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

Виконав:
Студент групи ШІ-11
Станько Олег Ігорович

Львів 2024

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

Мета: навчитися працювати з файлами, створення бібліотек.

Теоретичні відомості

1. Вступ до Роботи з Файлами
2. Символи і Рядкові Змінні
3. Текстові Файли
4. Бінарні Файли
5. Стандартна бібліотека та робота з файлами
6. Створення й використання бібліотек

ідуальний план опрацювання теорії

1. Вступ до Роботи з Файлами
   https://www.youtube.com/watch?v=o7XT7cTChXE&ab_channel=%D0%A8%D0%BA%D0%BE%D0%BB%D0%B0%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
   Навчився відкривати, читати, записувати, закривати файл;

   Витрачено 1 годин
2. Символи і Рядкові Змінні
   Рядки C-style в C++ / Уроки по C++ / aCode
   навчився працювати з char та string.
   Витрачено 40 хвилин
3. Текстові Файли
   Базовий курс програмування на C++. Робота з текстовими файлами в C++. - Українське програмування
   обробляти рядки з файлу, помилки зчитування.
   Витрачено 20 хвилин
4. Бінарні Файли
   std::basic_fstream - cppreference.com
   зчитання, запис бінарних даних
   Витрачено 30 хвилин
5. Стандартна бібліотека та робота з файлами
   std::basic_ifstream - cppreference.com
   std::basic_ofstream - cppreference.com
   std::basic_fstream - cppreference.com

   Витрачено 30 хвилин
6. Створення й використання бібліотек
   створювати власні бібліотек у C++
   C++: створення й використання бібліотек
   Витрачено 30 хвилин

Виконання роботи:

1. Опрацювання завдання та вимог до програм.

VNS Lab 6

18. Всі слова рядка, які починаються із цифри відсортувати за спаданням.

VNS Lab 8

18. Структура "Книга":

- назва;

- автор;

- рік видання;

- кількість сторінок.

Знищити 3 елементи з початку файлу, додати елемент перед елементом із зазначеною назвою

VNS Lab 9

18. 1) Скопіювати з файлу F1 у файл F2 всі рядки, у яких немає однакових слів. 2) Визначити кількість голосних букв у першому рядку файлу F2

Algotester Lab 4

Вам дано масив a з N цілих чисел.
Спочатку видаліть масиву aa усі елементи що повторюються, наприклад масив [1, 3, 3, 4] має перетворитися у [1, 3, 4].
Після цього оберніть посортовану версію масиву aa 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.

Вхідні дані

У першому рядку цілі числа N та K

У другому рядку N цілих чисел - елементи масиву a

Вихідні дані

У першому рядку ціле число N - розмір множини a
У наступному рядку N цілих чисел - множина a

написати 2 варіанти розвязку, один з використанням засобів STL (std::unique, std::sort, std::rotate), інший зі своєю реалізацією.

Algotester Lab 6

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 йдутьQ 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.

Наявніть 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.

Вхідні дані

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

У наступному рядку ціле число Q - кількість записів

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

Вихідні дані

Q разів відповідь у наступному форматі:

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

Class Practice Work

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.

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.

Self-practice task

Зеник та Марічка грають у поле чудес. Спочатку Зеник пише на дошці загадане слово й закриває всі його букви. За один хід Марічка називає букву, а Зеник відкриває всі такі букви у слові.

Вам необхідно визначити, за яку мінімальну кількість ходів Марічка зможе відкрити всі букви у слові.

Наприклад, якщо Зеник загадав слово MAMA, то Марічка зможе його відкрити за два ходи, назвавши букви M та A.

Вхідні дані

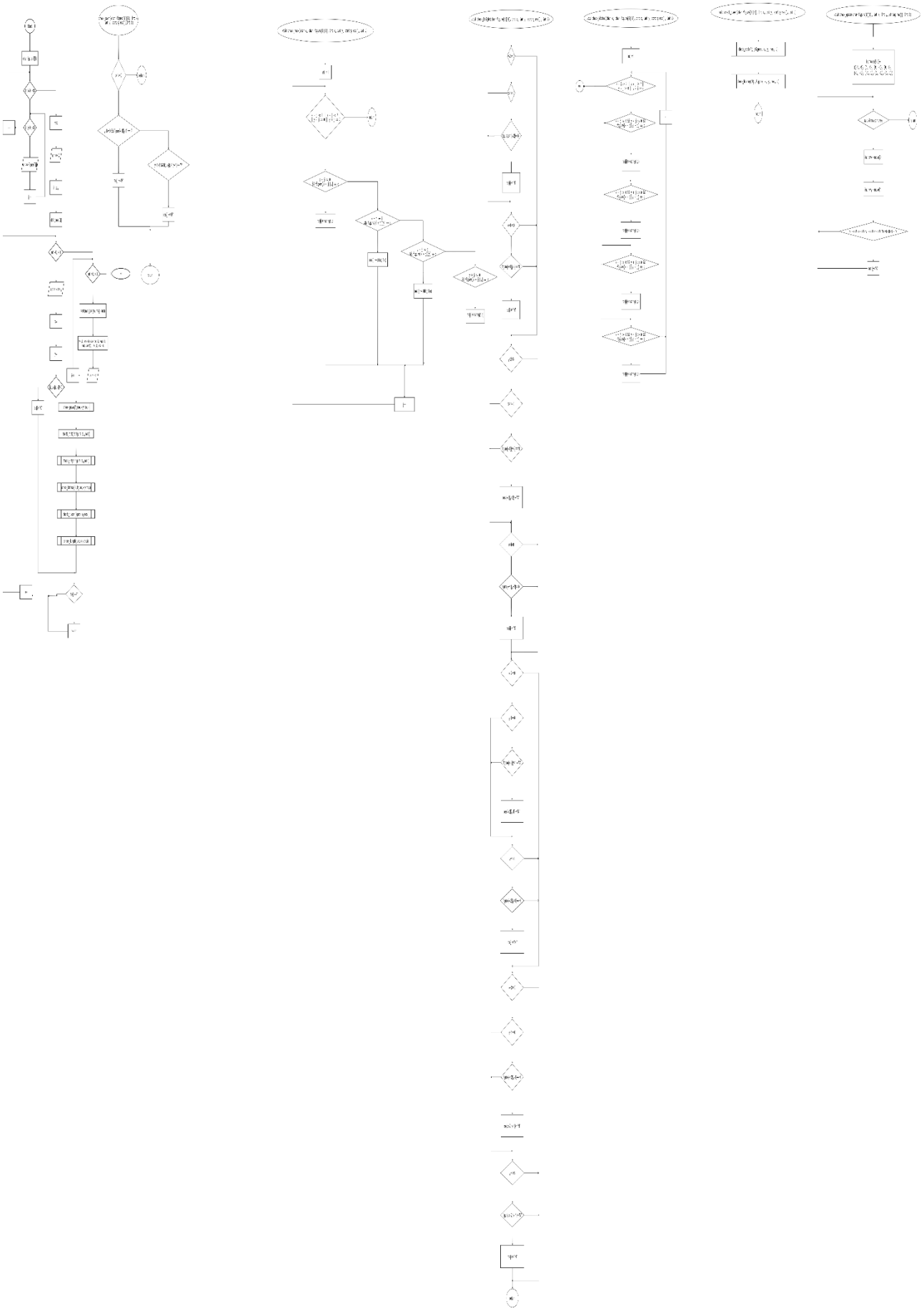
Вхідні дані містять рядок s — загадане Зеником слово.

Вихідні дані

В одному рядку виведіть ціле число — мінімальну кількість ходів.

2. Блок-схеми

ALGO LAB 6



Коди:

VNS LAB 6

```cpp
#include <iostream>
#include <string>
#include <sstream>
#include <cctype>

using namespace std;

bool starts_with_digit(const string& word)
{
    return !word.empty() && isdigit(word[0]);
}

void sort_words_starting_with_digit(string& input) {
    stringstream ss(input);
    string words[100];
    int count = 0;

    string word;
    while (ss >> word) {
        if (starts_with_digit(word)) {
            words[count++] = word;
        }
    }

    for (int i = 0; i < count - 1; ++i)
    {
        for (int j = i + 1; j < count; ++j)
        {
            if (words[i] < words[j])
            {
                swap(words[i], words[j]);
            }
        }
    }

    cout << "Words starting with a digit (sorted in descending order):" << endl;
    for (int i = 0; i < count; ++i)
    {
        cout << words[i] << " ";
    }
    cout << endl;
}

int main()
{
    string input;
    cout << "Enter a string: ";
    getline(cin, input);
    sort_words_starting_with_digit(input);
    return 0;
}
```

VNS lab 8

```cpp
#include <iostream>
#include <fstream>
#include <iomanip>
#include <string>
#include <random>
#include <vector>

using namespace std;

struct book {
    string name;
    string author;
    int year_of_publication;
    int pages;
};

bool is_file_empty(const string& filename);
void deletion(const string& filename);
void add_some_books(const string& filename);
void add_book_before_given(const string& filename, const string& given, const string& book_name, const string& new_author, int year_
void print(const string& filename);

int main() {
    const string filename = "books.txt";
    int choice;

    do {
        cout << "\n1. Delete 3 books from start\n"
             << "2. Add book before a given book\n"
             << "3. Exit\n"
             << "Enter your choice: ";
        cin >> choice;

        if (is_file_empty(filename)) {
            cout << "The file is empty. Adding default books...\n";
            add_some_books(filename);
        }

        switch (choice) {
            case 1:
                deletion(filename);
                break;
            case 2: {
                string given_book, book_name, new_author;
                int year_of_publication, pages;

                cout << "Enter the name of the book before which to add: ";
                cin >> given_book;
                cout << "Enter new book name: ";
                cin >> book_name;
                cout << "Enter new author: ";
                cin >> new_author;
                cout << "Enter year of publication: ";
                cin >> year_of_publication;
                cout << "Enter number of pages: ";
                cin >> pages;

                add_book_before_given(filename, given_book, book_name, new_author, year_of_publication, pages);
                break;
            }
            case 3:
                cout << "Exiting...\n";
                break;
            default:
                cerr << "Invalid choice!\n";
        }
        print(filename);
    } while (choice != 3);

    return 0;
}

void deletion(const string& filename) {
    ifstream inFile(filename);
    if (!inFile.is_open()) {
        cerr << "Error opening file for reading!\n";
        return;
    }

    vector<string> lines;
    string line;
    int current_line = 0;
    while (getline(inFile, line)) {
        if (current_line >= 3) {
            lines.push_back(line);
        }
        current_line++;
    }
    inFile.close();

    ofstream outFile(filename, ios::trunc);
    if (!outFile.is_open()) {
        cerr << "Error opening file for writing!" << endl;
        return;
    }

    for (const auto& line : lines) {
        outFile << line << endl;
    }
    outFile.close();
}
```

```cpp

void add_some_books(const string& filename) {
    ofstream outFile(filename, ios::app);
    if (!outFile.is_open()) {
        cerr << "Error opening file for writing!\n";
        return;
    }

    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<> dis(1, 500);

    book books[10];
    for (int i = 0; i < 10; ++i) {
        books[i].name = "name" + to_string(i + 1);
        books[i].author = "author" + to_string(i + 1);
        books[i].pages = dis(gen);
        books[i].year_of_publication = 1500 + dis(gen);

        outFile << books[i].name << " " << books[i].author << " "
                << books[i].year_of_publication << " " << books[i].pages << endl;
    }
    outFile.close();
}

void add_book_before_given(const string& filename, const string& given, const string& book_name, const string& new_author, int year_of_publication, int pages) {
    ifstream inFile(filename);
    if (!inFile.is_open()) {
        cerr << "Error opening file for reading!\n";
        return;
    }

    vector<book> books;
    book temp;
    book new_book = {book_name, new_author, year_of_publication, pages};
    bool inserted = false;

    while (inFile >> temp.name >> temp.author >> temp.year_of_publication >> temp.pages) {
        if (!inserted && temp.name == given) {
            books.push_back(new_book);
            inserted = true;
        }
        books.push_back(temp);
    }
    inFile.close();

    ofstream outFile(filename, ios::trunc);
    if (!outFile.is_open()) {
        cerr << "Error opening file for writing!" << endl;
        return;
    }

    for (const auto& b : books) {
        outFile << b.name << " " << b.author << " " << b.year_of_publication
                << " " << b.pages << endl;
    }
    outFile.close();
}

void print(const string& filename) {
    ifstream inFile(filename);
    if (inFile.is_open()) {
        string line;
        while (getline(inFile, line)) {
            cout << line << endl;
        }
        inFile.close();
    } else {
        cerr << "Error opening file for reading!\n";
    }
}

bool is_file_empty(const string& filename) {
    ifstream inFile(filename, ios::ate | ios::binary);
    if (!inFile.is_open()) {
        return true;
    }
    return inFile.tellg() == 0;
}

```

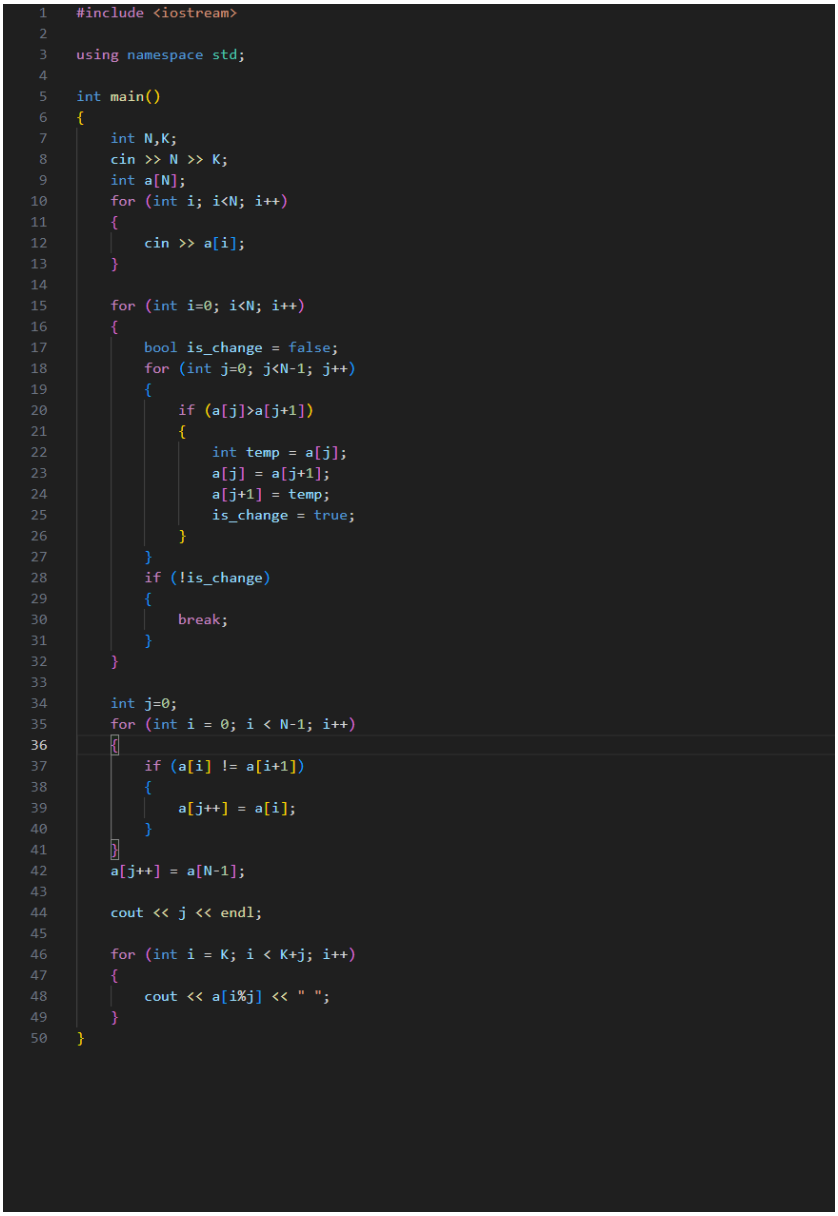
VNS LAB 9

```cpp
#include <iostream>
#include <sstream>
#include <cstring>
#include <unordered_set>
#include <algorithm>

using namespace std;

bool has_duplicate_words(const string& line) {
    istringstream iss(line);
    unordered_set<string> word_set;
    string word;

    while (iss >> word) {
        if (word_set.find(word) != word_set.end()) {
            return true;
        }
        word_set.insert(word);
    }
    return false;
}

void copy_unique_lines(const char* file_from, const char* file_to) {
    ifstream inFile(file_from);
    if (!inFile.is_open()) {
        cerr << "Error opening source file for reading!\n";
        return;
    }

    ofstream outFile(file_to);
    if (!outFile.is_open()) {
        cerr << "Error opening destination file for writing!\n";
        return;
    }

    string line;
    while (getline(inFile, line)) {
        if (!has_duplicate_words(line)) {
            outFile << line << endl;
        }
    }

    inFile.close();
    outFile.close();
}

int count_vowels(const string& line) {
    int count = 0;
    for (char c : line) {
        c = tolower(c);
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' || c == 'y') {
            count++;
        }
    }
    return count;
}

void add_some_lines(const char* file1) {
    ofstream outFile(file1, ios::app);
    if (!outFile.is_open()) {
        cerr << "Error opening file for writing!\n";
        return;
    }

    string line;
    int n;
    cout << "Enter the number of lines you want to add: ";
    cin >> n;
    cin.ignore();
    cout << "Enter the lines you want to add:\n";
    for (int i = 0; i < n; ++i) {
        getline(cin, line);
        outFile << line << endl;
    }
    outFile.close();
}

void process_file(const char* file1, const char* file2) {
    add_some_lines(file1);
    copy_unique_lines(file1, file2);

    cout << "Content of " << file2 << ":\n";
    ifstream debugFile(file2);
    string debugLine;
    while (getline(debugFile, debugLine)) {
        cout << debugLine << endl;
    }
    debugFile.close();

    ifstream inFile(file2);
    if (!inFile.is_open()) {
        cerr << "Error opening destination file for reading!\n";
        return;
    }

    string line;
    int lineNumber = 1;
    while (getline(inFile, line)) {
        int vowel_count = count_vowels(line);
        cout << "Number of vowels in line " << lineNumber << " of " << file2 << ": " << vowel_count << endl;
        lineNumber++;
    }

    inFile.close();
}

int main() {
    const char* file1 = "F1.txt";
    const char* file2 = "F2.txt";

    process_file(file1, file2);

    return 0;
}
```

Algo lab 4.1

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

using namespace std;

int main()
{
    int N,K;
    cin >> N >> K;
    int a[N];
    for (int i; i<N; i++)
    {
        cin >> a[i];
    }

    sort(a, a + N);
    auto last = unique(a, a + N);
    int size = last - a;
    std::rotate(a, a + K%size, a + size);
    cout << size << endl;
    for (int i = 0; i < size; i++)
    {
        cout << a[i] << " ";
    }
}
```

Algo lab 4.2



```cpp
#include <iostream>

using namespace std;

int main()
{
    int N,K;
    cin >> N >> K;
    int a[N];
    for (int i; i<N; i++)
    {
        cin >> a[i];
    }

    for (int i=0; i<N; i++)
    {
        bool is_change = false;
        for (int j=0; j<N-1; j++)
        {
            if (a[j]>a[j+1])
            {
                int temp = a[j];
                a[j] = a[j+1];
                a[j+1] = temp;
                is_change = true;
            }
        }
        if (!is_change)
        {
            break;
        }
    }

    int j=0;
    for (int i = 0; i < N-1; i++)
    {
        if (a[i] != a[i+1])
        {
            a[j++] = a[i];
        }
    }
    a[j++] = a[N-1];

    cout << j << endl;

    for (int i = K; i < K+j; i++)
    {
        cout << a[i%j] << " ";
    }
}
```

ALGO LAB 6

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

using namespace std;

void check_pawn(char figure[8][8], int x, int y, string res[],int i)
{
    if (x - 1 >= 0)
    {
        if (y - 1 >= 0 && figure[x - 1][y - 1] == 'P')
        {
            res[i] += "P";
        }
        if (y + 1 < 8 && figure[x - 1][y + 1] == 'P')
        {
            res[i] += "P";
        }
    }
}

void check_rook(char c, char figure[8][8], int x, int y, string res[], int i)
{
    int j = 1;
    while (x - j >= 0 || x + j <= 7 || y - j >= 0 || y + j <= 7)
    {
        if (x - j >= 0 && figure[x - j][y] == c)
        {
            res[i] += string(1, c);
        }
        if (x + j <= 7 && figure[x + j][y] == c)
        {
            res[i] += string(1, c);
        }
        if (y - j >= 0 && figure[x][y - j] == c)
        {
            res[i] += string(1, c);
        }
        if (y + j <= 7 && figure[x][y + j] == c)
        {
            res[i] += string(1, c);
        }
        j++;
    }
}

void check_knight(char figure[8][8], int x, int y, string res[], int i)
{
    if (y-2>=0)
    {
        if (x-1>=0)
        {
            if (figure[x-1][y-2] == 'N')
            {
                res[i] += "N";
            }
        }
        if (x+1<8)
        {
            if (figure[x+1][y-2] == 'N')
            {
                res[i] += "N";
            }
        }
    }
    if (y+2<8)
    {
        if (x-1>=0)
        {
            if (figure[x-1][y+2] == 'N')
            {
                res[i] += "N";
            }
        }
        if (x+1<8)
        {
            if (figure[x+1][y+2] == 'N')
            {
                res[i] += "N";
            }
        }
    }
    if (x-2>=0)
    {
        if (y-1>=0)
        {
            if (figure[x-2][y-1] == 'N')
            {
                res[i] += "N";
            }
        }
        if (y+1<8)
        {
            if (figure[x-2][y+1] == 'N')
            {
                res[i] += "N";
            }
        }
    }
    if (x+1<8)
    {
        if (y-1>=0)
        {
            if (figure[x+2][y-1] == 'N')
            {
                res[i] += "N";
            }
        }
        if (y+1<8)
        {
            if (figure[x+2][y+1] == 'N')
            {
                res[i] += "N";
            }
        }
    }
}
```

```cpp

void check_bishop(char c, char figure[8][8], int x, int y, string res[], int i)
{
    int j = 1;
    while (x - j >= 0 || x + j <= 7 || y - j >= 0 || y + j <= 7)
    {
        if (x - j >= 0 && y - j >= 0 && figure[x - j][y - j] == c)
        {
            res[i] += string(1, c);
        }
        if (x + j <= 7 && y + j <= 7 && figure[x + j][y + j] == c)
        {
            res[i] += string(1, c);
        }
        if (x - j >= 0 && y + j <= 7 && figure[x - j][y + j] == c)
        {
            res[i] += string(1, c);
        }
        if (x + j <= 7 && y - j >= 0 && figure[x + j][y - j] == c) {
            res[i] += string(1, c);
            }
        j++;
    }
}
void check_queen(char figure[8][8], int x, int y, string res[], int i)
{
    check_rook('Q',figure, x, y, res, i);
    check_bishop('Q',figure, x, y, res, i);
}
void check_king(char figure[8][8], int x, int y, string res[], int i)
{
    int moves[8][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}, {-1, -1}, {-1, 1}, {1, -1}, {1, 1}};
    for (auto &move : moves) {
        int nx = x + move[0];
        int ny = y + move[1];
        if (nx >= 0 && nx < 8 && ny >= 0 && ny < 8 && figure[nx][ny] == 'K')
        {
            res[i] += "K";
        }
    }
}

int main()
{
    char figure[8][8];
    for (int i=0; i<8; i++)
    {
        for (int j=0; j<8; j++)
        {
            cin >> figure[i][j];
        }
    }
    int Q;
    cin >> Q;
    int x, y;
    string res[Q]={};
    for (int i=0; i<Q; i++)
    {
        cin >> x >> y;
        x--;
        y--;
        if (figure[x][y]!='0')
        {
            res[i]="X";
        }
        else
        {
            check_pawn(figure,x,y,res,i);
            check_rook('R',figure,x,y,res,i);
            check_knight(figure,x,y,res,i);
            check_bishop('B',figure,x,y,res,i);
            check_queen(figure,x,y,res,i);
            check_king(figure,x,y,res,i);
            if (res[i] == "")
            {
                res[i]="0";
            }
        }
    }

    for (int i=0; i<Q; i++)
    {
        sort(res[i].begin(), res[i].end());
        res[i].erase(unique(res[i].begin(), res[i].end()), res[i].end());
        cout << res[i] << endl;
    }
}
```

Practice

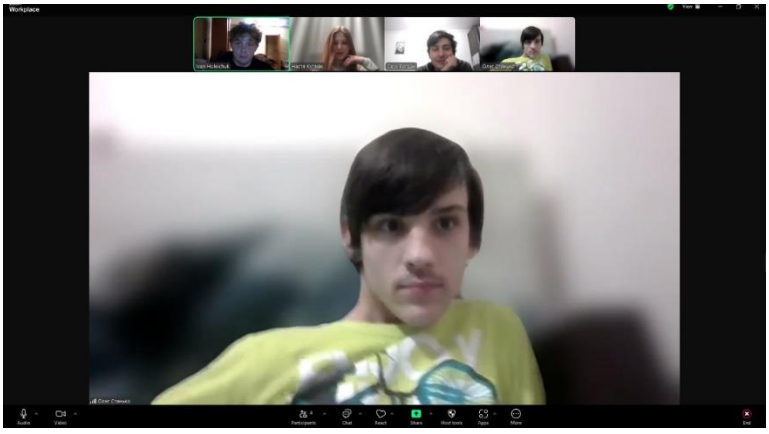
nauka > epic_5 > practise.cpp > main()

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

using namespace std;


enum FileOpResult
{
    Success,
    Failure
};


FileOpResult write_to_file(const char *name, const char *content);
FileOpResult copy_file(const char *file_from, const char *file_to);

int main()
{
    const char* filename1 = "file1.txt";
    const char* filename2 = "file2.txt";
    char content[256];

    cout << "enter a content(line)\n ";
    cin.getline(content, sizeof(content));

    FileOpResult result1 = write_to_file(filename1, content);
    if (result1 == Success) {
        cout << "Success. File created." << endl;
    } else {
        cout << "Failure. File could not be created." << endl;
    }

    FileOpResult result2 = copy_file(filename1, filename2);
    if (result2 == Success) {
        cout << "Success. File is copied." << endl;
    } else {
        cout << "Failure coping file." << endl;
    }
}

FileOpResult write_to_file(const char *name, const char *content)
{
    if (name == nullptr || strlen(name) == 0) {
        return FileOpResult::Failure;
    }

    FILE* f = fopen(name, "w");
    if (f == NULL) {
        return FileOpResult::Failure;
    }

    int len = strlen(content);
    size_t written = fwrite(content, sizeof(char), len, f);

    if (written != len) {
        fclose(f);
        return FileOpResult::Failure;
    }

    if (fclose(f) != 0) {
        return FileOpResult::Failure;
    }

    return FileOpResult::Success;
}

FileOpResult copy_file(const char *file_from, const char *file_to)
{
    if (file_from == nullptr || strlen(file_from) == 0 || file_to == nullptr || strlen(file_to) == 0)
    {
        return FileOpResult::Failure;
    }

    FILE* f1 = fopen(file_from, "r");
    FILE* f2 = fopen(file_to, "w");

    if (f1 == NULL || f2 == NULL)
    {
        return FileOpResult::Failure;
    }

    char buffer[512];
    while(fgets(buffer, sizeof(buffer), f1) != NULL)
    {
        if(fputs(buffer, f2) == EOF){
            fclose(f1);
            fclose(f2);
            return FileOpResult::Failure;
        }
    }

    if (fclose(f1) != 0 || fclose(f2) != 0)
    {
        return FileOpResult::Failure;
    }

    return FileOpResult::Success;
}
```

Self practice work

```cpp
#include <iostream>
#include <string>
using namespace std;

int main()
{
    string s;
    getline(cin, s);

    bool was_changed;
    for (int i = 0; i < s.length(); i++)
    {
        was_changed = false;
        for (int j = 0; j < s.length() - 1; j++)
        {
            if (s[j] > s[j + 1]) {
                char temp = s[j];
                s[j] = s[j + 1];
                s[j + 1] = temp;
                was_changed = true;
            }
        }
        if (!was_changed)
        {
            break;
        }
    }

    int rez = 0;
    for (int i = 0; i < s.length() - 1; i++)
    {
        if (s[i] != s[i + 1])
        {
            rez++;
        }
    }
    rez++;
    cout << rez;

    return 0;
}
```

Робота з командою



Зустрічалися в зумі 19 листопада

Висновок: Виконуючи 5 епік, я ознайомився з основними принципами роботи з файлами у C++. Вивчив текстові та бінарні файли, зокрема операції відкриття, читання, запису та закриття. Опанував перевірку стану файлу для обробки помилок і забезпечення надійності. Окрім цього, ознайомився зі стандартними бібліотеками для роботи з файлами (ifstream, ofstream, fstream).