Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту


# Звіт

**про виконання лабораторних та практичних робіт блоку № 5**

**На тему: “Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.”**

**З дисципліни:** «Основи програмування»

**до:**

Практичних Робіт до блоку № 5

**Виконав:**

Студент групи ШІ-11

Голейчук Іван Миколайович

Львів 2024

**Тема роботи**:
"Основи роботи з файлами та рядковими змінними у програмуванні на C++"

**Мета роботи**:
Розробити системне розуміння принципів роботи з файлами та рядковими змінними у мові C++. Дослідити методи взаємодії з текстовими та бінарними файлами, а також функціональність стандартної бібліотеки для обробки файлів. Ознайомитися зі створенням власних бібліотек, організацією коду, та вивчити практичні прийоми парсингу, серіалізації даних і обробки помилок.

**Теоретичні відомості:**

Вступ до Роботи з Файлами:
Основні операції з файлами: відкриття, читання, запис, закриття
Робота з файловими дескрипторами
C-style читання з файлу та запис до файлу
Перевірка стану файлу: перевірка помилок, кінець файлу
Базові приклади читання та запису в файл
Символи і Рядкові Змінні:
Робота з char та string: основні операції і методи
Стрічкові літерали та екранування символів
Конкатенація, порівняння та пошук у рядках
Текстові Файли:
Особливості читання та запису текстових файлів
Обробка рядків з файлу: getline, ignore, peek
Форматування тексту при записі: setw, setfill, setprecision
Парсинг текстових файлів: розділення на слова, аналіз структури
Обробка помилок при роботі з файлами
Бінарні Файли:
Вступ до бінарних файлів: відмінності від текстових, приклади (великі дані, ігрові ресурси, зображення)
Читання та запис бінарних даних
Робота з позиціонуванням у файлі: seekg, seekp
Серіалізація об'єктів у бінарний формат
Стандартна бібліотека та робота з файлами:
Огляд стандартної бібліотеки для роботи з файлами
Потоки вводу/виводу: ifstream, ofstream, fstream
Обробка помилок при роботі з файлами
Створення й використання бібліотек:
Вступ до створення власних бібліотек у C++
Правила розбиття коду на header-и(.h) та source(.cpp) файли
Статичні проти динамічних бібліотек: переваги та використання
Інтерфейси бібліотек: створення, документування, версіонування
Використання сторонніх бібліотек у проектах

# Індивідуальний план опрацювання теорії:

## Вступ до Роботи з Файлами:

**Опрацьовано:**
Розглянуто основні операції роботи з файлами: відкриття, читання, запис, закриття. Вивчено принципи роботи з файловими дескрипторами та виконання C-style операцій з файлами. Проведено аналіз перевірки стану файлів (помилки, кінець файлу) з демонстрацією базових прикладів.
**Джерела інформації:**

- Лекції Олександра Пшеничного
- Практичні заняття
- Використання штучного інтелекту (чат GPT)
- YouTube

## Символи і Рядкові Змінні:

**Опрацьовано:**
Вивчено основні операції з типами даних `char` і `string`, включаючи роботу зі стрічковими літералами, екрануванням символів, конкатенацією, порівнянням та пошуком у рядках. Розглянуто приклади ефективної роботи з рядковими змінними.
**Джерела інформації:**

- Лекції Олександра Пшеничного
- Практичні заняття
- Використання штучного інтелекту (чат GPT)
- YouTube

## Текстові Файли:

**Опрацьовано:**
Досліджено специфіку роботи з текстовими файлами: методи читання (`getline`, `ignore`, `peek`) і запису даних, використання функцій форматування тексту (`setw`, `setfill`, `setprecision`). Вивчено техніки парсингу файлів для аналізу їхньої структури та обробки рядків. Здійснено практику перевірки помилок при роботі з файлами.
**Джерела інформації:**

- Лекції Олександра Пшеничного
- Практичні заняття
- Використання штучного інтелекту (чат GPT)
- YouTube

## Бінарні Файли:

**Опрацьовано:**
Вивчено основи роботи з бінарними файлами, їх відмінності від текстових. Ознайомлено з методами читання, запису, серіалізації даних і позиціонування у файлі (`seekg`, `seekp`). Наведено приклади використання бінарних файлів для збереження великих даних.
**Джерела інформації:**

- Лекції Олександра Пшеничного
- Практичні заняття
- Використання штучного інтелекту (чат GPT)
- YouTube

## Стандартна бібліотека та робота з файлами:

**Опрацьовано:**
Досліджено можливості стандартної бібліотеки для роботи з файлами: використання потоків (`ifstream`, `ofstream`, `fstream`), обробка помилок, а також інтеграція сторонніх бібліотек у проєкти.
**Джерела інформації:**

- Лекції Олександра Пшеничного
- Практичні заняття
- Використання штучного інтелекту (чат GPT)
- YouTube

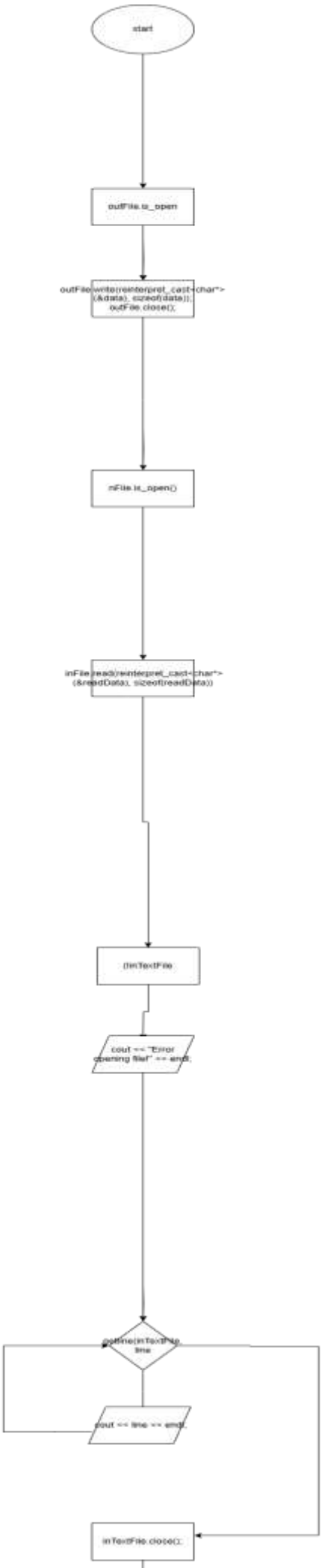
## Створення й використання бібліотек:

**Опрацьовано:**
Розглянуто основи створення власних бібліотек у C++, правила розподілу коду між файлами (`header` і `source`). Вивчено переваги статичних і динамічних бібліотек, принципи документування та версіонування інтерфейсів, інтеграцію бібліотек у проєкти.
**Джерела інформації:**

- Лекції Олександра Пшеничного
- Практичні заняття
- Використання штучного інтелекту (чат GPT)
- YouTube

**Виконання роботи:**

## Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-9)


start
outFile.is_open
outFile.write(reinterpret_cast<char*>
(&data), sizeof(data));
outFile.close();
inFile.is_open()
inFile.read(reinterpret_cast<char*>
(&readData), sizeof(readData))
!inTextFile
cout << "Error
opening file!" << endl;
getline(inTextFile,
line
cout << line << endl;
inTextFile.close();

Epic 5 Task 3 - Lab# programming: VNS Lab 6

23. Для рядка знайти довжину найкоротшого слова.

```cpp
#include <iostream>
#include <cstring>
#include <climits>

using namespace std;

int main() {

    char s[256];
    cout << "Enter the text: ";
    cin.getline(s, 256);

    int minlen = INT_MAX, curlen = 0;

    for (int i = 0; s[i] != '\0'; i++) {
        if (s[i] != ' ' && s[i] != '.') {
            curlen++;
        }
        else if (curlen > 0) {
            if (curlen < minlen) minlen = curlen;
            curlen = 0;
        }
    }

    if (minlen == INT_MAX) {
        cout << "There are no words in the text." << endl;
    }
    else {
        cout << "The length of the shortest word is: " << minlen << endl;
    }

    return 0;
}
```

Epic 5 Task 4 - Lab# programming: VNS Lab 8

23. Структура "Стадіон":

- назва;
- рік будівлі;
- кількість площадок;
- види спорту.

Знищити всі елементи, у яких рік будівлі меншее заданого, додати 2 елементи перед елементом із зазначеним номером.

```
#include <iostream>

#include <fstream>
#include <vector>
#include <string>

using namespace std;

struct stadium {
    string name;
    int year;
    int fields;
    string sports;
};

void create(const string& fname, const vector<stadium>& data) {
    ofstream f(fname, ios::binary);
    if (!f) {
        cerr << "File opening error" << endl;
        return;
    }
    for (const auto& s : data) {
        f.write((char*)&s, sizeof(stadium));
    }
    f.close();
}

void print(const string& fname) {
    ifstream f(fname, ios::binary);
    if (!f) {
        cerr << "File opening error" << endl;
        return;
    }
    stadium s;
    while (f.read((char*)&s, sizeof(stadium))) {
        cout << "Name: " << s.name << ", Year: " << s.year
            << ", Fields: " << s.fields
            << ", Sports: " << s.sports << endl;
```

```
    }
    f.close();
}

void del(const string& fname, int y) {
    ifstream f(fname, ios::binary);
    if (!f) {
        cerr << "File opening error" << endl;
        return;
    }
    vector<stadium> data;
    stadium s;
    while (f.read((char*)&s, sizeof(stadium))) {
        if (s.year >= y) {
            data.push_back(s);
        }
    }
    f.close();
    ofstream out(fname, ios::binary | ios::trunc);
    if (!out) {
        cerr << "File opening error" << endl;
        return;
    }
    for (const auto& item : data) {
        out.write((char*)&item, sizeof(stadium));
    }
    out.close();
}

void add(const string& fname, const stadium& e1, const stadium& e2, int pos) {
    ifstream f(fname, ios::binary);
    if (!f) {
        cerr << "File opening error" << endl;
        return;
    }
    vector<stadium> data;
    stadium s;
    while (f.read((char*)&s, sizeof(stadium))) {
        data.push_back(s);
    }
    f.close();
    if (pos < 0 || pos > data.size()) {
        cerr << "Invalid position" << endl;
        return;
    }
    data.insert(data.begin() + pos, { e1, e2 });
    ofstream out(fname, ios::binary | ios::trunc);
    if (!out) {
        cerr << "File opening error" << endl;
        return;
    }
```

```
    for (const auto& item : data) {
        out.write((char*)&item, sizeof(stadium));
    }
    out.close();
}

int main() {
    string fname = "stadiums.dat";
    vector<stadium> data = {
        {"Olympic", 1990, 5, "Football, Basketball"},
        {"Lviv Arena", 2012, 3, "Football"},
        {"Dnipro Arena", 1985, 4, "Football, Volleyball"}
    };

    create(fname, data);
    cout << "File content after creation:" << endl;
    print(fname);

    int y = 2000;
    del(fname, y);
    cout << "\nFile content after deletion:" << endl;
    print(fname);

    stadium e1 = { "New Stadium 1", 2023, 6, "Football, Tennis" };
    stadium e2 = { "New Stadium 2", 2024, 8, "Basketball, Hockey" };
    int pos = 1;
    add(fname, e1, e2, pos);
    cout << "\nFile content after addition:" << endl;
    print(fname);

    return 0;
}
```

John Black - Epic 5 Task 5 - Lab# programming: VNS Lab 9

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <map>

using namespace std;

bool has_dup(const string& s) {
    istringstream iss(s);
```

```
    map<string, int> wc;
    string w;
    while (iss >> w) {
        wc[w]++;
        if (wc[w] > 1) return true;
    }
    return false;
}

int max_a(const string& s) {
    istringstream iss(s);
    string w;
    int max_a = 0, pos = -1, cur = 0;
    while (iss >> w) {
        cur++;
        int cnt = 0;
        for (char c : w) {
            if (tolower(c) == 'a') cnt++;
        }
        if (cnt > max_a) {
            max_a = cnt;
            pos = cur;
        }
    }
    return pos;
}

int main() {
    string f1 = "F1.txt", f2 = "F2.txt";

    ofstream out(f1);
    if (!out) {
        cerr << "Error opening F1" << endl;
        return 1;
    }
    out << "This is a test line with two test words.\n";
    out << "Another line for verification.\n";
    out << "This line has no duplicate words.\n";
    out << "Here is a line with many words, many words here.\n";
    out << "This line contains the word aaaa many times.\n";
    out << "Just a meaningless line.\n";
    out << "This is another line with text.\n";
    out << "This line contains words words words.\n";
    out << "A single line.\n";
    out << "A line for testing word verification.\n";
    out.close();

    ifstream in(f1);
    ofstream out2(f2);
    if (!in || !out2) {
        cerr << "File error" << endl;
```

```
        return 1;
    }
    string line;
    while (getline(in, line)) {
        if (has_dup(line)) {
            out2 << line << endl;
        }
    }
    in.close();
    out2.close();

    in.open(f1);
    if (!in) {
        cerr << "Error opening F1" << endl;
        return 1;
    }

    int max_pos = -1, line_num = 0, cur_line = 0;
    while (getline(in, line)) {
        cur_line++;
        int pos = max_a(line);
        if (pos > max_pos) {
            max_pos = pos;
            line_num = cur_line;
        }
    }
    in.close();

    cout << "Line number with the word having the most 'A's: " << line_num <<
endl;

    return 0;
}
```

23.

1) Скопіювати з файлу F1 у файл F2 всі рядки, у яких міститься два однакових слова.
2) Визначити номер слова, у якому найбільше букв «А».

Epic 5 Task 6 - Lab# programming: Algotester Lab 4

# Lab 4v1

*Limits: 1 sec., 256 MiB*

Вам дано 2 цілих чисел масиви, розміром $N$ та $M$.

Ваше завдання вивести:

1. Різницю N-M
2. Різницю M-N
3. Їх перетин
4. Їх обеднання
5. Їх симетричну різницю

## Input

У першому рядку ціле число $N$ - розмір масиву 1

У другому рядку $N$ цілих чисел - елементи масиву 1

У третьому рядку ціле число $M$ - розмір масиву 2

У четвертом рядку $M$ цілих чисел - елементи масиву 2

## Output

Вивести результат виконання 5 вищезазначених операцій у форматі:

У першому рядку ціле число $N$ - розмір множини

У наступному рядку $N$ цілих чисел - посортована у порядку зростання множина

```cpp
#include <iostream>
#include <set>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    int N, M;

    cin >> N;
    vector<int> a(N);
    for (int i = 0; i < N; i++) {
        cin >> a[i];
    }

    cin >> M;
    vector<int> b(M);
    for (int i = 0; i < M; i++) {
        cin >> b[i];
    }

    sort(a.begin(), a.end());
    sort(b.begin(), b.end());
```

```
    vector<int> diff;
    set_difference(a.begin(), a.end(), b.begin(), b.end(), back_inserter(diff));
    cout << diff.size() << endl;
    for (size_t i = 0; i < diff.size(); i++) {
        if (i > 0) cout << " ";
        cout << diff[i];
    }
    cout << endl;

    vector<int> revDiff;
    set_difference(b.begin(), b.end(), a.begin(), a.end(),
back_inserter(revDiff));
    cout << revDiff.size() << endl;
    for (size_t i = 0; i < revDiff.size(); i++) {
        if (i > 0) cout << " ";
        cout << revDiff[i];
    }
    cout << endl;

    vector<int> intersec;
    set_intersection(a.begin(), a.end(), b.begin(), b.end(),
back_inserter(intersec));
    cout << intersec.size() << endl;
    for (size_t i = 0; i < intersec.size(); i++) {
        if (i > 0) cout << " ";
        cout << intersec[i];
    }
    cout << endl;

    vector<int> unio;
    set_union(a.begin(), a.end(), b.begin(), b.end(), back_inserter(unio));
    cout << unio.size() << endl;
    for (size_t i = 0; i < unio.size(); i++) {
        if (i > 0) cout << " ";
        cout << unio[i];
    }
    cout << endl;

    vector<int> symDiff;
    set_symmetric_difference(a.begin(), a.end(), b.begin(), b.end(),
back_inserter(symDiff));
    cout << symDiff.size() << endl;
    for (size_t i = 0; i < symDiff.size(); i++) {
        if (i > 0) cout << " ";
        cout << symDiff[i];
    }
    cout << endl;

    return 0;
}
```

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void print(const vector<int>& res) {
    cout << res.size() << endl;
    for (int x : res) cout << x << " ";
    cout << endl;
}

vector<int> un(const vector<int>& a, const vector<int>& b) {
    vector<int> res;
    int i = 0, j = 0;
    while (i < a.size() && j < b.size()) {
        if (a[i] < b[j]) res.push_back(a[i++]);
        else if (b[j] < a[i]) res.push_back(b[j++]);
        else {
            res.push_back(a[i]);
            i++; j++;
        }
    }
    while (i < a.size()) res.push_back(a[i++]);
    while (j < b.size()) res.push_back(b[j++]);
    return res;
}

vector<int> inter(const vector<int>& a, const vector<int>& b) {
    vector<int> res;
    int i = 0, j = 0;
    while (i < a.size() && j < b.size()) {
        if (a[i] < b[j]) i++;
        else if (b[j] < a[i]) j++;
        else {
            res.push_back(a[i]);
            i++; j++;
        }
    }
    return res;
}

vector<int> diff(const vector<int>& a, const vector<int>& b) {
    vector<int> res;
    int i = 0, j = 0;
    while (i < a.size() && j < b.size()) {
        if (a[i] < b[j]) res.push_back(a[i++]);
        else if (b[j] < a[i]) j++;
        else {
            i++;
```

```
            j++;
        }
    }
    while (i < a.size()) res.push_back(a[i++]);
    return res;
}

vector<int> symDiff(const vector<int>& a, const vector<int>& b) {
    vector<int> res1 = diff(a, b);
    vector<int> res2 = diff(b, a);
    return un(res1, res2);
}

int main() {
    int N, M;
    cin >> N;
    vector<int> n(N);
    for (int i = 0; i < N; i++) {
        cin >> n[i];
    }
    cin >> M;
    vector<int> m(M);
    for (int i = 0; i < M; i++) {
        cin >> m[i];
    }
    sort(n.begin(), n.end());
    sort(m.begin(), m.end());

    vector<int> d1 = diff(n, m);
    vector<int> d2 = diff(m, n);
    vector<int> interRes = inter(n, m);
    vector<int> unRes = un(n, m);
    vector<int> symRes = symDiff(n, m);

    print(d1);
    print(d2);
    print(interRes);
    print(unRes);
    print(symRes);

    return 0;
}
```

Epic 5 Task 7 - Lab# programming: Algotester Lab 6

## Lab 6v1

*Limits: 2 sec., 256 MiB*

Вам дано $N$ слів та число $K$.

Ваше завдання перечислити букви в словах, які зустрічаються в тексті більше-рівне ніж $K$ разів (саме слово, не буква!).

Великі та маленькі букви вважаються однаковими, виводити необхідно малі, посортовані від останьої до першої у алфавіті. Букву потрібно виводити лише один раз.

У випадку якщо таких букв немає - вивести "Empty!".

### Input

Цілі числа $N$ та $K$ - загальна кількість слів та мінімальна кількість слів щоб враховувати букви цього слова в результаті.

$N$ стрічок $s$

### Output

У першому рядку ціле число $M$ - кількість унікальних букв

У другому рядку унікальні букви через пробіли

### Constraints

$1 \leq K \leq N \leq 10^5$

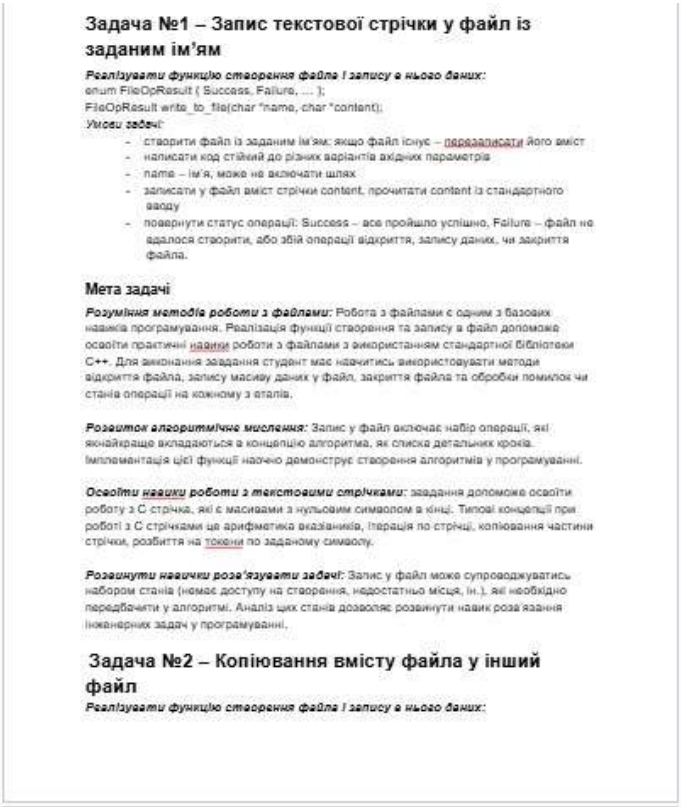
$1 \leq |s_i| \leq 10$

$s_i \in a..Z$

```cpp
#include <iostream>
#include <string>
#include <unordered_map>
#include <set>
#include <vector>
#include <algorithm>
using namespace std;

void toLower(string& str) {
    for (char& c : str) {
        c = tolower(c);
    }
}

int main() {
    int N, K;
    cin >> N >> K;

    unordered_map<string, int> freq;
    vector<string> words(N);
    for (int i = 0; i < N; i++) {
        cin >> words[i];
        toLower(words[i]);
        freq[words[i]]++;
    }

    set<char> letters;
```

```
    for (const auto& [word, count] : freq) {
        if (count >= K) {
            for (char c : word) {
                letters.insert(c);
            }
        }
    }

    if (letters.empty()) {
        cout << "Empty!" << endl;
        return 0;
    }

    vector<char> sortedLetters(letters.begin(), letters.end());
    sort(sortedLetters.rbegin(), sortedLetters.rend());

    cout << sortedLetters.size() << endl;
    for (size_t i = 0; i < sortedLetters.size(); i++) {
        cout << sortedLetters[i];
        if (i != sortedLetters.size() - 1) {
            cout << " ";
        }
    }
    cout << endl;

    return 0;
}
```

Epic 5 Task 8 - Practice# programming: Class Practice Tasк



### Задача №1 – Запис текстової стрічки у файл із заданим ім'ям

***Реалізувати функцію створення файла і запису в нього даних:***
enum FileOpResult { Success, Failure, ... };
FileOpResult write_to_file(char *name, char *content);
***Умови задачі:***

- створити файл із заданим ім'ям: якщо файл існує – перезаписати його вміст
- написати код стійкий до різних варіантів вхідних параметрів
- name – ім'я, може не включати шлях
- записати у файл вміст стрічки content, прочитати content із стандартного вводу
- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, запису даних, чи закриття файла.

### Мета задачі

***Розуміння методів роботи з файлами:*** Робота з файлами є одним з базових навиків програмування. Реалізація функції створення та запису в файл допоможе освоїти практичні навики роботи з файлами з використанням стандартної бібліотеки C++. Для виконання завдання студент має навчитись використовувати методи відкриття файла, запису масиву даних у файл, закриття файла та обробки помилок чи станів операції на кожному з етапів.

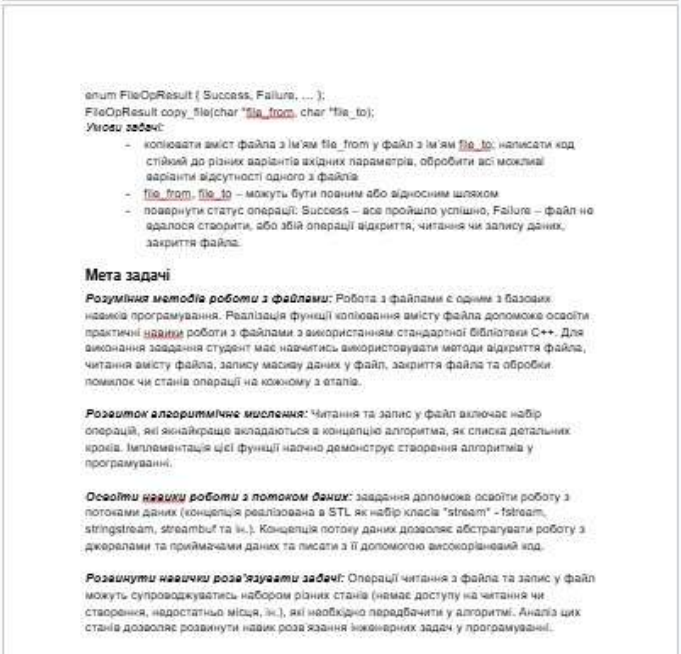
***Розвиток алгоритмічне мислення:*** Запис у файл включає набір операцій, які якнайкраще вкладаються в концепцію алгоритма, як списка детальних кроків. Імплементація цієї функції наочно демонструє створення алгоритмів у програмуванні.

***Освоїти навики роботи з текстовими стрічками:*** завдання допоможе освоїти роботу з C стрічка, які є масивами з нульовим символом в кінці. Типові концепції при роботі з C стрічками це арифметика вказівників, Iтерація по стрічці, копіювання частини стрічки, розбиття на токени по заданому символу.

***Розвинути навички розв'язувати задачі:*** Запис у файл може супроводжуватись набором станів (немає доступу на створення, недостатньо місця, iн.), які необхідно передбачити у алгоритмі. Аналіз цих станів дозволяє розвинути навик розв'язання інженерних задач у програмуванні.

### Задача №2 – Копіювання вмісту файла у інший файл

***Реалізувати функцію створення файла і запису в нього даних:***



enum FileOpResult { Success, Failure, ... };
FileOpResult copy_file(char *file_from, char *file_to);
***Умови задачі:***

- копіювати вміст файла з ім'ям file_from у файл з ім'ям file_to; написати код стійкий до різних варіантів вхідних параметрів, обробити всі можливі варіанти відсутності одного з файлів
- file_from, file_to – можуть бути повним або відносним шляхом
- повернути статус операції: Success – все пройшло успішно, Failure – файл не вдалося створити, або збій операції відкриття, читання чи запису даних, закриття файла.

### Мета задачі

***Розуміння методів роботи з файлами:*** Робота з файлами є одним з базових навиків програмування. Реалізація функції копіювання вмісту файла допоможе освоїти практичні навики роботи з файлами з використанням стандартної бібліотеки C++. Для виконання завдання студент має навчитись використовувати методи відкриття файла, читання вмісту файла, запису масиву даних у файл, закриття файла та обробки помилок чи станів операції на кожному з етапів.

***Розвиток алгоритмічне мислення:*** Читання та запис у файл включає набір операцій, які якнайкраще вкладаються в концепцію алгоритма, як списка детальних кроків. Імплементація цієї функції наочно демонструє створення алгоритмів у програмуванні.

***Освоїти навики роботи з потоком даних:*** завдання допоможе освоїти роботу з потоками даних (концепція реалізована в STL як набір класів "stream" - fstream, stringstream, streambuf та iн.). Концепція потоку даних дозволяє абстрагувати роботу з джерелами та приймачами даних та писати з її допомогою високорівневий код.

***Розвинути навички розв'язувати задачі:*** Операції читання з файла та запис у файл можуть супроводжуватись набором різних станів (немає доступу на читання чи створення, недостатньо місця, iн.), які необхідно передбачити у алгоритмі. Аналіз цих станів дозволяє розвинути навик розв'язання інженерних задач у програмуванні.

```cpp
#include <iostream>
#include <fstream>
#include <cstring>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult write_to_file(const char* name, const char* content) {
    ofstream file(name);
```

```
    if (!file.is_open()) {
        return Failure;
    }

    file << content;

    if (file.fail()) {
        return Failure;
    }

    file.close();

    if (file.fail()) {
        return Failure;
    }

    return Success;
}

int main() {
    char filename[100], content[500];

    cout << "Enter the filename: ";
    cin.getline(filename, 100);

    cout << "Enter the content to write to the file: ";
    cin.getline(content, 500);

    FileOpResult result = write_to_file(filename, content);

    if (result == Success) {
        cout << "File written successfully!" << endl;
    }
    else {
        cout << "Error writing file!" << endl;
    }

    return 0;
}
```

```
#include <iostream>
#include <fstream>

using namespace std;

enum FileOpResult { Success, Failure };

FileOpResult copy_file(const char* file_from, const char* file_to) {
    ifstream src(file_from, ios::binary);
```

```
    if (!src.is_open()) {
        return Failure;
    }

    ofstream dest(file_to, ios::binary);
    if (!dest.is_open()) {
        return Failure;
    }

    dest << src.rdbuf();

    if (src.fail() || dest.fail()) {
        return Failure;
    }

    src.close();
    dest.close();

    if (src.fail() || dest.fail()) {
        return Failure;
    }

    return Success;
}

int main() {
    char file_from[100], file_to[100];

    cout << "Enter the source filename: ";
    cin.getline(file_from, 100);

    cout << "Enter the destination filename: ";
    cin.getline(file_to, 100);

    FileOpResult result = copy_file(file_from, file_to);

    if (result == Success) {
        cout << "File copied successfully!" << endl;
    }
    else {
        cout << "Error copying file!" << endl;
    }

    return 0;
}
```

Epic 5 Task 9  - Practice# programming:  Self Practice Task

```
#include <iostream>
```

```cpp
#include <fstream>
#include <string>

using namespace std;

int main() {
    int data = 123456;

    ofstream outFile("binary.dat", ios::binary);
    if (outFile.is_open()) {
        outFile.write(reinterpret_cast<char*>(&data), sizeof(data));
        outFile.close();
    }

    int readData;
    ifstream inFile("binary.dat", ios::binary);
    if (inFile.is_open()) {
        inFile.read(reinterpret_cast<char*>(&readData), sizeof(readData));
        cout << readData << endl;
        inFile.close();
    }

    ifstream inTextFile("example.txt");
    if (!inTextFile) {
        cout << "Error opening file!" << endl;
        return 1;
    }

    string line;
    while (getline(inTextFile, line)) {
        cout << line << endl;
    }

    inTextFile.close();
    return 0;
}
```

Робота у команді:

Ми зібрались одинраз та обговрили всі деталі епіку. Домовились що якщо в когось будуть питання то зберемось щераз.

**Висновок:** У результаті роботи вивчено основи роботи з файлами та рядковими змінними у C++. Розглянуто базові операції з файлами (відкриття, читання, запис, закриття), роботу з файловими дескрипторами та перевірку

стану файлів. Досліджено обробку текстових і бінарних файлів, включаючи форматування, парсинг і серіалізацію даних.

Проаналізовано методи роботи з рядковими змінними (char, string), їх об’єднання, порівняння та пошук. Вивчено використання потоків вводу/виводу (ifstream, ofstream, fstream) і основи створення власних бібліотек у C++. Розглянуто обробку помилок для забезпечення стабільності програм.

Ці знання створюють базу для вирішення складних програмних завдань.