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-12
Гаврих Юрій Дмитрович

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:

Навчитись працювати з файловою системою в C++ , рядками типу std::string та char. Використання стандартної бібліотеки та створення власних бібліотек.

Теоретичні відомості:

Базовий файловий ввід і вивід

Символьний тип даних char

Виконання роботи

Task 3 - Lab# programming: VNS Lab 6

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.
15. Визначити яке слово зустрічається в рядку найчастіше.

***Час виконання* ~ 1,5 год.**

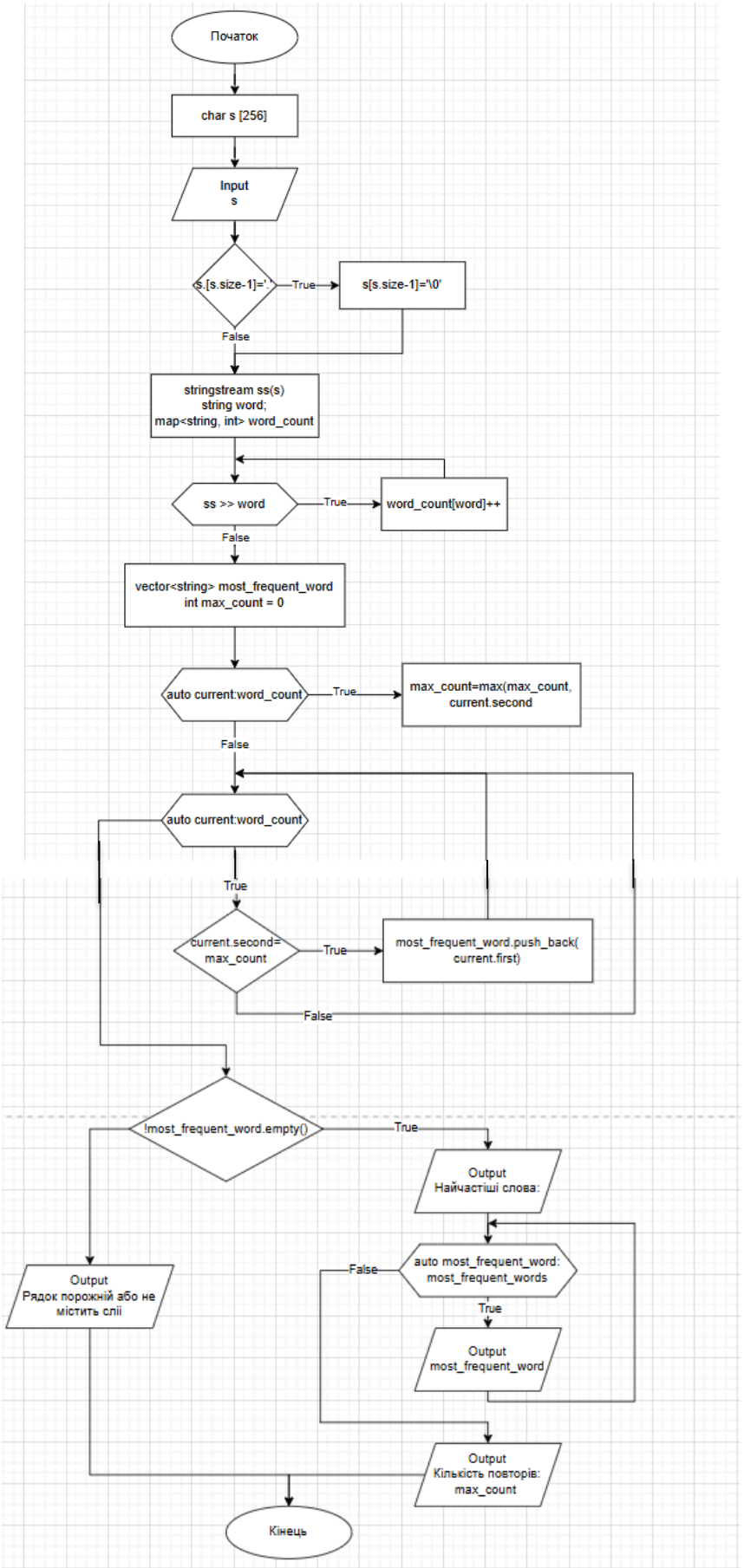
Розв'язок:

```cpp
#include <bits/stdc++.h>
#include<sstream>

using namespace std;

//Визначити яке слово зустрічається в рядку найчастіше.

int main() {
    char s[256];
    cout << "Введіть рядок : ";
    gets(s);

    // Видаляємо крапку в кінці, якщо вона є
    int len = strlen(s);
    if (len > 0 && s[len - 1] == '.') {
        s[len - 1] = '\0';
    }


    stringstream ss(s);
    string word;
    map<string, int> word_count;

    while (ss >> word) {
        word_count[word]++;
    }

    vector<string> most_frequent_words;
    int max_count = 0;

    for (auto current: word_count)
        max_count = max(max_count, current.second);

    for (auto current: word_count) {
        if (current.second == max_count) {
            most_frequent_words.push_back(current.first);
        }
    }

    if (!most_frequent_words.empty()) {
        cout << "Найчастіше(ші) слово(а): { ";
        for (auto most_frequent_word: most_frequent_words) cout << "\"" << most_frequent_word << "\" ";
        cout << "} кількість повторів: " << max_count << endl;
    } else {
        cout << "Рядок порожній або не містить слів." << endl;
    }

    return 0;
}
```

Результат виконання:

```
Введіть рядок : hi i am good hi am
Найчастіше(ші) слово(а): { "am" "hi" } кількість повторів: 2
```

Блок-схема:


Початок
char s [256]
Input
s
s.[s.size-1]='.'
True
s[s.size-1]='\0'
False
stringstream ss(s)
string word;
map<string, int> word_count
ss >> word
True
word_count[word]++
False
vector<string> most_frequent_word
int max_count = 0
auto current:word_count
True
max_count=max(max_count,
current.second
False
auto current:word_count
True
current.second=
max_count
True
most_frequent_word.push_back(
current.first)
False
!most_frequent_word.empty()
True
Output
Найчастіші слова:
auto most_frequent_word:
most_frequent_words
False
True
Output
most_frequent_word
Output
Рядок порожній або не
містить слії
Output
Кількість повторів:
max_count
Кінець

Task 4 - Lab# programming: VNS Lab 8

Час виконання ~ 2 год.

Сформувати двійковий файл із елементів, заданої у варіанті структури, роздрукувати його вмістиме, 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.

Структура "Автомобіль":

- марка;
- рік випуску;
- ціна;
- кольори.

Знищити всі елементи, у яких рік випуску менше заданого, додати елемент на початок файлу.

Розв'язок:

```cpp
#include <bits/stdc++.h>

using namespace std;

struct Car {
    char brand[50];
    int year;
    double price;
    char color[30];
};

const char *filename = "cars.bin";

void createFile() {
    FILE *file = fopen(filename, "wb");
    if (!file) {
        cerr << "Помилка відкриття файлу для запису." << endl;
        return;
    }

    int n;
    cout << "Введіть кількість автомобілів: ";
    cin >> n;

    for (int i = 0; i < n; ++i) {
        Car car;
        cout << "Марка: ";
        cin.ignore();
        cin.getline(car.brand, 50);
        cout << "Рік випуску: ";
        cin >> car.year;
        cout << "Ціна: ";
        cin >> car.price;
        cout << "Колір: ";
        cin.ignore();
        cin.getline(car.color, 30);

        fwrite(&car, sizeof(Car), 1, file);
    }

    fclose(file);
}

void printFile() {
    FILE *file = fopen(filename, "rb");
    if (!file) {
        cerr << "Помилка відкриття файлу для читання." << endl;
        return;
    }

    Car car;
    bool isEmpty = true;

    while (fread(&car, sizeof(Car), 1, file)) {
        isEmpty = false;
        cout << "Марка: " << car.brand
             << ", Рік випуску: " << car.year
             << ", Ціна: " << car.price
             << ", Колір: " << car.color << endl;
    }

    if (isEmpty) {
        cout << "Файл порожній." << endl;
    }

    fclose(file);
}

void deleteOldCars(int minYear) {
    FILE *file = fopen(filename, "rb");
    if (!file) {
        cerr << "Помилка відкриття файлу для читання." << endl;
        return;
    }

    vector<Car> cars;
    Car car;

    while (fread(&car, sizeof(Car), 1, file)) {
        if (car.year >= minYear) {
            cars.push_back(car);
        }
    }

    fclose(file);

    file = fopen(filename, "wb");
    if (!file) {
        cerr << "Помилка відкриття файлу для запису." << endl;
        return;
    }

    for (auto car: cars) {
        fwrite(&car, sizeof(Car), 1, file);
    }

    fclose(file);
}

void addCarToStart() {
    Car newCar;
    cout << "Додавання нового автомобіля:" << endl;
    cout << "Марка: ";
    cin.ignore();
    cin.getline(newCar.brand, 50);
    cout << "Рік випуску: ";
    cin >> newCar.year;
    cout << "Ціна: ";
    cin >> newCar.price;
    cout << "Колір: ";
    cin.ignore();
    cin.getline(newCar.color, 30);

    FILE *file = fopen(filename, "rb");
    if (!file) {
        cerr << "Помилка відкриття файлу для читання." << endl;
        return;
    }

    vector<Car> cars;
    Car car;

    while (fread(&car, sizeof(Car), 1, file)) {
        cars.push_back(car);
    }

    fclose(file);

    file = fopen(filename, "wb");
    if (!file) {
        cerr << "Помилка відкриття файлу для запису." << endl;
        return;
    }

    fwrite(&newCar, sizeof(Car), 1, file);


    for (auto car: cars) {
        fwrite(&car, sizeof(Car), 1, file);
    }

    fclose(file);
}

int main() {
    createFile();

    cout << endl << "Вміст файлу після створення:" << endl;
    printFile();

    int minYear;
    cout << endl << "Мінімальний рік випуску для збереження: ";
    cin >> minYear;
    deleteOldCars(minYear);

    cout << endl << "Вміст файлу після видалення старих автомобілів:" << endl;
    printFile();

    addCarToStart();

    cout << endl << "Вміст файлу після додавання нового автомобіля:" << endl;
    printFile();

    return 0;
}
```

Результат виконання:

```
Введіть кількість автомобілів: 3
Марка: AUDI
Рік випуску: 2010
Ціна: 12000
Колір: green
Марка: Dodge
Рік випуску: 2018
Ціна: 30000
Колір: red
Марка: Toyota
Рік випуску: 2015
Ціна: 20000
Колір: blue

Вміст файлу після створення:
Марка: AUDI, Рік випуску: 2010, Ціна: 12000, Колір: green
Марка: Dodge, Рік випуску: 2018, Ціна: 30000, Колір: red
Марка: Toyota, Рік випуску: 2015, Ціна: 20000, Колір: blue

Мінімальний рік випуску для збереження: 2012

Вміст файлу після видалення старих автомобілів:
Марка: Dodge, Рік випуску: 2018, Ціна: 30000, Колір: red
Марка: Toyota, Рік випуску: 2015, Ціна: 20000, Колір: blue
Додавання нового автомобіля:
Марка: Honda
Рік випуску: 2012
Ціна: 13000
Колір: black

Вміст файлу після додавання нового автомобіля:
Марка: Honda, Рік випуску: 2012, Ціна: 13000, Колір: black
Марка: Dodge, Рік випуску: 2018, Ціна: 30000, Колір: red
Марка: Toyota, Рік випуску: 2015, Ціна: 20000, Колір: blue
```

Task 5 - Lab# programming: VNS Lab 9

Час виконання ~ 3 год.

Розв'язок:

```cpp
#include <bits/stdc++.h>
#include <fstream>

using namespace std;

void createFileF1(const string &filename) {
    ofstream file(filename);
    if (!file) {
        cerr << "Помилка відкриття файлу F1 для запису." << endl;
        return;
    }

    vector<string> lines = {
            "Акація цвіте",
            "Вітер з моря",
            "Дерева у тіні",
            "Аромат весни",
            "Квітка розквітлА",
            "Сонце гріє землю",
            "Зелена трАвА",
            "Хвилі на морі",
            "Пісок гарячий",
            "Магнолія розквітлА"
    };

    for (auto line : lines) {
        file << line << endl;
    }

    file.close();
    cout << "Файл F1 створено." << endl;
}

void copyLinesToF2(const string &fileF1, const string &fileF2, int N1, int N2) {
    ifstream file1(fileF1);
    ofstream file2(fileF2);

    if (!file1) {
        cerr << "Помилка відкриття файлу F1 для читання." << endl;
        return;
    }
    if (!file2) {
        cerr << "Помилка відкриття файлу F2 для запису." << endl;
        return;
    }

    string line;
    int lineNumber = 0;

    while (getline(file1, line)) {
        lineNumber++;
        if (lineNumber >= N1 && lineNumber <= N2 && line.back() == 'A') {
            file2 << line << endl;
        }
    }

    file1.close();
    file2.close();
    cout << "Рядки скопійовано у файл F2." << endl;
}

int findLineWithMaxA(const string &fileF2) {
    ifstream file(fileF2);

    if (!file) {
        cerr << "Помилка відкриття файлу F2 для читання." << endl;
        return -1;
    }

    string line;
    int lineNumber = 0, maxLineNumber = -1, maxACount = 0;

    while (getline(file, line)) {
        lineNumber++;
        int aCount = 0;
        for (char ch : line) {
            if (ch == 'A'){
                aCount++;
            }
        }
        if (aCount > maxACount) {
            maxACount = aCount;
            maxLineNumber = lineNumber;
        }
    }

    file.close();
    return maxLineNumber;
}

int main() {
    const string fileF1 = "F1.txt";
    const string fileF2 = "F2.txt";

    createFileF1(fileF1);

    int N1, N2;
    cout << "Введіть номер рядка N1, N2: ";
    cin >> N1>>N2;

    copyLinesToF2(fileF1, fileF2, N1, N2);


    int maxLine = findLineWithMaxA(fileF2);
    if (maxLine != -1) {
        cout << "Номер рядка з найбільшою кількістю 'A' у файлі F2: " << maxLine << endl;
    } else {
        cout << "Файл F2 порожній або не знайдено рядків." << endl;
    }

    return 0;
}
```

Результат виконання:

```
Файл F1 створено.
Введіть номер рядка N1, N2: 2 8
Рядки скопійовано у файл F2.
Номер рядка з найбільшою кількістю 'A' у файлі F2: 2
```

Task 6 - Lab# programming: Algotester [Lab 4.2]
***Час виконання* ~ 1 год.**

Розв'язок №1:

```cpp
#include<bits/stdc++.h>
//4.2
using namespace std;

int main(){
    int n,k;
    cin>>n>>k;
    vector<int>a(n);
    for(int i=0;i<n;i++){
        cin>>a[i];
    }
    sort(a.begin(), a.end());
    auto it= unique(a.begin(), a.end());
    a.erase(it, a.end());

    n=a.size();
    k%=n;
     cout<<n<<endl;
    rotate(a.begin(),a.begin()+k, a.end());
    for(auto i:a){
        cout<<i<<" ";
    }

}
```

Розв'язок №2:

```cpp
#include<bits/stdc++.h>
//4.2
using namespace std;

int main(){
    int n,k;
    cin>>n>>k;
    vector<int>a(n),b,c;
    for(int i=0;i<n;i++){
        cin>>a[i];
    }
    sort(a.begin(), a.end());
    b.push_back(a[0]);
    for(int i=1;i<n;i++){
        if(a[i]!=a[i-1]) b.push_back(a[i]);
    }
    n=b.size();
    for(int i=0;i<n;i++){
        c.push_back(b[ (i+k)%n ]);
    }
    cout<<n<<endl;
    for(auto i:c)
        cout<<i<<" ";
}
```

Результат виконання:

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

Algotester [Lab 4.3]
Час виконання ~ 2 год.

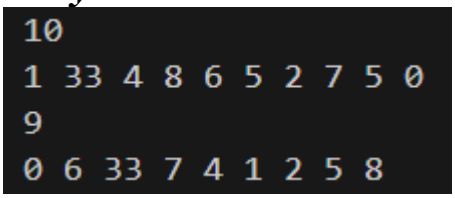
Розв'язок №1:

```cpp
#include<bits/stdc++.h>
//4.3
using namespace std;

int main(){
    int n;
    cin>>n;
    vector<int>a(n);
    for(int i=0;i<n;i++){
        cin>>a[i];
    }
    sort(a.begin(),a.end());
    auto it= unique(a.begin(),a.end());
    a.erase(it,a.end());

     vector<int> mod0, mod1, mod2;
     for (int x : a) {
         if (x % 3 == 0) mod0.push_back(x);
         else if (x % 3 == 1) mod1.push_back(x);
         else mod2.push_back(x);
     }

     sort(mod0.begin(), mod0.end());
     sort(mod1.rbegin(), mod1.rend());//по спаданню
     sort(mod2.begin(), mod2.end());

     a.clear();
     a.insert(a.end(), mod0.begin(), mod0.end());
     a.insert(a.end(), mod1.begin(), mod1.end());
     a.insert(a.end(), mod2.begin(), mod2.end());

     cout<<a.size()<<endl;
    for(auto i:a) cout<<i<<" ";
}
```

Розв'язок №2

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

void SplitByModul(const vector<int>& arr, vector<int>& mod0, vector<int>& mod1, vector<int>& mod2) {
    for (int num : arr) {
        if (num % 3 == 0) mod0.push_back(num);
        else if (num % 3 == 1) mod1.push_back(num);
        else mod2.push_back(num);
    }
}

void quickSort(vector<int>& arr, int left, int right, bool ascending) {
    if (left >= right) return;

    int mid = arr[(left + right) / 2];
    int i = left, j = right;

    while (i <= j) {
        while (ascending ? arr[i] < mid : arr[i] > mid) i++;
        while (ascending ? arr[j] > mid : arr[j] < mid) j--;

        if (i <= j) {
            swap(arr[i], arr[j]);
            i++;
            j--;
        }
    }

    if (left < j) quickSort(arr, left, j, ascending);
    if (i < right) quickSort(arr, i, right, ascending);
}

vector<int> removeDuplicates(const vector<int>& arr) {
    vector<int> result;
    for (int num : arr) {
        if (result.empty() || result.back() != num) {
            result.push_back(num);
        }
    }
    return result;
}

int main() {
    int N;
    cin >> N;
    vector<int> arr(N);

    for (int i = 0; i < N; i++) {
        cin >> arr[i];
    }
    vector<int> mod0, mod1, mod2;
    SplitByModul(arr, mod0, mod1, mod2);

    quickSort(mod0, 0, mod0.size() - 1, true);  // за зростанням
    quickSort(mod1, 0, mod1.size() - 1, false); // за спаданням
    quickSort(mod2, 0, mod2.size() - 1, true);  // за зростанням


    vector<int> result;
    result.insert(result.end(), mod0.begin(), mod0.end());
    result.insert(result.end(), mod1.begin(), mod1.end());
    result.insert(result.end(), mod2.begin(), mod2.end());
    result = removeDuplicates(result);

    cout << result.size() << endl;
    for (int num : result) {
        cout << num << " ";
    }
    cout << endl;
}

```

Результат виконання:



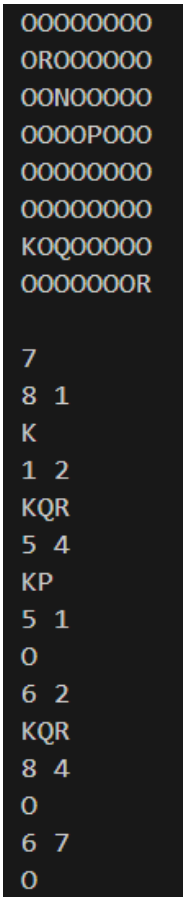
Task 7 - Lab# programming: Algotester [Lab 6.2]

***Час виконання* ~ 2 год.**

Розв'язок

```cpp
#include<bits/stdc++.h>

using namespace std;
vector<string> board(8);

bool could_king(int x, int y) {
    if (((0 <= x) && (x <= 7) && (0 <= y) && (y <= 7)) && (board[x][y] == 'K'))
        return true;
    else
        return false;
}

bool king(int x, int y) {
    vector<pair<int, int>> offset = {{1,  0},
                                     {1,  1},
                                     {1,  -1},
                                     {-1, 0},
                                     {-1, 1},
                                     {-1, -1},
                                     {0,  1},
                                     {0,  -1}};
    bool flag = false;
    for (int i = 0; i < 8; i++)
        if (could_king(x + offset[i].first, y + offset[i].second))
            flag = true;
    return flag;
}

bool tyra(int x, int y) {
    bool flag = false;
    for (int i = 0; i < 7; i++)
        if ((board[x][i] == 'R') || (board[i][y]) == 'R')
            flag = true;
    return flag;
}

bool could_slon(int x, int y) {
    if (((0 <= x) && (x <= 7) && (0 <= y) && (y <= 7)) && (board[x][y] == 'B'))
        return true;
    else
        return false;
}

bool slon(int x, int y) {
    bool flag = false;
    vector<pair<int, int>> offset = {{1,  1},
                                     {1,  -1},
                                     {-1, 1},
                                     {-1, -1}};
    for (int i = -7; i < 8; i++) {
        for (int j = 0; j < 4; j++) {
            if (could_slon(x + i * offset[j].first, y + i * offset[j].second))
                flag = true;
        }
    }
    return flag;
}

bool could_horse(int x, int y) {
    if (((0 <= x) && (x <= 7) && (0 <= y) && (y <= 7)) && (board[x][y] == 'N'))
        return true;
    else
        return false;
}

bool horse(int x, int y) {
    bool flag = false;
    vector<pair<int, int>> offset = {{2,  1},
                                     {2,  -1},
                                     {-2, 1},
                                     {-2, -1},

                                     {1,  2},
                                     {1,  -2},
                                     {-1, 2},
                                     {-1, -2}};
    for (int i = 0; i < 8; i++) {
        if (could_horse(x + offset[i].first, y + offset[i].second))
            flag = true;
    }
    return (flag);
}

bool pishak(int x, int y) {
    if (((0 <= x - 1) && (0 <= y - 1) && (board[x - 1][y - 1] == 'P')) ||
        ((0 <= x - 1) && (y + 1 <= 7) && (board[x - 1][y + 1] == 'P')))
        return true;
    else
        return false;
}

string find_result(int x, int y) {
    string result = "";
    if (slon(x, y)) result += "B";
    if (king(x, y)) result += "K";
    if (horse(x, y)) result += "K";
    if (pishak(x, y)) result += "P";
    if (slon(x, y) || tyra(x, y)) result += "Q";
    if (tyra(x, y)) result += "R";
    return result;
}

int main() {
    for (int i = 0; i < 8; i++) {
        cin >> board[i];
    }
    cin;
    int q, x, y;
    cin >> q;
    for (int i = 0; i < q; i++) {
        cin >> x >> y;
        if(board[x-1][y-1]!='O')
            cout<<board[x-1][y-1]<<endl;
        string rez=find_result(x-1,y-1);
        if(rez=="")
            cout<<"O"<<endl;
        else
            cout<<rez<<endl;
    }
}
```

Результат виконання:



Task 8 - Practice# programming: Class Practice Task

Час виконання~ 2 год.

*1)**Реалізувати функцію створення файла і запису в нього даних:***
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.

2) ***Реалізувати функцію створення файла і запису в нього даних:***
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.

Розв'язок:

```cpp
#include<bits/stdc++.h>
#include<fstream>

using namespace std;

void write_to_file(char *file_name, char *content) {
    ofstream file(file_name);
    if (!file) {
        cout << "Failure" << endl;
        return;
    }
    file << content << endl;
    file.close();
    cout << "Success" << endl;

}

void copy_file(char *file_from_copy, char *file_to_insert) {
    ofstream file_to(file_to_insert);
    ifstream file_from(file_from_copy);
    if ((!file_to) || (!file_from)) {
        cout << "Failure" << endl;
        return;
    }
    bool is_empty_file_from = true;

    string line;
    while (getline(file_from, line)) {
        file_to << line << endl;

        is_empty_file_from = false;
    }
    if (is_empty_file_from)
        cout << "Failure file_from is empty" << endl;
    else {
        cout << "Success" << endl;
    }


}

int main() {
    char content[50];
    char file_name[20];
    cout << "Filename: ";
    cin.getline(file_name, sizeof(file_name));
    cout << "Content: ";
    cin.getline(content, sizeof(content));

    write_to_file(file_name, content);


    char filename_from[20], filename_to[20];
    cout << "Filename from which copy: ";
    cin.getline(filename_from, sizeof(filename_from));
    cout << "Filename in which insert: ";
    cin.getline(filename_to, sizeof(filename_to));
    copy_file(filename_from, filename_to);
}
```

Task 9 - Practice# programming: Self Practice Task

Час виконання 1,5 год

Algotester 2133 [Катеринка]

В цій задачі потрібно знайти мінімальну відстань між **q** парами вершин у зваженому графі, так щоб шлях проходив через задану третю вершину.

Розв'язок:

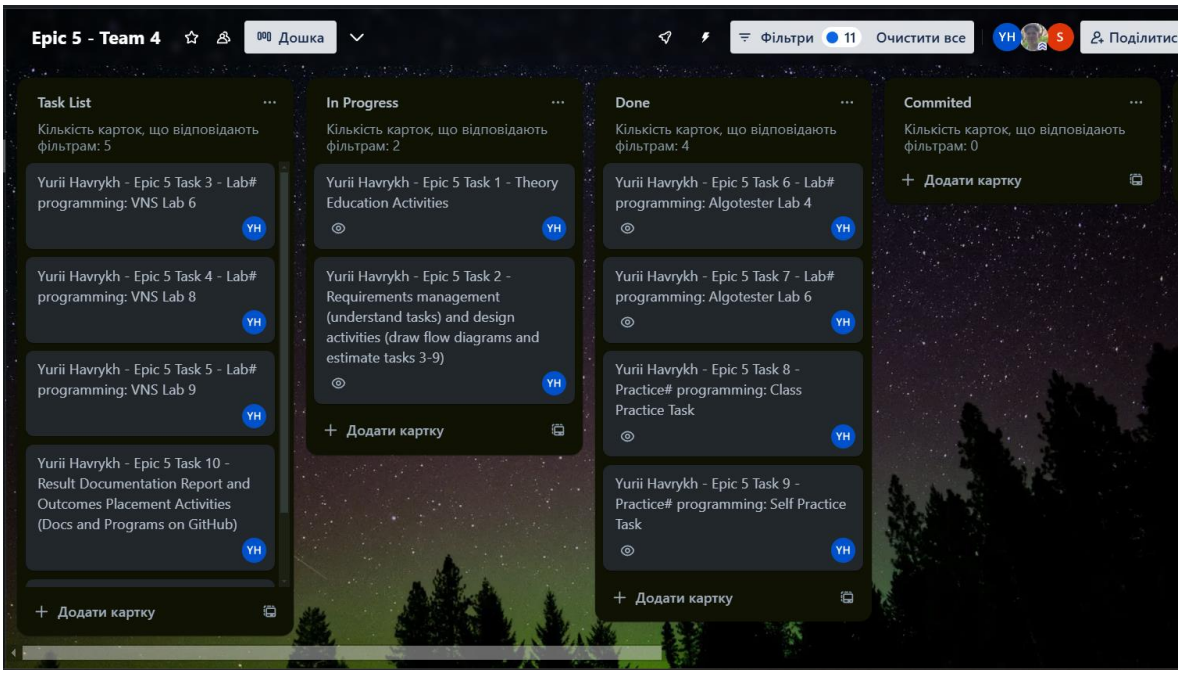
```cpp
#include<bits/stdc++.h>

using namespace std;
const int inf = 2e8;
//2133 Катеринка
void dijkstra(set<pair<int, int>> &que, const vector<vector<pair<int, int>>> &graph, vector<int> &dist) {
    while (!que.empty()) {

        int cur = que.begin()->second;
        que.erase(que.begin());
        for (auto curent: graph[cur]) {
            if (dist[curent.first] > dist[cur] + curent.second) {
                que.erase({dist[curent.first], curent.first});
                dist[curent.first] = dist[cur] + curent.second;
                que.insert({dist[curent.first], curent.first});
            }
        }
    }
}

int main() {
    int n, m, q, kat, from, to, d;
    cin >> n >> m >> q >> kat;
    vector<int> dist(n, inf);
    vector<vector<pair<int, int>>> graph(n);
    set<pair<int, int>> que;
    vector<pair<int, int>> find(q);
    for (int i = 0; i < m; i++) {
        cin >> from >> to >> d;
        graph[from - 1].push_back({to - 1, d});
        graph[to - 1].push_back({from - 1, d});
    }
    for (int i = 0; i < q; i++) {
        cin >> find[i].first >> find[i].second;
    }
    kat--;
    que.insert({0, kat});
    dist[kat] = 0;
    dijkstra(que, graph, dist);
    for (int i = 0; i < q; i++) {
        cout << dist[find[i].first - 1] + dist[find[i].second - 1] << endl;
    }
}
```

Результат виконання:

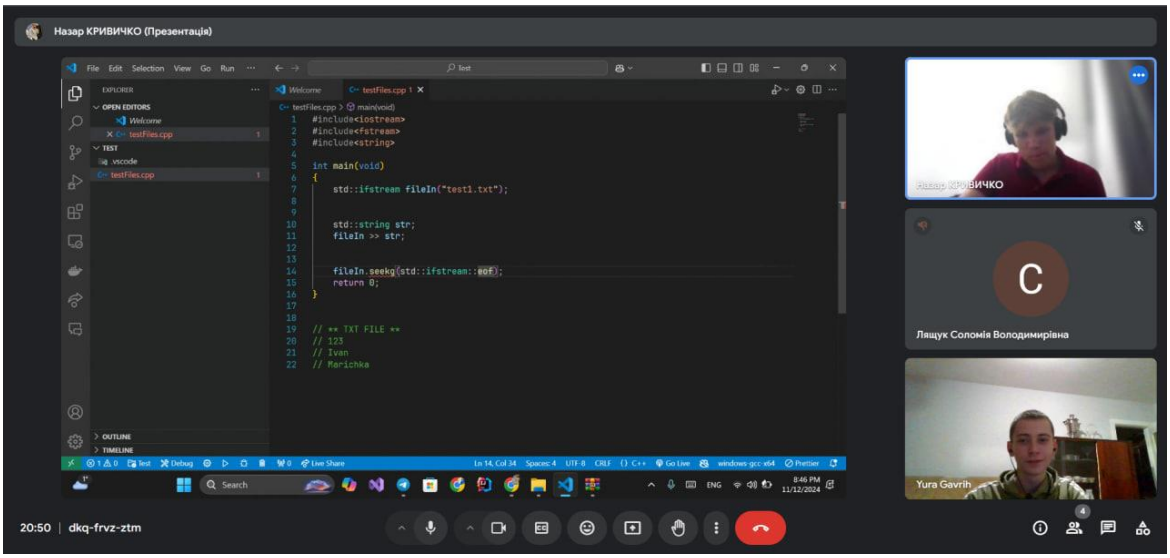
```
4 5 3 1
1 2 1
2 3 3
3 1 1
3 4 4
1 4 10
4 3
2 3
1 4
6
2
5
```

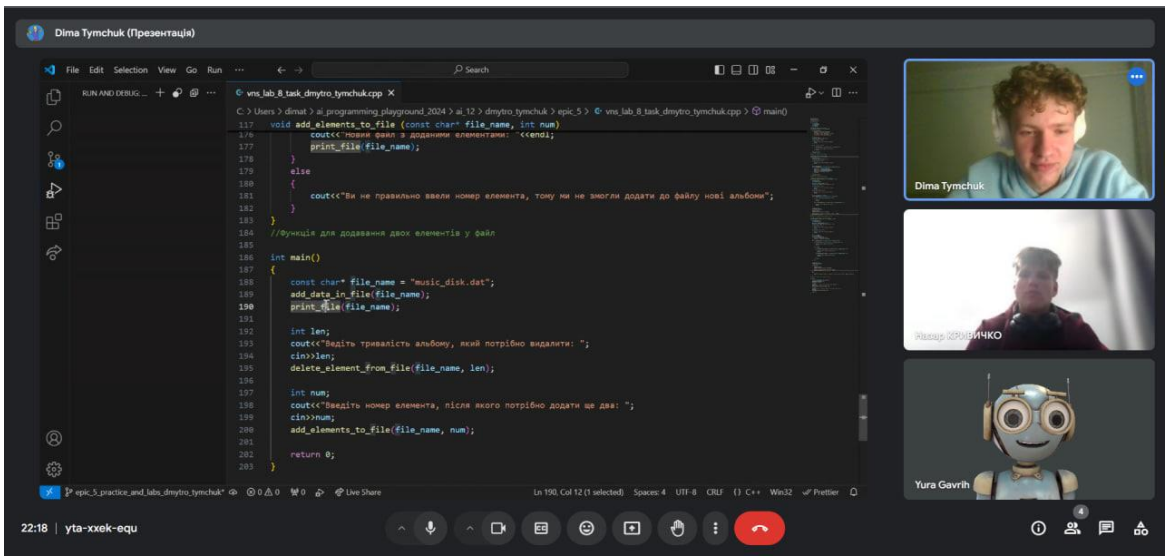
Task 10 - Result Documentation Report and Outcomes Placement Activities

Час виконання ~ 1,5 год.



Team meetings





Pull Request

Висновок:

В результаті виконання цієї роботи я навчився працювати з рядками типу char та string, закріпив свої знання роботи з файлами на C та C++.