

{

FILE\* f2;

f2 = fopen(name2, "at");

if (f2 == NULL)

{

puts("Cannot open file to append data\n"); return;

}

puts("\nСер.арифметичне:");

fprintf(f2, "\nСер.арифметичне:\n");

double min = 0, sum = 0, ser = 0; int kol = 0, h = 0;

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

{

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

if (matr[i][j] < min)

{

min = matr[i][j];

h = j;

}

}

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

{

kol++;

sum += matr[i][h];

}

ser = sum / kol;

printf("%7.1f\t", ser);

fprintf(f2, "%7.1f\t", ser);

fclose(f2);

}

**/\*Обчислити елементи вектора як суми елементів рядків матриці з парними індексами стовпців\*/**

**void create\_vector(double matr[M][N], double vekt[M], char\* name2)**

{

FILE\* f2;

f2 = fopen(name2, "at");

if (f2 == NULL)

{

puts("Cannot open file to append data\n"); return;

}

puts("\nСтворений вектор:");

fprintf(f2, "\nСтворений вектор:\n");

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

{

double sum = 0;

for (int j = 1; j < N; j+=2)

{

sum += matr[i][j];

}

vekt[i] = sum;

printf("%7.1f\t", vekt[i]);

fprintf(f2, "%7.1f\t", vekt[i]);

}

fclose(f2);

}

**//Динамічний масив з чисел, розміщених після першого двоцифровго числа матриці**

**void create\_masiv(double matr[M][N], char\* name2)**

{

FILE\* f2;

f2 = fopen(name2, "at");

if (f2 == NULL)

{

puts("Cannot open file to append data\n"); return;

}

puts("\nСтворений динамічний масив:");

fprintf(f2, "\nСтворений динамічний масив:\n");

int count = 0, g=0;

int k, h;

bool p = false;

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

{

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

{

//Перевірка до першого двоцифрового числа

if (abs(matr[i][j]) > 9 && abs(matr[i][j]) < 100 && !p)

{

p = true;

k = i; h = j;

}

if (p)

{

count++;

}

}

}

double\* arr = new double[count];

for (int i = k, g = 0; i < M; i++)

{

int j;

if (g == 0)

j = h;

else

j = 0;

for ( j ; j < N; j++)

{

arr[g++] = matr[i][j];

}

}

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

{

printf("%7.1f\t", arr[i]);

fprintf(f2, "%7.1f\t", arr[i]);

}

fclose(f2);

}

**//Вивести індекси всіх ком та крапок, які є у рядках**

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

{

FILE\* f3, \* f2;

char s[100]; int sum = 0;

f3 = fopen(name3, "rt");

if (f3 == NULL) { puts("Cannot open file to view"); return; }

f2 = fopen(name2, "at");

if (f2 == NULL) { puts("Cannot open file to append\n"); return; }

puts("\nІндекси всіх ком та крапок:");

fputs("\nІндекси всіх ком та крапок:\n", f2);

while (fgets(s, 100, f3))

{

for (int i = 0; i < strlen(s); i++)

{

if (s[i] == ',' || s[i] == '.')

{

printf("%i\t", i+sum);

fprintf(f2, "%i\t", i+sum);

}

}

sum += strlen(s);

}

fclose(f2);

fclose(f3);

}

**//Вивести слова файлу з довжиною більше 7 символів та обчислити їхню кількість**

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

{

FILE\* f3, \* f2;

int k = 0;

f3 = fopen(name3, "rt");

if (f3 == NULL) { puts("Cannot open file to view"); return; }

f2 = fopen(name2, "at");

if (f2 == NULL) { puts("Cannot open file to append\n"); return; }

char s[100], \* t = new char[100];

puts("\nСлова з довжиною > 7 символів:");

fputs("\nСлова з довжиною > 7 символів:\n", f2);

while (fgets(s, 100, f3))

{

t = strtok(s, " ,.;-^=:!?");

while (t != NULL)

{

if (strlen(t) > 7)

{

t = strtok(t, " ,.;-^=:!?");

k++;

puts(t);

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

}

t = strtok(NULL, " ,.;-^=:!?");

}

}

puts("\nЇх кількість:");

fputs("\nЇх кількість:\n", f2);

printf("%i\t", k);

fprintf(f2, "%i\t", k);

fclose(f2);

fclose(f3);

}

**//Редагувати файл F3.txt з текстовими даними**

**void redaguvany(char\* name3)**

{

char s[100];

char s\_text[100]; char text[100];

int choiсe = 0, num = 0, i = 0, p = 0, n = 0;

FILE\* f3;

f3 = fopen(name3, "rt");

if (f3 == NULL) { puts ("Cannot open file to view"); return; }

while (fgets(s, 100, f3)) p++;

char\*\* temp = new char\* [p];

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

temp[i] = new char[100];

fseek(f3, 0, SEEK\_SET);

while (fgets(s,100,f3) && p>n)

{

s[strlen(s) - 1] = '\0';

strcpy(temp[n], s);

n++;

}

fclose(f3);

f3 = fopen(name3, "wt");

cout<<"\nДля редагування по номеру рядка, введіть: 1\n" << "Для редагування по тексту які вони містять, введіть: 2\n";

cin >> choiсe;

if (choiсe == 1)

{

cout<<"\nВведіть текст:\n";

getchar();

gets\_s(text, 100);

cout<<"\nВведіть номер рядка, який потрібно редагувати:\n";

cin >> num;

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

if (i + 1 != num) fprintf(f3, "%s\n", temp[i]);

else fprintf(f3, "%s\n", text);

fclose(f3);

return;

}

else

{

cout<<"\nВведіть текст:\n";

getchar();

gets\_s(text, 100);

cout<<"\nВведіть текст для пошуку рядків з його вмістом для редагування:\n";

gets\_s(s\_text, 100);

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

if (!strstr(temp[i], s\_text)) fprintf(f3, "%s\n", temp[i]);

else fprintf(f3, "%s\n", text);

fclose(f3);

return;

}

}

**//Видалення рядків з файлу**

**void vudaleny(char\* name3)**

{

char s[100];

char text[100];

int choiсe = 0, num = 0, p = 0, n = 0;

FILE\* f3;

f3 = fopen(name3, "rt");

if (f3 == NULL) { puts("Cannot open file to view"); return; }

while (fgets(s, 100, f3)) p++;

char\*\* temp = new char\* [p];

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

temp[i] = new char[100];

fseek(f3, 0, SEEK\_SET);

while (fgets(s, 100, f3) && p > n)

{

s[strlen(s) - 1] = '\0';

strcpy(temp[n], s);

n++;

}

fclose(f3);

f3 = fopen(name3, "wt");

cout << "\nДля видалення по номеру рядка, введіть: 1\n" << "Для видалення по тексту які вони містять, введіть: 2\n";

cin >> choiсe;

if (choiсe == 1)

{

cout << "\nВведіть номер рядка, який потрібно видалити:\n";

cin >> num;

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

if (i + 1 != num) fprintf(f3, "%s\n", temp[i]);

fclose(f3);

return;

}

else

{

cout << "\nВведіть текст для пошуку рядків з його вмістом для видалення:\n";

getchar();

gets\_s(text, 100);

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

if (!strstr(temp[i], text)) fprintf(f3, "%s\n", temp[i]);

fclose(f3);

return;

}

}

**int main()**

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

double A[M][N], B[N];

char name1[] = "E:\\F1.txt";

char name2[] = "E:\\F2.txt";

char name3[] = "E:\\F3.txt";

//Перегляд файлу

view\_file(name1);

//Заповнити матрицю числами із файлу

create\_matrix(A, name1);

//Вивести матрицю на екран та у файл

output\_matrix(A, name2);

/\*Обчислити кількість додатніх елементів і кількість 0 показників вимірювань датчиків (елементів матриці)\*/

amount(A, name2);

/\*Обчислити середнє арифметичне вимірювання (стовпця матриці),який містить мінімальне значення матриці\*/

obchusleny(A, name2);

/\*Обчислити елементи вектора як суми елементів рядків матриці з парними індексами стовпців\*/

create\_vector(A, B, name2);

//Динамічний масив з чисел, розміщених після першого двоцифрового числа матриці

create\_masiv(A, name2);

//Переглянути файл

view\_file(name3);

//Вивести індекси всіх ком та крапок, які є у рядках

indexsu(name3, name2);

//Вивести слова файлу з довжиною більше 7 символів та обчислити їхню кількість

slova(name3, name2);

//Переглянути файл

view\_file(name2);

//Редагувати файл F3.txt з текстовими даними

redaguvany(name3);

//Переглянути файл

view\_file(name3);

//Видалення рядків з файлу

vudaleny(name3);

//Переглянути файл

view\_file(name3);

}