

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:

ВНС Лабораторної Роботи №6

ВНС Лабораторної Роботи №8

ВНС Лабораторної Роботи №9

Алготестер Лабораторної Роботи №4

Алготестер Лабораторної Роботи №6

Практичних Робіт до блоку №5

Виконала:

Студентка групи ІІІ-13

Ходацька Аліна Віталіївна

Львів 2024

Тема роботи:

Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.

Мета роботи:

Ознайомитися з принципами роботи з текстовими та бінарними файлами в C++, використанням стандартної бібліотеки для маніпуляцій з файлами, а також створенням і застосуванням власних бібліотек для організації коду.

Теоретичні відомості:

- Файли
- Символи і рядкові змінні
- Текстові файли
- Бінарні файли
- Стандартна бібліотека та робота з файлами
- Створення й використання бібліотек

Використані джерела:

Виконання роботи

Завдання №1 VNS Lab 6 Task 1 Variant 12

Постановка завдання: Задано рядок, що складається із символів. Символи поєднуються в слова. Слова одне від одного відокремлюються одним або декількома пробілами. Наприкінці тексту ставиться крапка. Текст містить не більше 255 символів. Виконати ввід рядка, використовуючи функцію gets(s) і здійснити обробку рядка у відповідності зі своїм варіантом.

Завдання: Перетворити рядок таким чином, щоб букви кожного слова в ньому були відсортовані за зростанням.

```

#include <iostream>
#include <cstring> // для роботи зі стрічками
#include <algorithm> // для сортування

using namespace std;

void sortWord(char* word) {
    int length = strlen(word);
    sort(word, word + length); // Сортуємо символи в слові
}

int main() {
    char str[256]; // Вхідний рядок, максимальна довжина 255 символів + 1 для '\0'

    // Читання рядка
    cout << "Enter a string: (max length 255 characters)" << endl;
    fgets(str, sizeof(str), stdin);

    // Видаляємо зайвий символ нового рядка, якщо він є
    size_t str_len = strlen(str); // Обчислюємо довжину рядка один раз
    if (str_len > 0 && str[str_len - 1] == '\n') {
        str[str_len - 1] = '\0';
    }

    int i = 0;
    char word[256]; // Буфер для одного слова
    int wordIndex = 0;

    bool firstWord = true; // Для коректного виведення пробілів

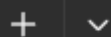
    while (i <= str_len) {
        if (str[i] != ' ' && str[i] != '\0') {
            // Збираємо символи слова
            word[wordIndex++] = str[i];
        }
        else {
            // Коли слово закінчено, сортуємо його і виводимо
            if (wordIndex > 0) {
                word[wordIndex] = '\0';
                sortWord(word);
                if (!firstWord) {
                    cout << " "; // Виводимо пробіл перед кожним словом після першого
                }
                cout << word;
                wordIndex = 0; // Очищаємо слово
                firstWord = false; // Після першого слова наступні будуть з пробілами
            }
        }
        i++;
    }
    cout << endl;
    return 0;
}

// використано fgets замість gets оскільки gets є застарілим і небезпечним(може викликати переповнення буфера)

```



Microsoft Visual Studio Debug Console



```

Enter a string: (max length 255 characters)
hello world
ehllo dlrow

```

Завдання №2 VNS Lab 8 Task 1 Variant 12

Постановка завдання: Сформувати двійковий файл із елементів, заданої у варіанті структури, роздрукувати його вміст, виконати знищення й додавання елементів у відповідності зі своїм варіантом, використовуючи для пошуку елементів що знищуються чи додаються, функцію. Формування, друк, додавання й знищення елементів оформити у вигляді функцій. Передбачити повідомлення про помилки при відкритті файлу й виконанні операцій вводу/виводу.

Завдання:

Структура "Музичний диск":

- назва;
- автор;
- тривалість;
- ціна.

Знищити перший елемент із заданою тривалістю, додати 2 елементи після елемента із заданим номером.


```
Microsoft Visual Studio Debug Console
File created successfully!
File contents:
Disk #1:
Title: Album 1
Author: Artist 1
Duration: 3600 seconds
Price: 15.99 USD
-----
Disk #2:
Title: Album 2
Author: Artist 2
Duration: 4200 seconds
Price: 20.99 USD
-----
Disk #3:
Title: Album 3
Author: Artist 3
Duration: 3900 seconds
Price: 18.49 USD
-----
Disk #4:
Title: Album 4
Author: Artist 4
Duration: 3000 seconds
Price: 12.99 USD
-----
Disk with duration 4200 seconds deleted.
Two disks added after disk #2
Updated file contents:
Disk #1:
Title: Album 1
Author: Artist 1
Duration: 3600 seconds
Price: 15.99 USD
-----
Disk #2:
Title: Album 3
Author: Artist 3
Duration: 3900 seconds
Price: 18.49 USD
-----
```

```
-----
Disk #3:
Title: Album 5
Author: Artist 5
Duration: 4500 seconds
Price: 22.99 USD
-----
Disk #4:
Title: Album 6
Author: Artist 6
Duration: 3600 seconds
Price: 19.99 USD
-----
Disk #5:
Title: Album 4
Author: Artist 4
Duration: 3000 seconds
Price: 12.99 USD
-----
Disk #6:
Title:
Author:
Duration: 0 seconds
Price: 0 USD
-----
```

Завдання №3 VNS Lab 9 Task 1 Variant 12

Постановка завдання: Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього інформацію.

Завдання:

- 1) Скопіювати з файлу F1 у файл F2 всі рядки, крім того рядка, у якій найбільше голосних букв.
- 2) Надрукувати номер цього рядка.


```

#include <iostream>
#include <fstream>
#include <string>
#include <cctype>

using namespace std;

// Функція для підрахунку голосних літер в рядку
int countVowels(const string& line) {
    int count = 0;
    for (char c : line) {
        c = tolower(c);
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
            count++;
        }
    }
    return count;
}

// Функція для створення текстового файлу F1 з кількома рядками
void createFileF1(const string& filename) {
    ofstream file(filename);
    if (!file) {
        cerr << "Error opening file for writing!" << endl;
        return;
    }

    file << "This is the first line." << endl;
    file << "Another line is here." << endl;
    file << "Some random text to count vowels." << endl;
    file << "The quick brown fox jumps over the lazy dog." << endl;
    file << "This line has more vowels than the others." << endl;
    file << "Short one." << endl;
    file << "Another example line with vowels." << endl;
    file << "How many vowels are in this one?" << endl;
    file << "Line with the highest vowels number." << endl;
    file << "Just another line with vowels." << endl;

    file.close();
}

// Функція для копіювання всіх рядків з файлу F1 у файл F2, за винятком рядка з найбільшими голосними
void copyExcludingMaxVowelsLine(const string& filenameF1, const string& filenameF2) {
    ifstream fileF1(filenameF1);
    ofstream fileF2(filenameF2);

    if (!fileF1 || !fileF2) {
        cerr << "Error opening files!" << endl;
        return;
    }

    string line;
    int maxVowels = -1;
    int lineNum = 0;
    int maxVowelLineNum = 0;
    string lines[100]; // Масив для збереження всіх рядків

    // Підрахунок голосних та збереження рядків
    while (getline(fileF1, line)) {
        int vowels = countVowels(line);
        lineNum++;

        if (vowels > maxVowels) {
            maxVowels = vowels;
            maxVowelLineNum = lineNum;
        }

        lines[lineNum - 1] = line; // Збереження рядка
    }

    // Копіювання всіх рядків, крім рядка з найбільшими голосними
    for (int i = 0; i < lineNum; i++) {
        if (i + 1 != maxVowelLineNum) { // Пропускаємо рядок з найбільшими голосними
            fileF2 << lines[i] << endl;
        }
    }

    cout << "The line with the most vowels is number: " << maxVowelLineNum << endl;

    fileF1.close();
    fileF2.close();
}

int main() {
    const string filenameF1 = "F1.txt";
    const string filenameF2 = "F2.txt";

    // Створення файлу F1
    createFileF1(filenameF1);

    // Копіювання з F1 в F2, пропускаючи рядок з найбільшими голосними
    copyExcludingMaxVowelsLine(filenameF1, filenameF2);

    return 0;
}

```



Microsoft Visual Studio Debug



The line with the most vowels is number: 5

Завдання №4 Algotester Lab 4 Task 1 Variant 3

Дано масив, який складається з N додатних цілих чисел.

Завдання - розділити його на три частини, по остачі від ділення на 3, по зростанню остачі (тобто спочатку йдуть числа, у яких остача 0, далі числа з остачею 1 і тоді нарешті числа з остачею 2).

Далі необхідно ті елементи, остача від ділення на 3 яких парна посортувати по зростанню, а ті, у яких остача 1 - по спаданню.

Після цього видаліть усі дублікати з масиву.

Виведіть результуючий масив.

Input

У першому рядку N - кількість чисел.

У другому рядку N чисел a_i - елементи масиву.

Output

У першому рядку M - кількість чисел у масиву

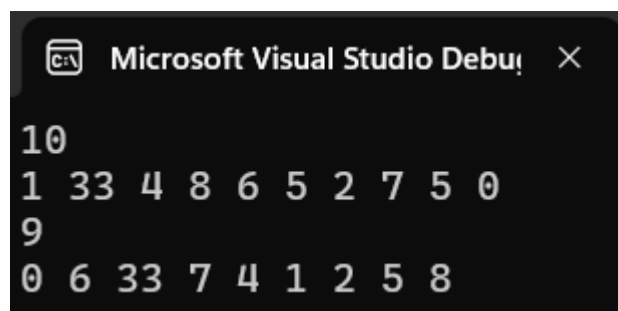
У другому рядку M посортованих за умовою чисел.

3 STL

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

bool compare0(int a, int b) {
    return a < b;
}
bool compare1(int a, int b) {
    return a > b;
}
int main() {
    int N;
    std::cin >> N;

    std::vector<int> arr(N);
    for (int i = 0; i < N; ++i) {
        std::cin >> arr[i];
    }
    // Розділяємо масив на три частини: за залишком 0, 1, 2
    std::vector<int> part0, part1, part2;
    for (int num : arr) {
        if (num % 3 == 0) part0.push_back(num);
        else if (num % 3 == 1) part1.push_back(num);
        else part2.push_back(num);
    }
    // Сортуюємо частини
    std::sort(part0.begin(), part0.end(), compare0); // За зростанням для залишку 0
    std::sort(part1.begin(), part1.end(), compare1); // За спаданням для залишку 1
    std::sort(part2.begin(), part2.end(), compare0); // За зростанням для залишку 2
    // Об'єднуємо масиви
    part0.insert(part0.end(), part1.begin(), part1.end());
    part0.insert(part0.end(), part2.begin(), part2.end());
    // Видаляємо дублікати
    part0.erase(std::unique(part0.begin(), part0.end()), part0.end());
    // Виводимо результат
    std::cout << part0.size() << "\n";
    for (int num : part0) {
        std::cout << num << " ";
    }
    std::cout << std::endl;
```



Microsoft Visual Studio Debug Console

```
10
1 33 4 8 6 5 2 7 5 0
9
0 6 33 7 4 1 2 5 8
```

Власна реалізація

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

void sort0(std::vector<int>& arr) {
    std::sort(arr.begin(), arr.end()); // За зростанням
}

void sort1(std::vector<int>& arr) {
    std::sort(arr.rbegin(), arr.rend()); // За спаданням
}

bool isUnique(const std::vector<int>& arr, int num) {
    return std::find(arr.begin(), arr.end(), num) ==
        arr.end();
}

int main() {
    int N;
    std::cin >> N;

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

    std::vector<int> part0, part1, part2;

    // Розділяємо масив за залишком при діленні на 3
    for (int num : arr) {
        if (num % 3 == 0) part0.push_back(num);
        else if (num % 3 == 1) part1.push_back(num);
        else part2.push_back(num);
    }

    // Сортуємо частини
    sort0(part0);
    sort1(part1);
    sort0(part2);

    // Об'єднуємо масиви
    std::vector<int> result;
    result.insert(result.end(), part0.begin(), part0.end());
    result.insert(result.end(), part1.begin(), part1.end());
    result.insert(result.end(), part2.begin(), part2.end());

    // Видаляємо дублікати
    std::vector<int> unique_result;
    for (int num : result) {
        if (isUnique(unique_result, num)) {
            unique_result.push_back(num);
        }
    }

    // Виводимо результат
    std::cout << unique_result.size() << "\n";
    for (int num : unique_result) {
        std::cout << num << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

Завдання №5 Algotester Lab 4 Task 2 Variant 2

Дано масив a з N цілих чисел.

Спочатку видаліть масиву a усі елементи що повторюються, наприклад масив $[1, 3, 3, 4]$ має перетворитися у $[1, 3, 4]$.

Після цього оберніть посортовану версію масиву a на K , тобто при $K=3$ масив $[1, 2, 3, 4, 5, 6, 7]$ перетвориться на $[4, 5, 6, 7, 1, 2, 3]$.

Виведіть результат.

Input

У першому рядку цілі числа N та K

У другому рядку N цілих чисел - елементи масиву a

Output

У першому рядку ціле число N - розмір множини a

У наступному рядку N цілих чисел - множина a

3 STL

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

using namespace std;

int main() {
    int N, K;
    cin >> N >> K;
    vector<int> a(N); // використовуємо вектор, бо масив не може змінювати розмір під час виконання

    for (int i = 0; i < N; ++i) { // цикл для введення елементів масиву
        cin >> a[i];
    }

    // Видалення дублікатів
    sort(a.begin(), a.end());
    auto last = unique(a.begin(), a.end());
    a.erase(last, a.end());

    // Обертання масиву на K
    int M = a.size(); // Розмір масиву після видалення дублікатів
    K = K % M; // У випадку, якщо K більше ніж розмір масиву
    rotate(a.begin(), a.begin() + K, a.end());

    // Виведення результату
    cout << M << endl;
    for (int num : a) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Своя реалізація

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

using namespace std;

// Функція для видалення дублікатів
void removeDuplicates(vector<int>& a) {
    sort(a.begin(), a.end());
    int uniqueCount = 0;
    for (int i = 1; i < a.size(); ++i) {
        if (a[i] != a[uniqueCount]) {
            ++uniqueCount;
            a[uniqueCount] = a[i];
        }
    }
    a.resize(uniqueCount + 1);
}

// Функція для обертання масиву
void rotateArray(vector<int>& a, int K) {
    int n = a.size();
    K = K % n;
    vector<int> temp(K);
    for (int i = 0; i < K; ++i) {
        temp[i] = a[i];
    }
    for (int i = K; i < n; ++i) {
        a[i - K] = a[i];
    }
    for (int i = 0; i < K; ++i) {
        a[n - K + i] = temp[i];
    }
}

int main() {
    int N, K;
    cin >> N >> K;
    vector<int> a(N);

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

    // Видалення дублікатів
    removeDuplicates(a);

    // Обертання масиву на K
    rotateArray(a, K);

    // Виведення результату
    cout << a.size() << endl;
    for (int num : a) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```



Microsoft Visual Studio Debug Console



```
10 11
5 6 2 3 1 2 3 3 4 7
7
5 6 7 1 2 3 4
```

Завдання №6 Algotester Lab 6 Variant 2

У вас є шахова дошка розміром 8×8 та дуже багато фігур.

Кожна клітинка може мати таке значення:

- Пуста клітинка O
- Пішак P
- Тура R
- Кінь N
- Слон B
- Король K
- Королева Q

Вам дають позиції фігур на дошці (всі фігури одного кольору, кількість королів може бути > 1).

Далі йдуть Q запитів з координатами клітинки $\{x, y\}$. На кожен запит ви маєте вивести стрічку s_i - посортовані за алфавітом букви фігур, які атакують цю клітинку (пішаки атакують вниз).

У випадку, якщо на клітинці стоїть якась фігура - виведіть символ X.

У випадку, якщо клітинку не атакують - виведіть O.

Наявність фігури у певній клітинці не блокує атаку для іншої фігури. Тобто якщо між турою та клітинкою стоїть інша фігура - вважається що тура атакує цю клітинку.

Input

У перших 8 рядках стрічка row_i - стан i -го рядка дошки.

У наступному рядку ціле число Q - кількість записів

У наступних Q рядках 2 цілих числа x та y - координати клітинки

Output

Q разів відповідь у наступному форматі:

Строка result - усі фігури, які атакують клітинку з запиту.


```

#include <iostream>
#include <vector>
#include <set>
#include <cmath>
#include <algorithm>

using namespace std;

struct Figure {
    char type;
    int x, y;
};

bool canAttack(const Figure& f, int targetX, int targetY) {
    switch (f.type) {
        case 'P': return (f.x == targetX - 1 && abs(f.y - targetY) == 1);
        case 'R': return (f.x == targetX || f.y == targetY);
        case 'N': return (abs(f.x - targetX) == 2 && abs(f.y - targetY) == 1) || (abs(f.x - targetX) == 1
&& abs(f.y - targetY) == 2);
        case 'B': return (abs(f.x - targetX) == abs(f.y - targetY));
        case 'K': return (abs(f.x - targetX) <= 1 && abs(f.y - targetY) <= 1);
        case 'Q': return (f.x == targetX || f.y == targetY || abs(f.x - targetX) == abs(f.y - targetY));
        default: return false;
    }
}

vector<string> processQueries(const vector<Figure>& figures, const vector<pair<int, int>>& queries) {
    vector<string> results;

    for (const auto& query : queries) {
        int x = query.first;
        int y = query.second;
        set<char> attackers;
        bool isOccupied = false;

        for (const auto& figure : figures) {
            if (figure.x == x && figure.y == y) {
                isOccupied = true;
                break;
            }
            if (canAttack(figure, x, y)) {
                attackers.insert(figure.type);
            }
        }

        if (isOccupied) {
            results.push_back("X");
        } else if (attackers.empty()) {
            results.push_back("0");
        } else {
            string attackersList(attackers.begin(), attackers.end());
            results.push_back(attackersList);
        }
    }

    return results;
}

int main() {
    vector<Figure> figures;
    for (int i = 0; i < 8; ++i) {
        string row;
        cin >> row;
        for (int j = 0; j < row.size(); ++j) {
            if (row[j] != '0') {
                figures.push_back({row[j], i + 1, j + 1});
            }
        }
    }

    int q;
    cin >> q;
    vector<pair<int, int>> queries(q);
    for (int i = 0; i < q; ++i) {
        cin >> queries[i].first >> queries[i].second;
    }

    vector<string> results = processQueries(figures, queries);
    for (const auto& result : results) {
        cout << result << endl;
    }

    return 0;
}

```



Microsoft Visual Studio Debug Console



```
00000000
0R000000
00N00000
0000P000
00000000
00000000
K0Q00000
0000000R
```

```
7
8 1
1 2
5 4
5 1
6 2
8 4
6 7
KR
NR
NP
Q
KQR
QR
O
```

Завдання №7 Practice Task

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.

Задача №2 – Копіювання вмісту файла у інший файл

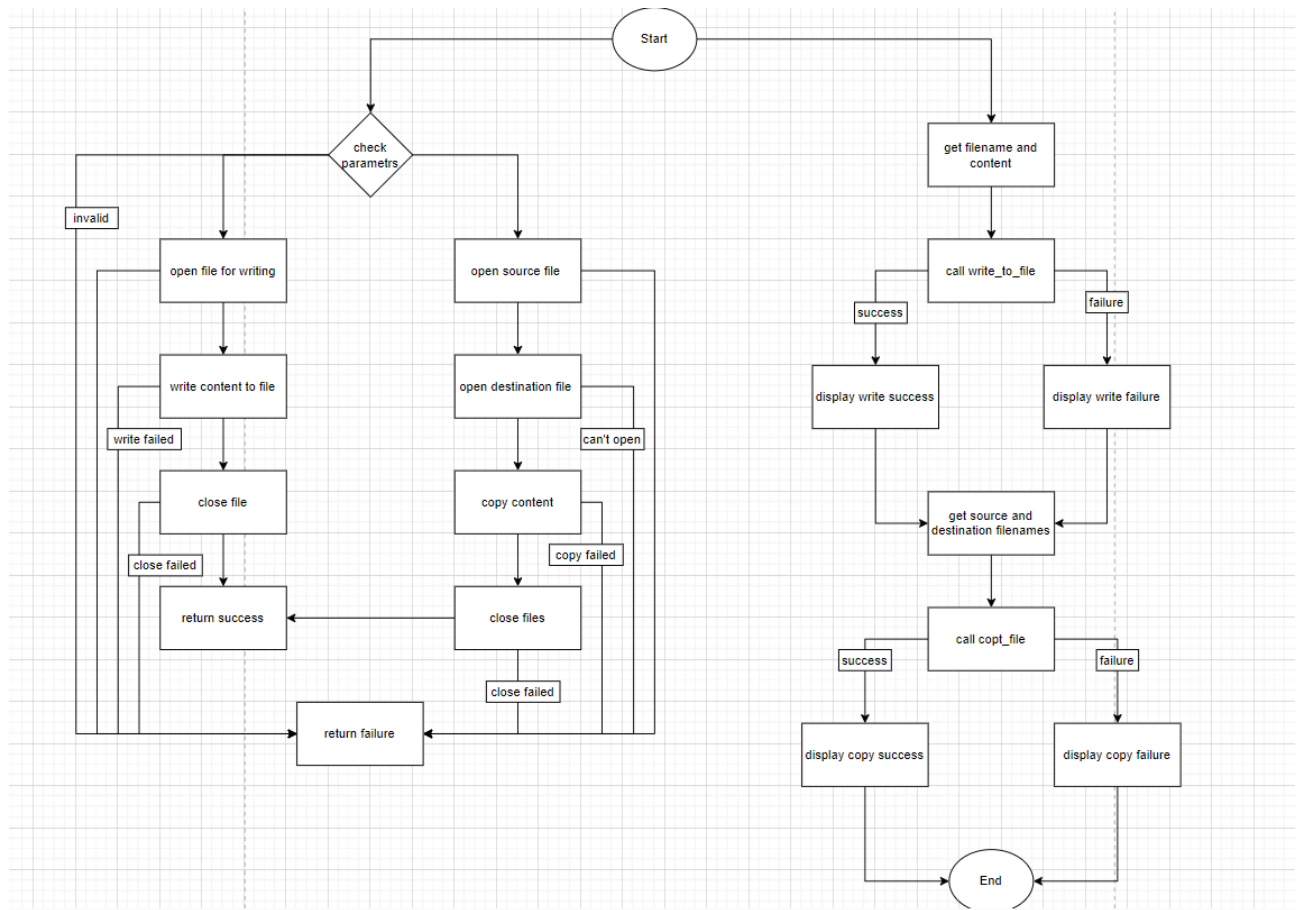
Реалізувати функцію створення файла і запису в нього даних:

```
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.



```

#include <iostream>
#include <fstream>
#include <string>

// Оголошення перерахунку для результату операції
enum FileOpResult { Success, Failure };

// Функція для запису в файл
FileOpResult write_to_file(const char* name, const char* content) {
    // Перевірка віхідних параметрів
    if (name == nullptr || content == nullptr) {
        return Failure;
    }

    // Спроба відкрити файл для запису
    std::ofstream outfile(name);
    if (!outfile.is_open()) {
        return Failure;
    }

    // Запис вмісту в файл
    outfile << content;

    // Перевірка чи запис був успішним
    if (outfile.fail()) {
        return Failure;
    }

    // Закриття файлу
    outfile.close();

    // Перевірка чи файл був успішно закритий
    if (outfile.fail()) {
        return Failure;
    }

    return Success;
}

// Функція для копіювання вмісту з одного файлу в інший
FileOpResult copy_file(const char* file_from, const char* file_to) {
    // Перевірка віхідних параметрів
    if (file_from == nullptr || file_to == nullptr) {
        return Failure;
    }

    // Спроба відкрити файл для читання
    std::ifstream infile(file_from, std::ios::binary);
    if (!infile.is_open()) {
        return Failure;
    }

    // Спроба відкрити файл для запису
    std::ofstream outfile(file_to, std::ios::binary);
    if (!outfile.is_open()) {
        infile.close();
        return Failure;
    }

    // Копіювання вмісту
    outfile << infile.rdbuf();

    // Перевірка чи запис був успішним
    if (outfile.fail() || infile.fail()) {
        infile.close();
        outfile.close();
        return Failure;
    }

    // Закриття файлів
    infile.close();
    outfile.close();

    // Перевірка чи файли були успішно закриті
    if (infile.fail() || outfile.fail()) {
        return Failure;
    }

    return Success;
}

int main() {
    // Запис у файл
    std::string filename;
    std::string content;

    std::cout << "Enter filename for write operation: ";
    std::getline(std::cin, filename);

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

    FileOpResult result = write_to_file(filename.c_str(),
    content.c_str());
    if (result == Success) {
        std::cout << "File written successfully." << std::endl;
    }
    else {
        std::cout << "Failed to write to file." << std::endl;
    }

    // Копіювання файлів
    std::string sourceFile;
    std::string destinationFile;

    std::cout << "\nEnter source filename for copy operation: ";
    std::getline(std::cin, sourceFile);

    std::cout << "Enter destination filename for copy operation: ";
    std::getline(std::cin, destinationFile);

    result = copy_file(sourceFile.c_str(), destinationFile.c_str());
    if (result == Success) {
        std::cout << "File copied successfully." << std::endl;
    }
    else {
        std::cout << "Failed to copy file." << std::endl;
    }

    return 0;
}

```



Microsoft Visual Studio Debu! X



```
Enter filename for write operation: hello world  
Enter content to write to the file: Hello World!  
File written successfully.
```

```
Enter source filename for copy operation: hello world  
Enter destination filename for copy operation: hello  
File copied successfully.
```

Завдання №8 Algotester Self Practice work Lab 4 Variant 1

Дано 2 цілих чисел масиви, розміром N та M.

Завдання вивести:

1. Різницю N-M
2. Різницю M-N
3. Їх перетин
4. Їх об'єднання
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 четвертому 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

```

#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

void printSet(const vector<int>& result) {
    cout << result.size() << endl;
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;
}

int main() {
    int N, M;

    // Вхідні дані
    cin >> N;
    vector<int> set1(N);
    for (int i = 0; i < N; ++i) {
        cin >> set1[i];
    }

    cin >> M;
    vector<int> set2(M);
    for (int i = 0; i < M; ++i) {
        cin >> set2[i];
    }

    // Сортуюмо масиви
    sort(set1.begin(), set1.end());
    sort(set2.begin(), set2.end());

    // Різниця N-M
    vector<int> diff1;
    set_difference(set1.begin(), set1.end(), set2.begin(), set2.end(), back_inserter(diff1));
    printSet(diff1);

    // Різниця M-N
    vector<int> diff2;
    set_difference(set2.begin(), set2.end(), set1.begin(), set1.end(), back_inserter(diff2));
    printSet(diff2);

    // Перетин
    vector<int> inter;
    set_intersection(set1.begin(), set1.end(), set2.begin(), set2.end(), back_inserter(inter));
    printSet(inter);

    // Об'єднання
    vector<int> uni;
    set_union(set1.begin(), set1.end(), set2.begin(), set2.end(), back_inserter(uni));
    printSet(uni);

    // Симетрична різниця
    vector<int> symDiff;
    set_symmetric_difference(set1.begin(), set1.end(), set2.begin(), set2.end(), back_inserter(symDiff));
    printSet(symDiff);

    return 0;
}

```

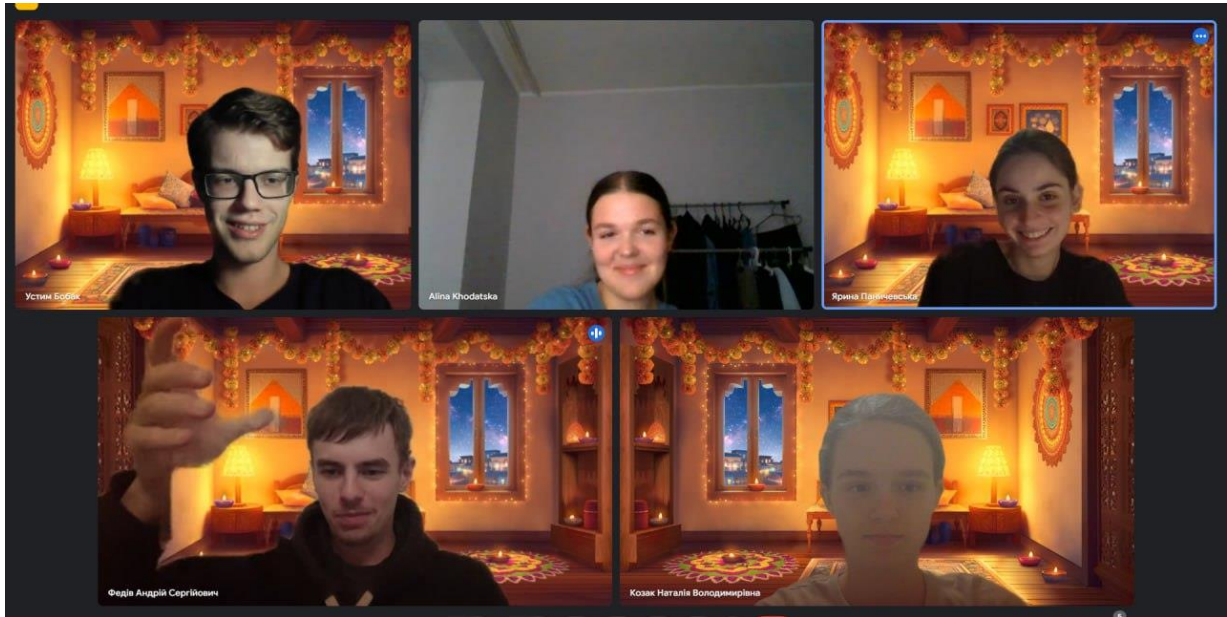
Microsoft Visual Studio Debug Console

```

5
1 2 3 4 5
5
4 5 6 7 8
3
1 2 3
3
6 7 8
2
4 5
8
1 2 3 4 5 6 7 8
6
1 2 3 6 7 8

```


Зустріч з командою та дошка в Trello



Висновок: в результаті виконання цього епіку я ознайомилась з принципами роботи з текстовими та бінарними файлами в C++, використанням стандартної бібліотеки для маніпуляцій з файлами, а також створенням і застосуванням власних бібліотек для організації коду.

Pull request: https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/643