**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ   
«ОДЕСЬКА ЮРИДИЧНА АКАДЕМІЯ»**

**ФАКУЛЬТЕТ КІБЕРБЕЗПЕКИ ТА ІНФОРМАЦІЙНИХ ТЕХЛОГОЛОГІЙ**

Кафедра інформаційних технологій

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

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

студентки 1 курсу групи 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 НА КУРСОВУ РОБОТУ**

з дисципліни «Алгоритмізація та програмування»

студентки Ярош Євгенії В’ячеславівни

спеціальності 121 Інженерія програмного забезпечення

першого курсу групи 111

Тема «Програмне опрацювання даних у файлах»

Варіант 4

Вхідні дані:

* текстовий файл *K*1.txt із числовими даними вимірювань датчиків у вигляді матриці розмірності *M* на *N* (*M* – кількість датчиків, *N* – кількість вимірювань);
* файл *K*3.txt з текстом.

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) як вхідні дані для програмного опрацювання файлових даних створити текстовий файл, наприклад, з ім’ям *K1*, в який записати числові дані вимірювань датчиків у вигляді матриці розмірності *M* на *N* (*M* – кількість датчиків, *N* – кількість вимірювань);

2) у програмному проекті розробити функцію, яка дозволить вивести на екран вміст текстового файлу *K1*, ім’я якого передаватиметься як аргумент із функції *main();*

3) створити функції для:

* формування числової матриці розмірності *M* на *N* зчитуванням її значень із файлу *K1*. Для базового та підвищеного рівня (\*) розмірність матриці *M* та *N* задати константами;
* виведення цієї матриці на екран та у файл *K2*;

4) створити функцію для опрацювання числової матриці з обчисленням значень відповідно до індивідуального варіанта: обчислити значення кількості вимірювань датчиків (елементів матриці), абсолютне значення яких менше 1, а також визначити кількість ненульових елементів лише у тих рядках, які не містять від’ємних елементів. Результати обчислень записати у файл *K2*;

5) створити функцію для опрацювання числової матриці з формуванням елементів вектора відповідно до індивідуального варіанта: обчислити елементи вектора як середнє арифметичне елементів першого та останнього рядків матриці. Результати обчислень записати у файл *K2*;

6) як вхідні дані для програмного опрацювання текстових даних створити ще один файл з ім’ям *K3.txt* та заповнити його коректними текстовими даними;

7) програмно переглянути на екрані вміст текстового файлу *K3.txt*, ім’я якого передаватиметься як аргумент із функції *main();*

8) розробити функцію, яка виконуватиме програмне опрацювання даних з файлу *K*3.txt:

* cтворити нові рядки з малих літер латиниці рядків файлу;
* вивести рядки, які містять хоча б одне слово лише з великих літер.

Програмно записати результати опрацювання текстових даних у файл (або файл *K2.txt*, або новий). Програмно переглянути на екрані вміст текстового файлу з результатами опрацювання текстових даних.

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |
| --- | --- | --- |
| № | Завдання | Термін виконання |
| 1 | Ознайомлення з завданням до курсової роботи. Підготування вхідних файлів | 15.03.2022-20.03.2022 |
| 2 | Частина 1. Опрацювання матриць | 21.03.2021-01.04.2022 |
| 3 | Частина 2. Опрацювання рядкових даних | 04.04.2022-15.04.2022 |
| 4 | Частина 3. Опрацювання текстових файлів | 18.04.2022-29.04.2022 |
| 5 | Оформлення пояснювальної записки | 02.05.2022-13.05.2022 |

**ЗМІСТ**

[Вступ 5](#_Toc103624354)

[1 Аналіз засобів та методів розробки 7](#_Toc103624355)

[2 Словесний опис алгоритмів для розв’язування задач 8](#_Toc103624356)

[2.1 Опис функції view\_file() 8](#_Toc103624357)

[2.2 Опис функції create\_matrix() 9](#_Toc103624358)

[2.3 Опис функції output\_matrix() 10](#_Toc103624359)

[2.4 Опис функції kilkist() 11](#_Toc103624360)

[2.5 Опис функції kilkist2() 12](#_Toc103624361)

[2.6 Опис функції create\_vector() 13](#_Toc103624362)

[2.7 Опис функції mal() 15](#_Toc103624363)

[2.8 Опис функції vel() 16](#_Toc103624364)

[2.9 Опис функції redag() 18](#_Toc103624365)

[2.9 Опис функції vud() 20](#_Toc103624366)

[3 Результати виконання програми 23](#_Toc103624367)

[4 Інструкція з використання програми 26](#_Toc103624368)

[Висновки з аналізом досягнутих результатів 28](#_Toc103624369)

[Список використаних джерел 30](#_Toc103624370)

[Додаток А Програмний код 31](#_Toc103624371)

[Додаток Б Схеми алгоритмів программи 36](#_Toc103624372)

# Вступ

**Актуальність теми** курсової роботи "Програмне опрацювання даних у файлах" зумовлена повсюдною практичною затребуваністю вмінь з розробки програмних проектів, які зберігають і обробляють дані у файлах. Тобто програма сама має створювати файли того чи іншого формату, записувати у них дані, на вимогу виводити або всі дані файлу, або лише ті, які задовольняють певній умові, вміти редагувати і видаляти певні дані файлу, надавати можливість створення нових файлів (звітних документів) з певними даними з основного файлу та ін.

**Об'єктом дослідження** є базові технології програмування алгоритмічною мовою високого рівня і засоби створення програмних проектів.

**Предмет дослідження** – програмні засоби мови C++ для опрацювання різнорідних даних, створення файлів різних форматів, записування у них даних, виведення вмісту цих файлів, відбирання даних за умовою, можливості редагування даних, створення нових файлів (звітних документів) з певними даними з основного файлу.

**Мета** курсової роботи полягає у:

* систематизації, закріпленні та розширенні теоретичних і практичних знань за фахом та використання їх під час розв’язання конкретних прикладних задач у галузі комп’ютерних технологій та програмування;
* розвиненні навичок самостійної роботи з літературою, нормативними документами, пошуку необхідної інформації, у тому числі в Інтернеті, оволодінні методикою дослідження й експериментування під час розв’язання задач і питань, які розроблюються у курсовій роботі;
* закріпленні знань алгоритмізації та навиків програмування алгоритмічною мовою високого рівня при опрацюванні різнорідних даних у файлах з можливістю їх програмного редагування;
* виявленні підготовки здобувачів вищої освіти до самостійної роботи.

Відповідно до поставленої мети визначено такі задачі:

* розкрити сутність технологій мови С++ з опрацювання різнорідних даних у файлах;
* дослідити наявні сучасні програмні засоби, які дозволяють створювати програмні проєкти мовою C++;
* вивчити засоби мови С++ з програмного створення файлів різних форматів, записування у них даних, перегляду вмісту цих файлів, відбору даних за умовою, редагування даних у файлах (для задач підвищеного рівня складності);
* розробити програмний проєкт для розв’язання завдання відповідно до індивідуального варіанта, який передбачає створення, заповнення даними, відбір та редагування файлів різних форматів.

Курсова робота складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. Загальний обсяг курсової роботи: 41 с., 17 рис., 2 додатки, 3 джерел.

# 1 Аналіз засобів та методів розробки

Під час роботи над курсовою використовувались теоретичні знання базових конструкцій мови програмування C++.

Кожна програма на C++, як правило, починається з директив препроцесора, їх можна впізнати за символом #. Під час написання програмного коду використовувались такі директиви:

* *iostream* – директива, яка організує введення та виведення;
* *stdlib.h –* директива, яка займається виділенням пам’яті та контролем процесу виконання програми;
* *string.h –* директива, яка дозволяє працювати з рядками;
* pragma warning(disable : 4996) *–* директива, щоб не виникали помилки компіляції у Visual Studio [1].

Для підключення функціоналу в програмі, надані операційною системою (Windows 95, 98,NT, 2000, XP) використовується бібліотека – windows.h [2].

Якщо потрібно, щоб компілятор використовував все, що є в просторі імен std треба прописати такий рядок – using namespace std [1].

В програмі використовуються функції:

* fopen() для того, щоб відкрити файл;
* puts() для виведення рядку;
* print() виводить на екран символи або значення;
* fgets() для зчитування даних;
* fprintf() виводить в файл символи і значення;
* strlen() рахує кількість символів до термінованого нуля;
* strtok() виконує пошук лексем;
* atof() перетворює рядок в значення типу double;
* fclose() для того, щоб закрити файл;
* *fseek()* скидає прапорець кінця файлу, пов'язаний із зазначеним потоком;
* *getchar()* повертає наступний символ стандартного потоку введення (stdin);
* *strcpy()* для копіювання нуль-термінованого рядка (включно з нуль-термінатором) у заданий буфер;
* *gets\_s()* зчитує символи з stdin і поміщає в масив символів, який вказує str.

Також використовуються умовний оператор if та оператори циклу for, while [3].

# 2 Словесний опис алгоритмів для розв’язування задач

## **2.1 Опис функції view\_file()**

Функція *view\_file()* здійснює відкривання текстового файлу для перегляду та зчитування.

Аргументами функції є: рядкова змінна *name* – назва фізичного файлу з числовими даними вимірювань датчиків у вигляді матриці, створеного у Блокноті для цієї частини роботи.

Функція має тип *void*, тобто не повертає жодного значення за допомогою оператора повернення *return*.

Локальні змінні:

* *s* – рядок довжиною до 100 символів для зчитування тексту із файлу;
* *f* – файлова змінна вказівник на структуру типу *FILE*, за допомогою якої здійснюється доступ до файлу.

Словесний алгоритм функції:

1. Відкрити файл:

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

Функція *fopen()* відкриває файл. Перший аргумент *name* – ім’я файлу, що відкривається; другий аргумент "*rt*" – це режим відкривання:

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

2. Перевірити, чи відкрилися файли. Якщо ні, то вивести на екран відповідне

повідомлення і перервати виконання програми:

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

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

3. У циклі while зчитувати з файлу послідовно рядки у s, допоки вони є у файлі (до кінця файлу) та вивести рядок на екран:

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

*{*

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

*puts(s);*

*}*

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

4.Закрити файл:

*fclose(f);*

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

Функція *create\_matrix*() заповнює матрицю значеннями з текстового файлу, створеного у Блокноті для цієї частини роботи.

Аргументами функції є: дійсний двовимірний масив з ім’ям *matr*; змінна *name* – назва фізичного файлу з числовими даними вимірювань датчиків у вигляді матриці, створеного у Блокноті.

Функція має тип *void*, тобто не повертає жодного значення за допомогою оператора повернення *return*.

Локальні змінні:

* *f* – файлова змінна вказівник на структуру типу *FILE*, за допомогою якої здійснюється доступ до файлу;
* *s* – рядок довжиною до 100 символів для зчитування тексту із файлу;
* *\*t* – вказівник;
* *i* – змінна цілого типу, слугує як індекс рядків матриці;
* *j* – змінна цілого типу, слугує як індекс стовпців матриці.

Словесний алгоритм функції:

1. Відкрити файл:

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

Функція *fopen()* відкриває файл. Перший аргумент *name* – ім’я файлу, що відкривається; другий аргумент "*rt*" – це режим відкривання:

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

2. Перевірити, чи відкрилися файли. Якщо ні, то вивести на екран відповідне

повідомлення і перервати виконання програми:

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

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

3. У циклі while зчитувати з файлу послідовно рядки у s, допоки вони є у файлі (до кінця файлу):

while (fgets(s, 100, f))

4. Заповнюємо матрицю даними з файлу:

*j = 0;*

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

У циклі використовуємо функцію *strtok* викликів функції розбиває рядок на лексеми, розділені символами роздільниками.

*while (t != NULL)*

*{*

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

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

*j++;*

*}*

*i++;*

Функція atof перетворює рядок в число типу double.

5.Закрити файл:

*fclose(f);*

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

Функція *output\_matrix()* здійснює виведення створеної матриці на екран та у файл з ім’ям *K2*.*txt*.

Аргументами функції є: дійсний двовимірний масив з ім’ям *matr*; змінна *name2* – назва фізичного файлу з даними та з результатами відбору, створеного у Блокноті.

Функція має тип *void*, тобто не повертає жодного значення за допомогою оператора повернення *return*.

Локальні змінні:

* *i* – змінна цілого типу, слугує як індекс рядків матриці;
* *j* – змінна цілого типу, слугує як індекс стовпців матриці;
* *f* – файлова змінна вказівник на структуру типу *FILE*, за допомогою якої здійснюється доступ до файлу.

Словесний алгоритм функції:

1. Створити файл для записування даних:

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

Функція *fopen()* відкриває файл. Перший аргумент *name* – ім’я файлу, що відкривається; другий аргумент "*wt*" – це режим відкривання:

"*wt*" – створити файл як текстовий для записування даних.

2. Перевірити, чи відкрилися файли. Якщо ні, то вивести на екран відповідне

повідомлення і перервати виконання програми:

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

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

3. За допомогою вкладеного циклу *for* здійснюємо виведення елементів матриці на екран, та запис елементів у файл:

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

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

*{*

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

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

*}*

*puts(" ");*

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

4.Закрити файл:

*fclose(f2);*

## **2.4 Опис функції kilkist()**

Функція *kilkist()* обчислює значення кількості елементів матриці, абсолютне значення яких менше 1 та виводить результати на екран та у файл з ім’ям *K2*.*txt*.

Аргументами функції є: дійсний двовимірний масив з ім’ям *matr*; змінна *name2* – назва фізичного файлу з даними та з результатами відбору, створеного у Блокноті для цієї частини роботи.

Функція має тип *void*, тобто не повертає жодного значення за допомогою оператора повернення *return*.

Локальні змінні:

* *h=0* – змінна цілого типу, слугує для рахування кількісті елементів менших за 1;
* *i* – змінна цілого типу, слугує як індекс рядків матриці;
* *j* – змінна цілого типу, слугує як індекс стовпців матриці;
* *f2* – файлова змінна, за допомогою якої здійснюється доступ до файлів.

Словесний алгоритм функції:

1. Відкрити файл:

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

Функція *fopen()* відкриває файл. Перший аргумент *name2* – ім’я файлу, що відкривається; другий аргумент "*at*" – це режим відкривання:

"*at*" – файл відкривається як текстовий для дописування даних у кінець.

2. Перевірити, чи відкрилися файли. Якщо ні, то вивести на екран відповідне повідомлення і перервати виконання програми:

*if (f2 == NULL)*

*{puts("Не вдається відкрити файл для додавання даних\n"); return;}*

3. За допомогою оператора *for* реалізовано вкладений цикл де за допомогою оператора *if* здійснюється перевірка елементів матриці на умову, якщо умова виконується, то змінна *h* збільшується на одиницю:

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

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

*if (fabs(matr[i][j]) < 1) { h++;}}}*

4. Результат вивести на екран та у файл з результатами обчислення:

*printf("\nКількість елементів менших за 1: %i\t", h);*

*fprintf(f2, "\nКількість елементів менших за 1: %i\n", h);*

5. Закрити файл:

*fclose(f2);*

## **2.5 Опис функції kilkist2()**

Функція *kilkist2()* обчислює кількість ненульових елементів лише у тих рядках, які не містять від’ємних елементів та виводить результати на екран та у файл з ім’ям *K2*.*txt*.

Аргументами функції є: дійсний двовимірний масив з ім’ям *matr*; змінна *name2* – назва фізичного файлу з даними та з результатами відбору, створеного у Блокноті для цієї частини роботи.

Функція має тип *void*, тобто не повертає жодного значення за допомогою оператора повернення *return*.

Локальні змінні:

* *kol=0* – змінна цілого типу, слугує для рахування кількісті ненульових елементів;
* *h=0* – змінна цілого типу, слугує для запам’ятовування індексу;
* *i* – змінна цілого типу, слугує як індекс рядків матриці;
* *j* – змінна цілого типу, слугує як індекс стовпців матриці;
* *f2* – файлова змінна, за допомогою якої здійснюється доступ до файлів.

Словесний алгоритм функції:

1. Відкрити файл:

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

Функція *fopen()* відкриває файл. Перший аргумент *name2* – ім’я файлу, що відкривається; другий аргумент "*at*" – це режим відкривання:

"*at*" – файл відкривається як текстовий для дописування даних у кінець.

2. Перевірити, чи відкрилися файли. Якщо ні, то вивести на екран відповідне повідомлення і перервати виконання програми:

*if (f2 == NULL)*

*{ puts("Не вдається відкрити файл для додавання даних\n"); return;}*

3. Вивести повідомлення про характер розв’язуваної задачі на екран та у файл із результатами:

*puts("\nКількість ненульових елементів:");*

*fprintf(f2, "\nКількість ненульових елементів:");*

4. За допомогою оператора *for* реалізовано вкладений цикл де за допомогою оператора *if* здійснюється перевірка елементів матриці на умову, якщо умова виконується, то запам’ятовуємо індекс:

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

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

*if (fabs(matr[i][j]) < 0) { h=i }}*

5. За допомогою оператора *for* реалізовано цикл де за допомогою оператора *if* здійснюється перевірка елементів матриці на умову, якщо умова виконується, то збільшуємо змінну *kol* на одиницю:

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

*{ if (i != h && matr[i][j] != 0) kol++;}}*

6. Результат вивести на екран та у файл з результатами обчислення:

*printf("%i\t", kol);*

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

7. Закрити файл:

*fclose(f2);*

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

Функція *create\_vector()* обчислює елементи вектора як середнє арифметичне елементів першого та останнього рядків матриці у файлі з ім’ям *K*2.txt, виводить результати на екран та у файл з ім’ям *K2*.*txt*.

Аргументами функції є: дійсний двовимірний масив з ім’ям *matr*; дійсний одновимірний масив *vekt*; змінна *name2* – назва фізичного файлу з даними та з результатами відбору, створеного у Блокноті для цієї частини роботи.

Функція має тип *void*, тобто не повертає жодного значення за допомогою оператора повернення *return*.

Локальні змінні:

* *sum* – змінна дійсного типу яка рахує суму чисел першого і останнього рядків;
* *ser* – змінна дійсного типу яка рахує середнє арифметичне елементів першого і останнього рядків;
* *p* – змінна цілого типу, слугує для перевірки істинності;
* *i* – змінна цілого типу, слугує як індекс рядків матриці;
* *j* – змінна цілого типу, слугує як індекс стовпців матриці;
* *f2* – файлова змінна, за допомогою якої здійснюється доступ до файлів.

Словесний алгоритм функції:

1.Відкрити файл:

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

Функція *fopen()* відкриває файл. Перший аргумент *name2* – ім’я файлу, що відкривається; другий аргумент "*at*" – це режим відкривання:

"*at*" – файл відкривається як текстовий для дописування даних у кінець.

2. Перевірити, чи відкрилися файли. Якщо ні, то вивести на екран відповідне повідомлення і перервати виконання програми:

*if (f2 == NULL)*

*{ puts("Не вдається відкрити файл для додавання даних\n"); return;}*

3. Вивести повідомлення про характер розв’язуваної задачі на екран та у файл із результатами:

*puts("\nСтворений вектор:");*

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

4. За допомогою оператора *for* реалізовано вкладений цикл де за допомогою оператора *if* здійснюється перевірка елементів матриці на умову, якщо умова виконується елементи підсумовуються в змінній *sum*, та змінна *h*=1:

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

*{*

*sum = 0, ser = 0;*

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

*if (i == 0 || i == M - 1)*

*{ sum += matr[i][j]; p = 1;}*

*}*

5. За допомогою двох операторів *if* здійснюється перевірка. Якщо умова істинна, то рахується середнє арифметичне елементів та записується в одновимірний масив *vekt* та вивести результат на екран та у файл з результатами обчислення:

*if (i == 0 || i == M - 1) {*

*if (p) {*

*ser = sum / N;*

*vekt[i] = ser;*

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

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

6. Закрити файл:

*fclose(f2);*

## **2.7 Опис функції mal()**

Функція *mal*() створює нові рядки з малих літер латиниці рядків файлу *K3.txt*, виводить їх на екран та у файл з ім’ям *K2.txt*.

Аргументами функції є: рядкова змінна *name3* – назва фізичного файлу з текстовими даними, створеного у Блокноті для цієї частини роботи; *name2* – ім’я файлу з результатами відбору.

Функція має тип *void*, тобто не повертає жодного значення за допомогою оператора повернення return.

Локальні змінні:

* *i* – змінна цілого типу, слугує як індекс рядків матриці;
* *j* – змінна цілого типу, слугує як індекс стовпців матриці;
* *f2* – файлова змінна, за допомогою якої здійснюється доступ до файлів.

Словесний алгоритм роботи функції:

1. Відкрити файли та перевірити, чи відкрилися файли. Якщо ні, то вивести на екран відповідне повідомлення і перервати виконання програми:

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

*if (f3 == NULL) { puts("Не вдається відкрити файл для перегляду"); return; }*

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

*if (f2 == NULL) { puts("Не вдається відкрити файл для додавання даних\n"); return; }*

Функція *fopen()* відкриває файл. Перший аргумент *name3* або *name2* – ім’я файлу, що відкривається; другий аргумент "*rt*" або "*at*" – це режим відкривання:

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

"*at*" – файл відкривається як текстовий для дописування даних у кінець.

2. Вивести повідомлення про характер розв’язуваної задачі на екран та у файл із результатами:

*puts("\nНові рядки з малих літер латиниці:");*

*fputs("\nНові рядки з малих літер латиниці:\n", f2);*

3. У циклі *while* зчитувати з файлу послідовно рядки у *s*, допоки вони є у файлі (до кінця файлу):

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

4. За допомогою оператора *for* реалізуємо вкладений цикл з умовним оператор *if* та перевіряємо умову:

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

*if (s[i] >= 'a' && s[i] <= 'z'){*

*printf("%c", s[i]);*

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

5. Вивести пробіл:

*printf(" ");*

*fprintf(f2, " ");*

6. Закрити файли:

*fclose(f2);*

*fclose(f3);*

## **2.8 Опис функції vel()**

Функція *vel()* знаходить рядки, які містять хоча б одне слово лише з великих літер у файлі з ім’ям *K3.txt*, виводить їх на екран та у файл з ім’ям *K2.txt*.

Аргументами функції є: рядкова змінна *name3* – назва фізичного файлу з текстовими даними, створеного у Блокноті для цієї частини роботи; *name2* – ім’я файлу з результатами відбору.

Функція має тип *void*, тобто не повертає жодного значення за допомогою оператора повернення *return*.

Локальні змінні:

*s* – рядок довжиною до 100 символів для зчитування тексту із файлу;

*result* – рядок довжиною до 100 символів для копіювання;

*word* – рядок довжиною до 20 символів для копіювання;

*f2* та *f3* – файлові змінні, за допомогою якої здійснюється доступ до файлів;

\*t – вказівник;

*k*=0, *ch*=0 – змінні для вирішення задачі.

Словесний алгоритм роботи функції:

1. Відкрити файли та перевірити, чи відкрилися файли. Якщо ні, то вивести на екран відповідне повідомлення і перервати виконання програми:

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

*if (f3 == NULL) { puts("Не вдається відкрити файл для перегляду"); return; }*

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

*if (f2 == NULL) { puts("Не вдається відкрити файл для додавання даних\n"); return; }*

Функція *fopen()* відкриває файл. Перший аргумент *name3* або *name2* – ім’я файлу, що відкривається; другий аргумент "*rt*" або "*at*" – це режим відкривання:

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

"*at*" – файл відкривається як текстовий для дописування даних у кінець.

2. Вивести повідомлення про характер розв’язуваної задачі на екран та у файл із результатами:

*puts("\nРядки, які містять хоча б одне слово лише з великих літер:");*

*fputs("\nРядки, які містять хоча б одне слово лише з великих літер:\n", f2);*

3. У циклі *while* зчитувати з файлу послідовно рядки у *s*, допоки вони є у файлі (до кінця файлу):

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

4. За допомогою функції *strcpy* копіюємо файл та ділимо його на рядки:

*strcpy(rezult, s);*

*t = strtok(s, " ,.:;/\t");*

5.У циклі *while* читаємо рядок до кінця. За допомогою функції *strcpy* копіюємо рядок. Створюємо цикл *for* та за допомогою умовного оператора *if* перевіряємо чи символ являється великою буквою, якщо так, то збільшуємо змінну *k* на 1:

*while (t != NULL)*

*{*

*strcpy(word, t);*

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

*if ('A' <= t[i] && t[i] <= 'Z' || 'А' <= t[i] && t[i] <= 'Я')*

*k++;*

6. За допомогою умовного оператору if перевіряємо чи змінна дорівнює довжині слова, якщо так, то змінна *ch*=1, Змінна *k*=0 Використати функцію *strok()* для розміщення слів:

*if (k == strlen(t))*

*ch = 1;*

*k = 0;*

*t = strtok(NULL, " ,.:;/\t");*

*}*

*7.* За допомогою умовного оператору if перевіряємо чи змінна дорівнює 1, якщо так, то виводимо слово, та обнуляємо змінну:

*if (ch)*

*{*

*cout << rezult; ch = 0;*

*fprintf(f2, "%s", rezult);*

*}*

8. Закрити файли:

*fclose(f2);*

*fclose(f3);*

## **2.9 Опис функції redag()**

Функція *redag()* 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;
* *text1* – рядок, що має довжину до 100 символів для копіювання тексту;
* *text2* – рядок, що має довжину до 100 символів для введеня тексту;
* *f3* – файлова змінна, за допомогою якої здійснюється доступ до файлу;
* *h = 0, k = 0, l = 0, v = 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.

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

*if (f3 == NULL) { puts("Не вдається відкрити файл для перегляду"); return; }*

3. 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 *l* на одиницю:

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

*l++;*

4. Створюємо динамічний масив:

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

5. Формуємо цикл *for* для зчитування даних та записуємо у тимчасовий масив:

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

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

6. Переміститися на початок файлу:

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

7. 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 *l>v*:

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

8. Копіюємо рядки:

*{ s[strlen(s) – 1] = "\0";*

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

*v++; }*

9. Закрити файл:

*fclose(f3);*

10. 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.

11. 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 >> h;*

12. Перевіряємо за допомогою *if* чи дорівнює змінна *h* одиниці:

*if (h == 1)*

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

*cout << "\nВведіть текст:\n";*

*getchar();*

*gets\_s(text2,100);*

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

*cin >> k;*

14. Формуємо цикл *for* для зчитування даних та перевіряємо чи дорівнює номер рядка тому, який ми хочемо редагувати. Якщо ні, то виводимо скопійовані дані, якщо так то виводимо змінений текст:

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

*if ( i+1 != k)*

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

*else fprintf (f3, "%s\n", text2);*

15. Закрити файл:

*fclose(f3);*

16. Якщо *h* 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(text2,100);*

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

*gets\_s(text1, 100);*

17. Формуємо цикл *for* для зчитування даних та перевіряємо чи не зустрівся нам рядок із заданим текстом, якщо ні то виводимо скопійовані дані, якщо так то виводимо нові записані дані:

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

*if (!strstr(temp[i], text1)) fprint (f3, "%s\n", temp[i]);*

*else fprintf (f3, "%s\n", text2);*

1. Закрити файл:

*fclose(f3);*

## **2.9 Опис функції vud()**

Функція *vud()* 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;
* *h = 0, k = 0, l = 0, v =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 *l* на одиницю:

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

1. Створюємо динамічний масив:

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

1. Формуємо цикл *for* для зчитування даних та записуємо у тимчасовий масив:

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

*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 *l>v*:

*while (fgets(s, 100, f3) && l>v)*

1. Копіюємо рядки:

*s[strlen(s) – 1] = "\0";*

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

*v++;*

1. Закрити файл:

*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 >> h;*

1. Перевіряємо за допомогою *if* чи змінна *h*=1:

*if (h == 1)*

1. Якщо так, то вводимо номер рядка який потрібно видалити та формує цикл *for* для зчитування даних:

*cin >> k;*

*for (int i = 0; i < l; 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 != k)*

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

1. Закрити файл:

*fclose(f3);*

1. Якщо *h* 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 та перевіряємо чи не зустрівся нам рядок із заданим текстом, якщо ні то виводимо скопійовані дані:

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

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

1. Закрити файл:

*fclose(f3);*

# 3 Результати виконання програми

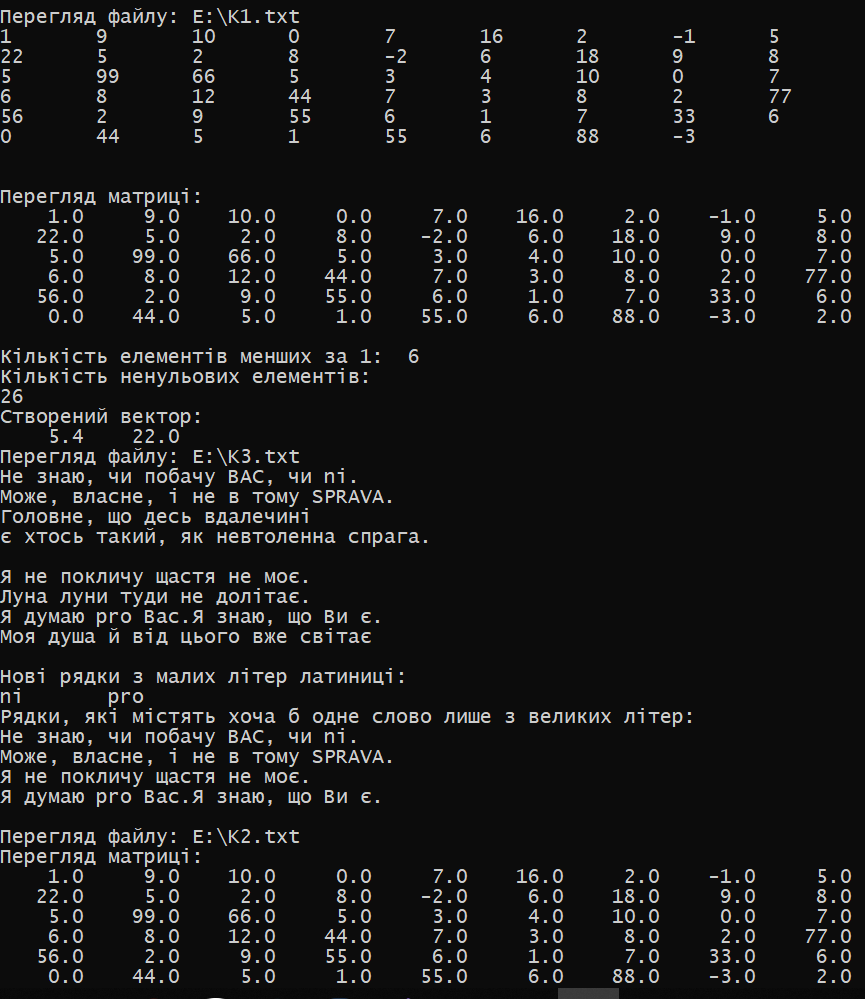


Рисунок 3.1 – Скріншот консольного вікна з результатами роботи проекту

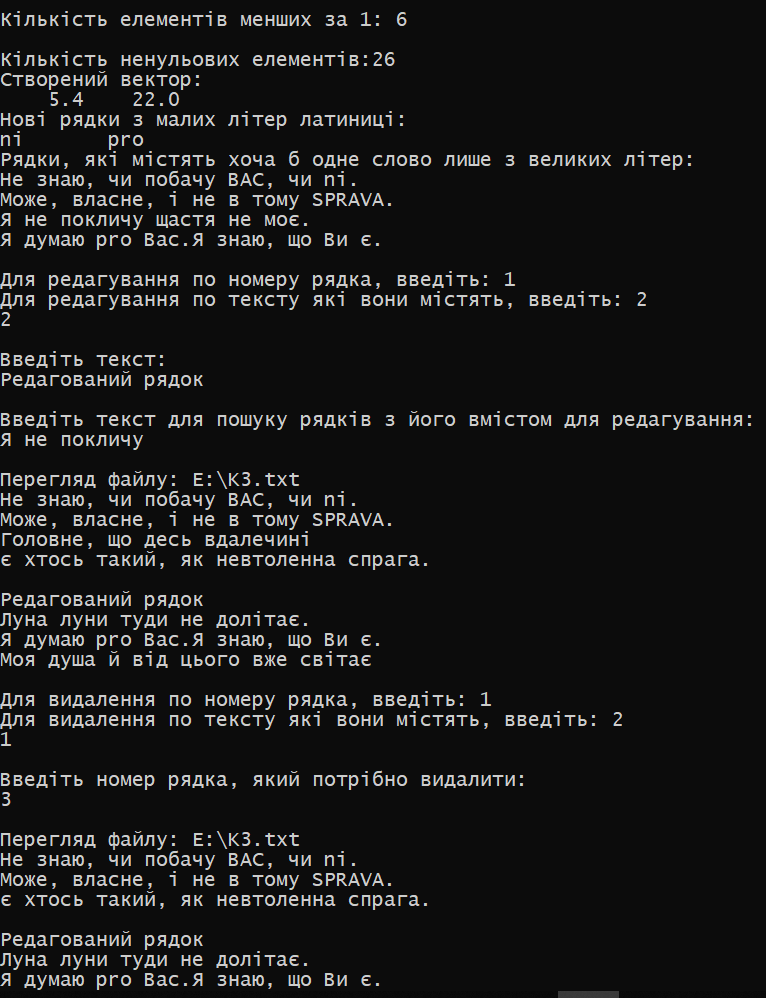


Рисунок 3.2 – Скріншот консольного вікна з результатами роботи проекту

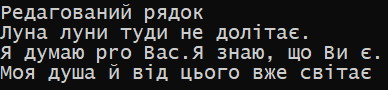


Рисунок 3.3 – Скріншот консольного вікна з результатами роботи проекту

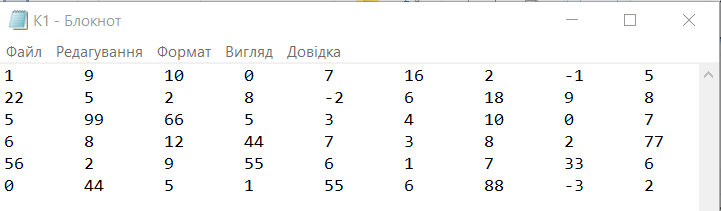


Рисунок 3.4 – Вигляд текстового файлу з вхідною матрицею числових показників вимірювань

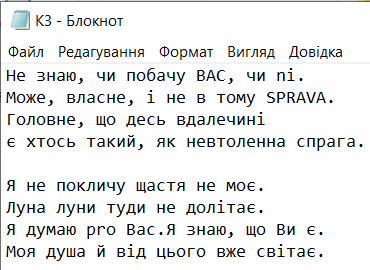


Рисунок 3.5 – Вигляд вхідного текстового файлу з текстовими даними для подальшого програмного опрацювання

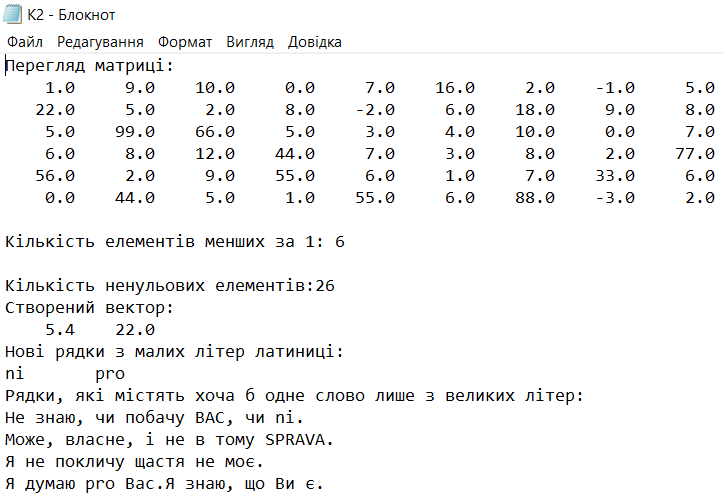


Рисунок 3.6 – Вигляд звітного текстового файлу з результатами програмного опрацювання

# 4 Інструкція з використання програми

Розроблена в роботі програма призначена для обробки текстових файлів, які містять матрицю у вигляді масиву і символьні рядки.

Щоб програма коректо працювала, пристрій має відповідати таким мінімальним системним вимогами:

* операційна система Windows 7 та вище;
* процесор з частотою 1,6 ГГц або більш потужний;
* 2 ГБ оперативної пам'яті;
* відеокарта, що підтримує мінімальну роздільну здатність дисплея 720p (1280 на 720).

У програмному коді використовуються цілі і дійсні типи даних, які використовують більшу кількість пам'яті для роботи. Так само для роботи програми необхідно створення двох текстових файлів. Ці файли повинні містити вихідні дані. Перший містить початкову матрицю, яка буде оброблюватись. Другий файл містить текстові рядки. У третій файл будуть записуватись результати роботи.

Основний функціонал нашої програми:

* перегляд вхідної матриці з файлу *K1*;
* перегляд вхідних рядкових та текстових даних з файлу *K3*.
* обчислення елементів вектору як середнє арифметичне елементів першого та останнього рядків матриці;
* обчислення значення кількості вимірювань датчиків (елементів матриці), абсолютне значення яких менше 1;
* визначення кількості ненульових елементів лише у тих рядках, які не містять від’ємних елементів;
* створення нових рядків з малих літер латиниці рядків файлу;
* виведення рядків, які містять хоча б одне слово лише з великих літер;
* редагування та видалення даних;
* запис усіх опрацьованих результатів в текстовий файл *K2.*

Для роботи програми є обов’язковим створення двох текстових файлів з назвами *K1, K3*. Вони повинні бути розташовані в кореневій папці диску E.

Як вхідні дані для опрацювання та обчислення датчиків матриці в створеному файлі *K1* потрібно записати числові дані вимірювань датчиків у вигляді матриці розмірності *M* на *N*, де *M* – кількість датчиків, *N* – кількість вимірювань.

Як вхідні дані для опрацювання текстових та символьних даних в створеному файлі *K3* потрібно записати потрібні текстові дані.

Якщо усі пункти виконані і відповідають інструкції, то програма готова до запуску, після чого ви можете переглянути результати опрацювання файлів *K1* та *K3.* Також для більшої зручності усі опрацьовані результати матриці, рядкових даних та текстових рядків будуть доступні в кореневій папці в створеному файлі *K2*.

Якщо програма правильно та точно запрацювала, то в консольному вікні буде переглянута вхідна матриця з файлу *K1.* Далі будуть зображені результати опрацювання датчиків (елементів) матриці згідно з основним функціоналом програми, а саме: результатами обчислення елементів вектору як середнє арифметичне елементів першого та останнього рядків матриці; обчислення значення кількості вимірювань датчиків (елементів матриці), абсолютне значення яких менше 1; визначення кількості ненульових елементів лише у тих рядках, які не містять від’ємних елементів;

На останок буде зображено результати опрацювання рядкових даних та текстових рядків, вхідними даними якого є файл *K3,* а саме: створення нових рядків з малих літер латиниці рядків файлу; виведення рядків, які містять хоча б одне слово лише з великих літер; редагування та видалення даних.

# Висновки з аналізом досягнутих результатів

Під час роботи над курсовою роботою було закріплено та розширено теоретичні та практичні навички і знання за фахом, які були використані під час розв’язування прикладних задач індивідуального варіанта завдання. Розвинуто навички самостійної роботи з літературою, нормативними документами, пошуку необхідної інформації, оволодінні методикою дослідження та експериментування під час розв’язування задач і питань, які розроблювались у курсовій роботі. Закріплено знання алгоритмізації та навички програмування алгоритмічною мовою при опрацюванні різнорідних даних у файлах з можливістю їх програмного редагування. Досліджено наявні сучасні програмні засоби, які дозволяють створювати програмні проєкти мовою С++. Розкрито сутність технологій мови С++ з опрацюванням різнорідних даних у файлах. Вивчено засоби мови С++ з програмного створення файлів різних форматів, записування у них даних, перегляду вмісту файлів, відбору даних за умовою.

Розроблено програмний проєкт для розв’язування завдання відповідно до індивідуального варіанта, який передбачав створення, заповнення даними, відбір та редагування файлів різних форматів. Створено текстовий файл *K1,* в який записали числові дані вимірювань датчиків у вигляді матриці розмірності *M* на *N*, де *M* – кількість датчиків, *N* – кількість вимірювань. У програмному проекті розроблено функцію, яка дозволить вивести на екран вміст текстового файлу *K1,* ім’я якого передається як аргумент із функції *main().*

Створено функції для формування числової матриці розмірності *M* на *N* зчитуванням її значень із файлу *K1;* виведення цієї матриці на екран та у файл *K2*. Створено функції для опрацювання числової матриці з обчисленням значення кількості вимірювань датчиків (елементів матриці), абсолютне значення яких менше 1; визначення кількості ненульових елементів лише у тих рядках, які не містять від’ємних елементів та результати обчислення програмно записано у файл *K2*. Створено функцію для опрацювання числової матриці з формування елементів вектору та обчислено елементи вектору як середнє арифметичне елементів першого та останнього рядків матриці та результати обчислення програмно записано у файл *K2*.

Як вхідні дані для опрацювання текстових даних створено ще один файл з ім’ям *K3* та заповнено його коректними текстовими даними. Програмно переглянули на екрані вміст текстового файлу *K3,* ім’я якого передається як аргумент із функції *main().* Розробили функцію, яка виконуватиме програмне опрацювання даних з файлу *K3,* а саме: створення нових рядків з малих літер латиниці рядків файлу; виведення рядків, які містять хоча б одне слово лише з великих літер; редагування та видалення даних. Програмно записали всі результати опрацювання текстових даних у файл *K2.* Програмно переглянули на екрані вміст текстового файлу з результатами опрацювання текстових даних.

Написані функції в програмному коді можуть використовуватись у різних сферах діяльності. Опрацювання матриць може бути корисним у роботі з таблицями та даними у сфері бухгалтерії, при опрацюванні вимірювань різнорідних датчиків під час автоматизації певних процесів тощо. Опрацювання текстових рядків може бути корисним у сфері видавництва, наприклад, для перевірки пунктуаційних помилок та їх виправлення.

# Список використаних джерел

1. Трофименко О.Г., Прокоп Ю.В., Задерейко О.В. Алгоритмізація та програмування : навч.-метод. посібник. Одеса: Фенікс, 2020. 310 с. URL: <http://dspace.onua.edu.ua/handle/11300/12345> (дата звернення 13.05.2022).
2. Windows.h. URL: <https://rsdn.org/forum/cpp.applied/1284681.hot> (дата звернення 13.05.2022).
3. Управляющие символы в программировании на C++. URL: http://cppstudio.com/post/256/ (дата звернення 14.05.2022).

# Додаток А Програмний код

#include <iostream>

#include <stdlib.h>

#include <string.h>

#include <windows.h>

#pragma warning(disable : 4996)

using namespace std;

const int M = 6, N = 9;

**//Переглянути файл**

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

{

char s[100]; FILE\* f;

f = fopen(name, "rt");

if (f == NULL) { puts("Неможливо відкрити файл для перегляду"); 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("Неможливо відкрити файл для перегляду"); 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("Неможливо відкрити файл"); return; }

printf("\n\nПерегляд матрицi:\n");

fprintf(f2, "Перегляд матрицi:\n");

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

{

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

{

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

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

}

puts("");

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

}

fclose(f2);

return;

}

**//Обчислити значення кількості елементів матриці, абсолютне значення яких менше 1**

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

{

FILE\* f2;

int i, j, h = 0;

f2 = fopen(name2, "at");

if (f2 == NULL)

{

puts("Не вдається відкрити файл для додавання даних\n"); return;

}

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

{

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

{

if (matr[i][j] < 1) h++;

}

}

printf("\nКількість елементів менших за 1: %i\t", h);

fprintf(f2, "\nКількість елементів менших за 1: %i\n", h);

fclose(f2);

}

**/\*Визначити кількість ненульових елементів лише у тих рядках, які не містять від’ємних елементів\*/**

**void kilkist2(double matr[M][N], char\* name2)**

{

FILE\* f2;

f2 = fopen(name2, "at");

if (f2 == NULL)

{

puts("Не вдається відкрити файл для додавання даних\n"); return;

}

puts("\nКількість ненульових елементів:");

fprintf(f2, "\nКількість ненульових елементів:");

int kol = 0, h = 0;

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

{

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

{

if (matr[i][j] < 0) h = i;

}

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

{

if (i != h && matr[i][j] != 0) kol++;

}

}

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

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

fclose(f2);

}

**/\*Обчислити елементи вектора як середнє арифметичне елементів першого та останнього рядків матриці\*/**

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

{

double sum = 0, ser = 0; int p = 0;

FILE\* f2;

f2 = fopen(name2, "at");

if (f2 == NULL)

{

puts("Не вдається відкрити файл для додавання даних\n"); return;

}

puts("\nСтворений вектор:");

fprintf(f2, "\nСтворений вектор:\n");

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

{

sum = 0, ser = 0;

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

{

if (i == 0 || i == M - 1)

{

sum += matr[i][j]; p = 1;

}

}

if (i == 0 || i == M - 1)

{

if (p)

{

ser = sum / N;

vekt[i] = ser;

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

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

}

}

}

fclose(f2);

}

**//Створити нові рядки з малих літер латиниці рядків файлу**

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

{

FILE\* f3, \* f2;

char s[100];

f3 = fopen(name3, "rt");

if (f3 == NULL) { puts("Не вдається відкрити файл для перегляду"); return; }

f2 = fopen(name2, "at");

if (f2 == NULL) { puts("Не вдається відкрити файл для додавання даних\n"); return; }

puts("\nНові рядки з малих літер латиниці:");

fputs("\nНові рядки з малих літер латиниці:\n", f2);

while (fgets(s, 100, f3))

{

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

{

if (s[i] >= 'a' && s[i] <= 'z')

{

printf("%c", s[i]);

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

}

}

printf(" ");

fprintf(f2, " ");

}

fclose(f2);

fclose(f3);

}

**//Вивести рядки, які містять хоча б одне слово лише з великих літер**

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

{

FILE\* f3, \* f2;

f3 = fopen(name3, "rt");

if (f3 == NULL) { puts("Не вдається відкрити файл для перегляду"); return; }

f2 = fopen(name2, "at");

if (f2 == NULL) { puts("Не вдається відкрити файл для додавання даних\n"); return; }

puts("\nРядки, які містять хоча б одне слово лише з великих літер:");

fputs("\nРядки, які містять хоча б одне слово лише з великих літер:\n", f2);

char s[100], rezult[100], word[20], \* t; int k = 0;

bool ch = 0;

while (fgets(s, 100, f3))

{

strcpy(rezult, s);

t = strtok(s, " ,.:;/\t");

while (t != NULL)

{

strcpy(word, t);

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

if ('A' <= t[i] && t[i] <= 'Z' || 'А' <= t[i] && t[i] <= 'Я')

k++;

if (k == strlen(t))

{

ch = 1;

k = 0;

t = strtok(NULL, " ,.:;/\t");

}

}

if (ch)

{

cout << rezult; ch = 0;

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

}

}

fclose(f2);

fclose(f3);

}

**int main()**

{

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

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

char name1[] = "E:\\K1.txt";

char name2[] = "E:\\K2.txt";

char name3[] = "E:\\K3.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 значення кількості елементів матриці, абсолютне значення яких менше 1\*/

kilkist(A, name2)

/\*Визначити кількість ненульових елементів лише у тих рядках, які не містять від’ємних елементів\*/

kilkist2(A, name2);

/\*Обчислити елементи вектора як середнє арифметичне елементів першого та останнього рядків матриці\*/

create\_vector(A, B, name2);

//Переглянути файл

view\_file(name3);

//Створити нові рядки з малих літер латиниці рядків файлу

mal(name3, name2);

//Вивести рядки, які містять хоча б одне слово лише з великих літер

vel(name3, name2);

//Переглянути файл

view\_file(name2);

}

# Додаток Б Схеми алгоритмів программи

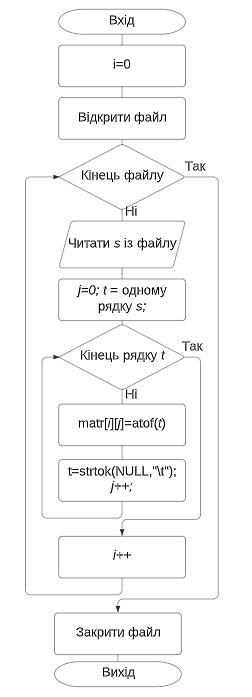
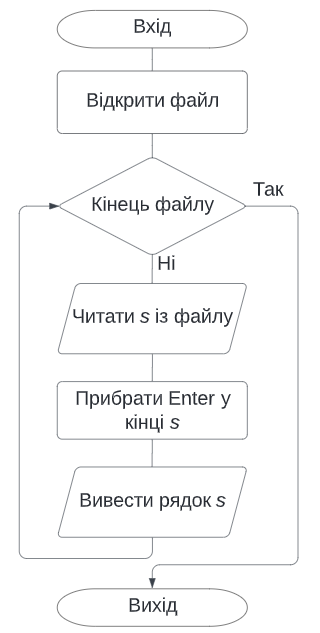


Рисунок Б.1 – Схема алгоритму Рисунок Б.2 – Схема алгоритму

функції *view\_file* функції *create\_matrix*

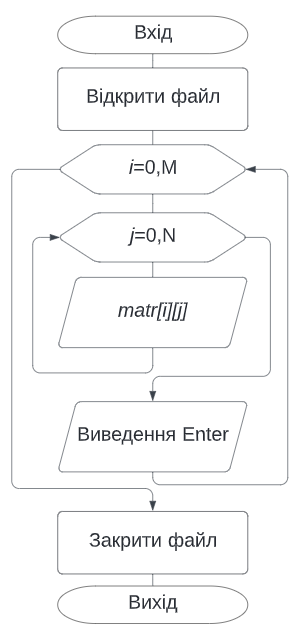
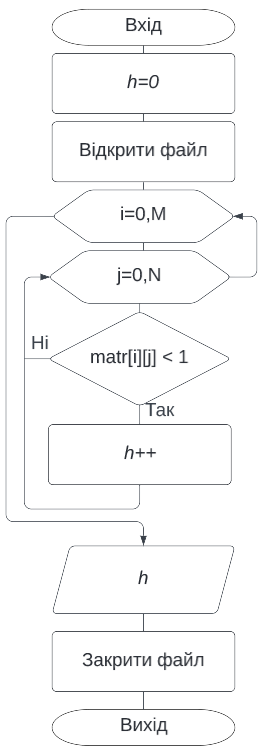
 ****

Рисунок Б.3 – Схема алгоритму Рисунок Б.4 – Схема алгоритму

функції *output\_matrix* функції *kilkist*

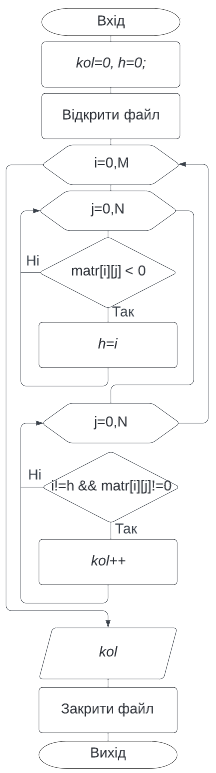
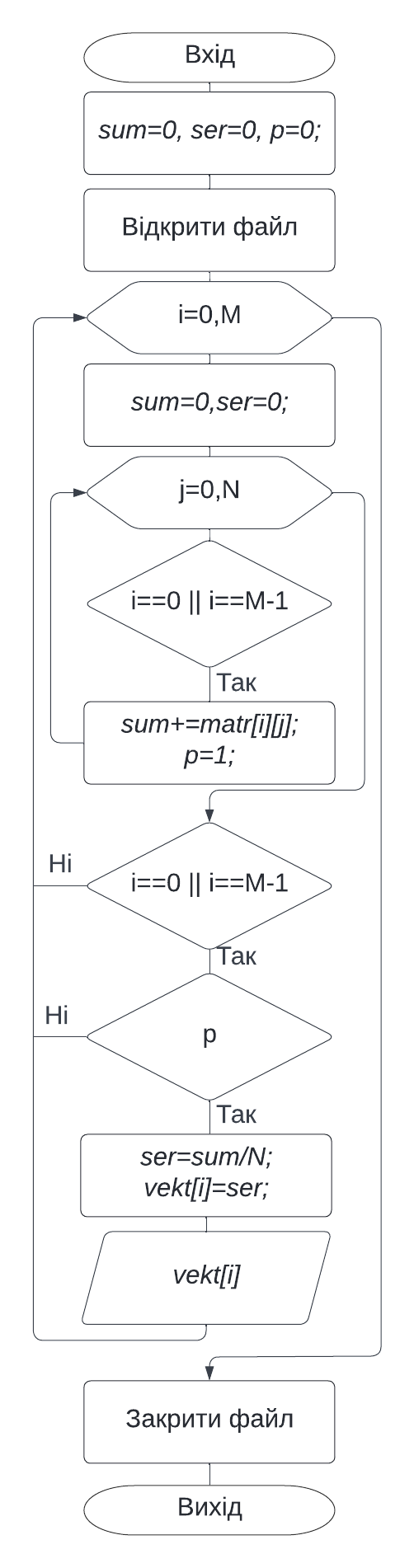
 

Рисунок Б.5 – Схема алгоритму Рисунок Б.6 – Схема алгоритму

функції *kilkist2* функції *create\_vector*

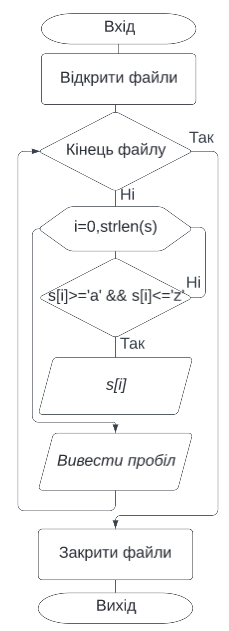
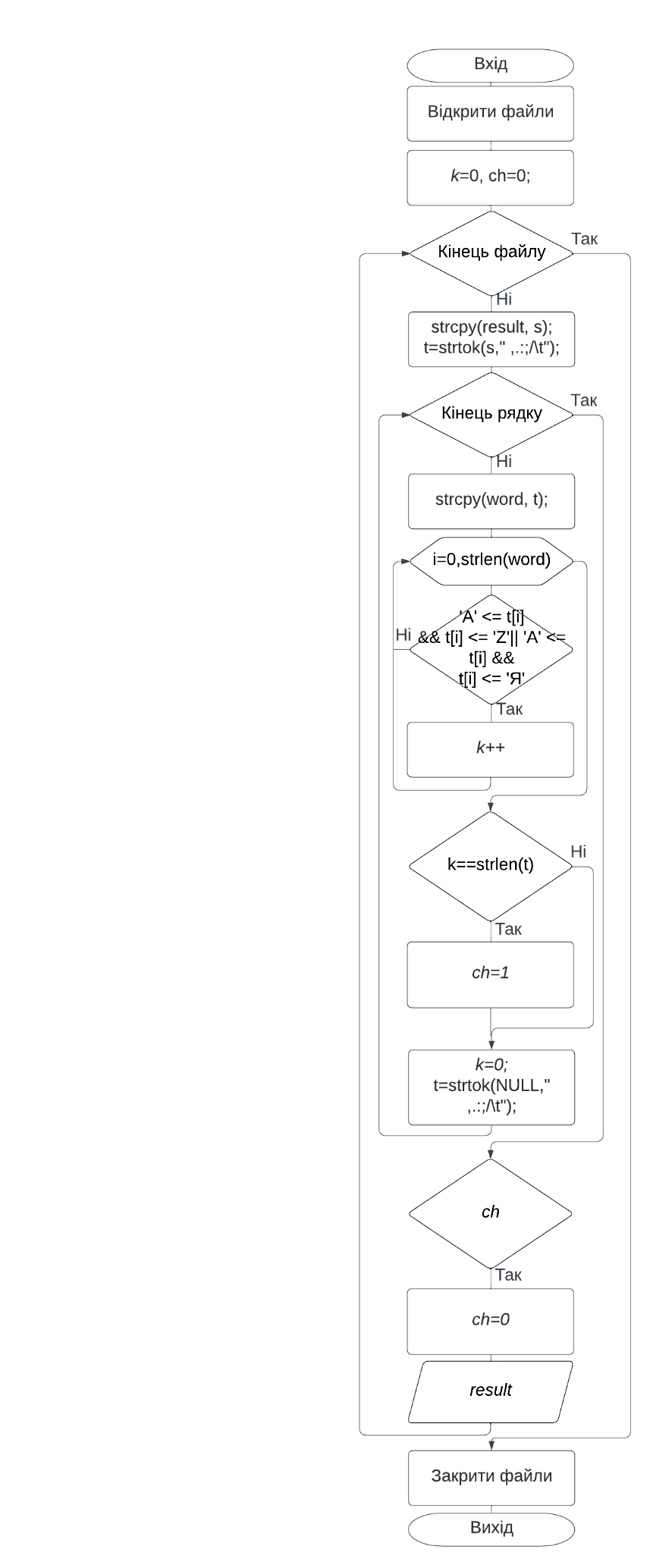
 

Рисунок Б.7 – Схема алгоритму Рисунок Б.8 – Схема алгоритму

функції *mal* функції *vel*

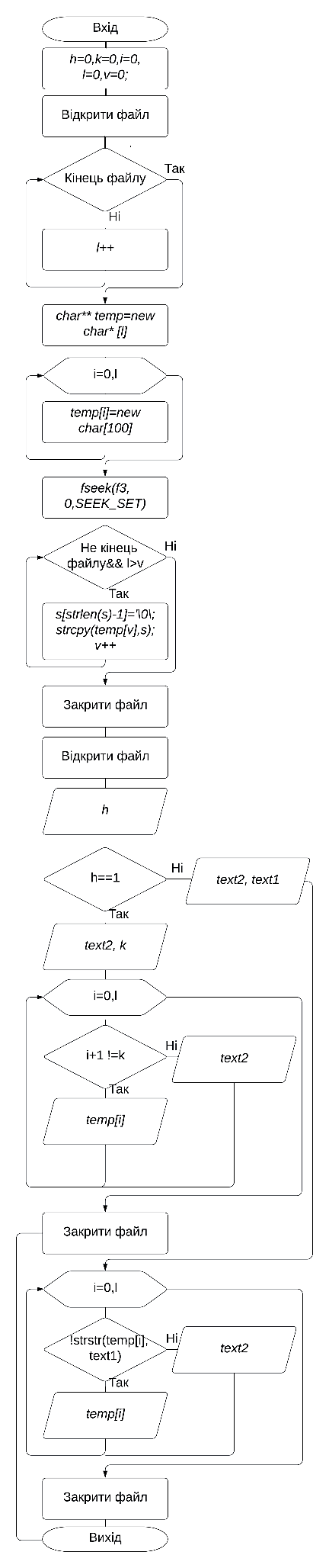
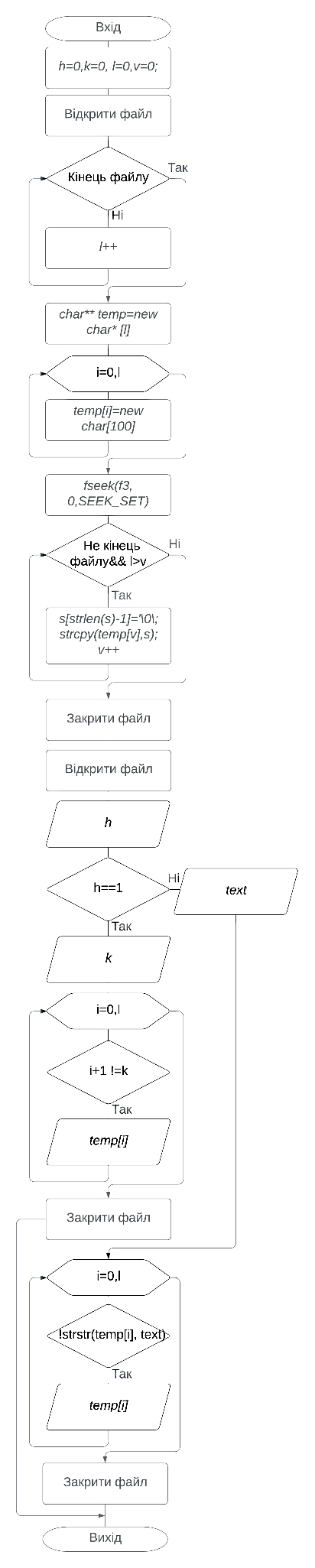
** 

Рисунок Б.9 – Схема алгоритму Рисунок Б.10 – Схема алгоритму

функції *redag* функції *vud*

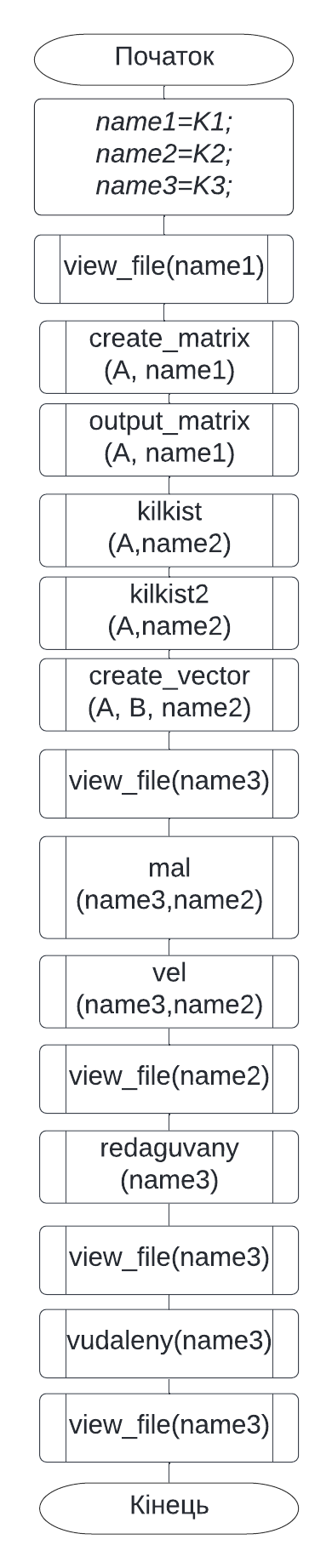


Рисунок Б.11 – Схема алгоритму

функції *main*