Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт
про виконання лабораторних та практичних робіт блоку № 5

з дисципліни: «Основи програмування»

Виконав:

Студент групи ШІ-11

Лопатін Володимир Дмитрович

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 роботи:

Ознайомитися з принципами роботи з файлами в програмуванні, зокрема з текстовими та бінарними файлами, символами й рядковими змінними. Дослідити можливості стандартної бібліотеки для роботи з файлами, навчитися застосовувати методи відкриття, читання, запису та закриття файлів. Розробити власні бібліотеки та інтегрувати їх у програмні проекти.

Теоретичні відомості:

- Файли
- Текстові файли
- Бінарні файли
- Стандартна бібліотека
- Створення та використання бібліотек

Файли:

Не був знайомий, на парах отримав базове поняття, дорозібрався через різні ресурси.
Витрачено 40 хв.

Текстові файли:

Уперше зіткнувся в цьому епіку.

Витратив 40 хвилин.

Бінарні файли:

Не мав найменшого поняття.

Витратив 45 хвилин.

Стандартна бібліотека:

Не шарив узагалі.

Для повного розуміння вивчав 45 хвилин.

Створення та використання бібліотек:

Пояснили ChatGPT та викладач на парі.

На повне ознайомлення загалом витратив 1 годину.

Виконання роботи:

1) Опрацювання завдання та вимог до програм та серидовища:

Завдання №1 з ВНС

«Лабораторна № 6 варіант 19»

Потрібно було ввести текст за допомогою gets(), а потім видалити з нього всі не ідентифікатори.

Вимоги:

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.

- 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.

Завдання №2 з ВНС

«Лабораторна №8 варіант 19»

Потрібно створити відповідні функції для створення бінарного файлу та запису туди елементів структури «Фільм», потім видалити два елементи з кінця та додати елемент у файл після певного елемента.

Завдання №3 з ВНС

«Лабораторна №9 варіант 19**»**

Потрібно було створити текстовий файл та додати туди 10 рядків тексту щонайменше, потім треба було скопіювати додругого файлу ті рядки, що не містять повторних слів з першим словом і підрахувати кількість приголосних першого рядка для другого файлу.

Завдання №4

«Практичне завдання»

Вимоги:

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.

- 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.

Завдання №5

«Лабораторна 4 з Algotester»

Завдання ввести кількість елементів майбутнього масиву та інше ціле число K, потім ввести всі елементи масиву, видалити з нього всі повтори та відсортувати. Потім треба перенести перші K елементів в кінець і вивести результуючий масив.

Завдання №6

«Лабораторна 6 з Algotester»

Задача полягала в тому, щоб за введеним розташуванням шахових фігур визначити фігури, що будуть атакувати конкретну клітинку.

Завдання №7

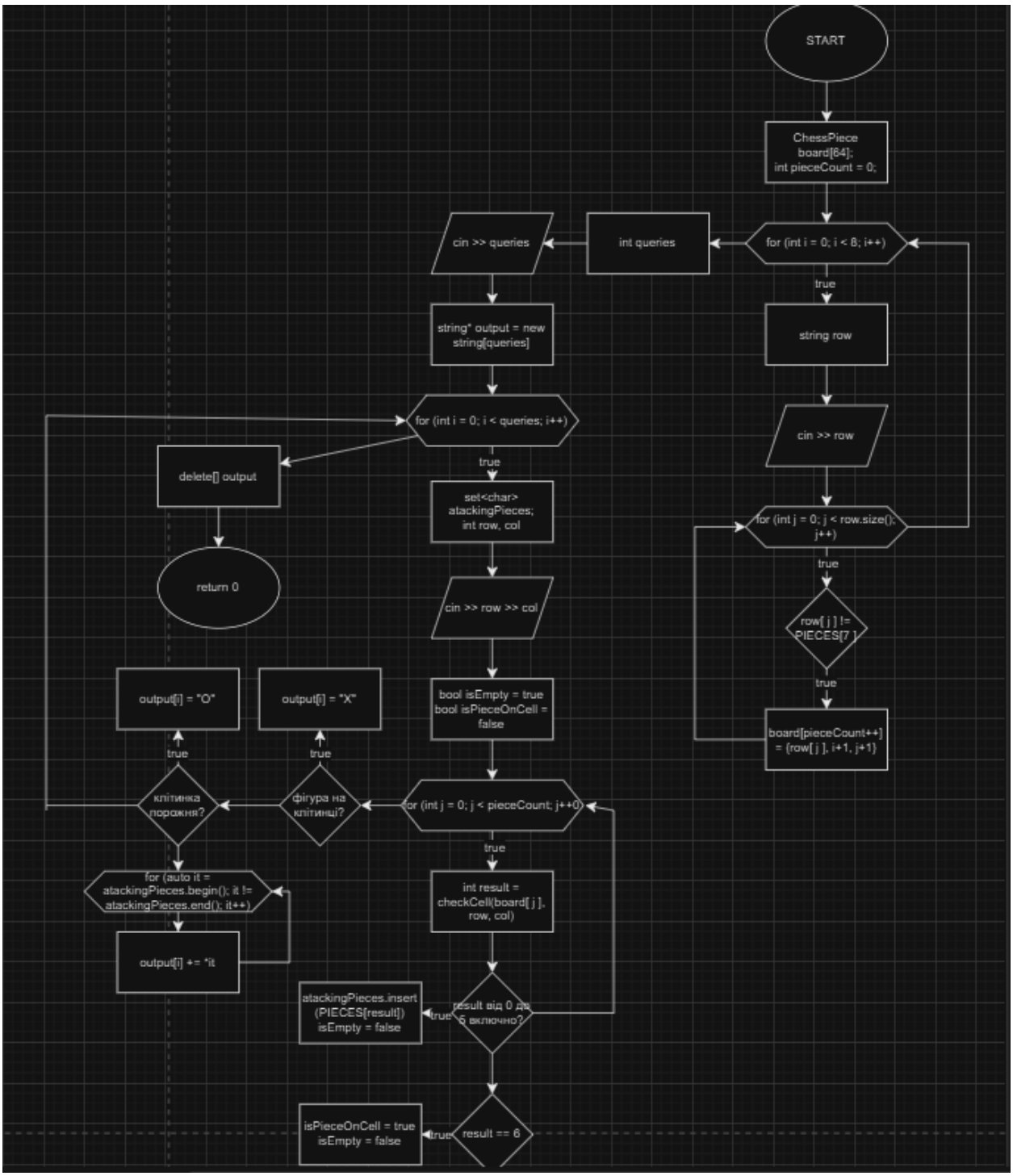
«Додаткове завдання з Algotester»

Потрібно було знайти початковий ген, маючи два модифіковані, тобто знайти найбільшу спільну послідовність літер у рядку.

2) Дизайн та планова оцінка часу виконання завдань:

Завдання №6

Хотів зробити за 3 години.



1) Код програм з посиланням на зовнішні ресурси:

Завдання №1

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

bool isIdentifier(string& word) {
    if (word.empty()) return false;
    if (!isalpha(word[0]) && word[0] != '_') return false;
    for (int i = 1; i < word.length(); i++) {
        if (!isalnum(word[i]) && word[i] != '_') return false;
    }
    const vector<std::string> keywords = {
        "int", "if", "else", "while", "for", "return", "class", "float"
    };
    for (const auto& keyword : keywords) {
        if (word == keyword) return false;
    }
    return true;
}

int main() {
    char s[256];
    cout << "Enter the text: ";
    gets(s);

    vector<string> identifiers;
    string word;
    for (int i = 0; i < strlen(s); i++) {
        if (s[i] == '.' || isspace(s[i])) {
            if (isIdentifier(word)) {
                identifiers.push_back(word);
            }
            word.clear();
        } else {
            word += s[i];
        }
    }
    string result;
    for (int j = 0; j < identifiers.size(); j++) {
        result += identifiers[j] + ' ';
    }
    cout << "Identifiers in the text: " << result;
    return 0;
}
```

Завдання №2

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

using namespace std;

struct Film {
    char title[50];
    char director[50];
    char country[30];
    double profit;
};

void createFile(const string& filename, const vector<Film>& films) {
    ofstream file(filename, ios::binary);
    if (!file) {
        cerr << "Помилка відкриття файлу для запису.\n";
        return;
    }
    for (const auto& film : films) {
        file.write(reinterpret_cast<const char*>(&film), sizeof(Film));
    }
    file.close();
}

void printFile(const string& filename) {
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Помилка відкриття файлу для читання.\n";
        return;
    }
    Film film;
    cout << "Вміст файлу:\n";
    while (file.read(reinterpret_cast<char*>(&film), sizeof(Film))) {
        cout << "Назва: " << film.title << "\n"
             << "Режисер: " << film.director << "\n"
             << "Країна: " << film.country << "\n"
             << "Прибуток: " << film.profit << "\n\n";
    }
    file.close();
}

void deleteLastTwo(const string& filename) {
    vector<Film> films;
```

```
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Помилка відкриття файлу для читання.\n";
        return;
    }
    Film film;
    while (file.read(reinterpret_cast<char*>(&film), sizeof(Film))) {
        films.push_back(film);
    }
    file.close();
    if (films.size() < 2) {
        cerr << "Недостатньо елементів для видалення.\n";
        return;
    }
    films.pop_back();
    films.pop_back();
    ofstream outFile(filename, ios::binary);
    if (!outFile) {
        cerr << "Помилка відкриття файлу для запису.\n";
        return;
    }
    for (const auto& f : films) {
        outFile.write(reinterpret_cast<const char*>(&f), sizeof(Film));
    }
    outFile.close();
}

void addAfterTitle(const string& filename, const string& targetTitle, const Film& newFilm)
    vector<Film> films;
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Помилка відкриття файлу для читання.\n";
        return;
    }
    Film film;
    while (file.read(reinterpret_cast<char*>(&film), sizeof(Film))) {
        films.push_back(film);
    }
    file.close();
    bool found = false;
    ofstream outFile(filename, ios::binary);
    if (!outFile) {
        cerr << "Помилка відкриття файлу для запису.\n";
        return;
```

```
    for (size_t i = 0; i < films.size(); ++i) {
        outFile.write(reinterpret_cast<const char*>(&films[i]), sizeof(Film));
        if (string(films[i].title) == targetTitle) {
            outFile.write(reinterpret_cast<const char*>(&newFilm), sizeof(Film));
            found = true;
        }
    }
    outFile.close();
    if (!found) {
        cerr << "Елемент із заданою назвою не знайдено.\n";
    }
}

int main() {
    const string filename = "films.bin";
    vector<Film> films = {
        {"Inception", "Christopher Nolan", "USA", 829.89},
        {"Parasite", "Bong Joon-ho", "South Korea", 258.4},
        {"Lobster", "Yorgos Lantimos", "UK", 123.6},
        {"Moonrise Kingdom", "Wes Anderson", "USA", 68.3}
    };
    createFile(filename, films);
    cout << "Початковий файл:\n";
    printFile(filename);
    deleteLastTwo(filename);
    cout << "\nПісля видалення останніх двох елементів:\n";
    printFile(filename);
    Film newFilm = {"Avatar", "James Cameron", "USA", 2847.24};
    addAfterTitle(filename, "Inception", newFilm);
    cout << "\nПісля додавання нового елемента після 'Inception':\n";
    printFile(filename);
    return 0;
}
```

Завдання №3

```cpp
#include <iostream>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>
#include <cctype>
using namespace std;

int countConsonants(const string& line) {
    int consonants = 0;
    for (char ch : line) {
        if (isalpha(ch)) {
            char lower = tolower(ch);
            if (lower != 'a' && lower != 'e' && lower != 'i' && lower !
                consonants++;
            }
        }
    }
    return consonants;
}

int main() {
    ofstream fileF1("F1.txt");
    if (fileF1.is_open()) {
        fileF1 << "apple banana cherry\n";
        fileF1 << "orange apple grape\n";
        fileF1 << "pear plum apple\n";
        fileF1 << "banana orange pear\n";
        fileF1 << "mango kiwi peach\n";
        fileF1 << "lemon lime orange\n";
        fileF1 << "grape banana kiwi\n";
        fileF1 << "strawberry orange lemon\n";
        fileF1 << "blueberry peach mango\n";
        fileF1 << "pineapple cherry grape\n";
        fileF1.close();
    }

    ifstream fileF1Read("F1.txt");
    ofstream fileF2("F2.txt");
    vector<string> filteredLines;
```

```
    ifstream fileF1Read("F1.txt");
    ofstream fileF2("F2.txt");
    vector<string> filteredLines;
    if (fileF1Read.is_open() && fileF2.is_open()) {
        string line;
        while (getline(fileF1Read, line)) {
            istringstream iss(line);
            string firstWord;
            iss >> firstWord;

            string word;
            bool hasDuplicate = false;
            while (iss >> word) {
                if (word == firstWord) {
                    hasDuplicate = true;
                    break;
                }
            }
            if (!hasDuplicate) {
                filteredLines.push_back(line);
                fileF2 << line << "\n";
            }
        }
        fileF1Read.close();
        fileF2.close();
    }

    if (!filteredLines.empty()) {
        int consonantsCount = countConsonants(filteredLines[0]);
        cout << consonantsCount << endl;
    } else {
        cout << "0" << endl;
    }

    return 0;
}
```

Завдання №4

```cpp
#include <iostream>
#include <fstream>
#include <cstring>
using namespace std;

enum FileOpResult {
    Success,
    Failure
};

FileOpResult write_to_file(const char* name, const char* content) {
    if (name == nullptr || content == nullptr || strlen(name) == 0) {
        return Failure;
    }
    ofstream file(name);
    if (!file.is_open()) {
        return Failure;
    }
    file << content;
    if (file.fail()) {
        file.close();
        return Failure;
    }
    file.close();
    if (file.fail()) {
        return Failure;
    }

    return Success;
}

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

    ifstream src(file_from, ios::binary);
    if (!src.is_open()) {
        return Failure;
    }
```

```
    ofstream input(file_to, ios::binary);
    if (!input.is_open()) {
        return Failure;
    }
    input << src.rdbuf();
    if (src.fail() || input.fail()) {
        src.close();
        input.close();
        return Failure;
    }
    src.close();
    input.close();
    if (src.fail() || input.fail()) {
        return Failure;
    }

    return Success;
}

int main() {
    char filename[256];
    char content[1024];
    cout << "Введіть ім'я файлу для запису: ";
    cin.getline(filename, sizeof(filename));
    cout << "Введіть текст для запису у файл: ";
    cin.getline(content, sizeof(content));
    if (write_to_file(filename, content) == Success) {
        cout << "Файл успішно створено і записано." << endl;
    } else {
        cerr << "Помилка запису у файл." << endl;
        return 1;
    }
    char file_from[256], file_to[256];
    cout << "Введіть ім'я файлу-джерела для копіювання: ";
    cin.getline(file_from, sizeof(file_from));
    cout << "Введіть ім'я файлу-призначення: ";
    cin.getline(file_to, sizeof(file_to));
    if (copy_file(file_from, file_to) == Success) {
        cout << "Файл успішно скопійовано." << endl;
    } else {
        cerr << "Помилка копіювання файлу." << endl;
        return 1;
    }
    return 0;
```

Завдання №5

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
int main() {
    int N, K;
    vector<int> arr;
    cin >> N >> K;
    if (N < 1 || N > 1000 || K < 1 || K > 1000) return 0;
    for (int i = 0; i < N; i++) {
        int el;
        cin >> el;
        if (el < 0 || el > 100) return 0;
        arr.push_back(el);
    }
    sort(arr.begin(), arr.end());
    vector<int> deleted;
    for (int j = 0; j < N; j++) {
        if ( arr[j]!=arr[j-1] || j == 0) {
            deleted.push_back(arr[j]);
        }

    }
    N = deleted.size();
    K %= N;
    for (int i = 0; i < K; ++i) {
        int first = deleted[0];
        for (int j = 0; j < N; ++j) {
            deleted[j] = deleted[j+1];
        }
        deleted[N-1] = first;
    }
    cout << N << endl;
    for (int s = 0; s < N; s++) {
        cout << deleted[s] << " ";
    }
    return 0;
}
```

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

int main() {
    int N, K;
    cin >> N >> K;
    if (N < 1 || N > 1000 || K < 1 || K > 1000) return 0;
    int* arr = new int[N];
    for (int i = 0; i < N; i++) {
        int el;
        cin >> el;
        if (el < 0 || el > 100) {
            delete[] arr;
            return 0;
        }
        arr[i] = el;
    }

    int stack[1000];
    int top = -1;
    int low = 0, high = N - 1;
    stack[++top] = low;
    stack[++top] = high;

    while (top >= 0) {
        high = stack[top--];
        low = stack[top--];

        int pivot = arr[high];
        int index = low - 1;
        for (int j = low; j < high; ++j) {
            if (arr[j] <= pivot) {
                index++;
                int temp = arr[index];
                arr[index] = arr[j];
                arr[j] = temp;
            }
        }
        int temp = arr[index + 1];
        arr[index + 1] = arr[high];
        arr[high] = temp;
        int pi = index + 1;
```

```
        if (pi - 1 > low) {
            stack[++top] = low;
            stack[++top] = pi - 1;
        }
        if (pi + 1 < high) {
            stack[++top] = pi + 1;
            stack[++top] = high;
        }
    }

    int* deleted = new int[N];
    int newSize = 0;
    for (int j = 0; j < N; j++) {
        if (j == 0 || arr[j] != arr[j - 1]) {
            deleted[newSize++] = arr[j];
        }
    }

    N = newSize;
    K %= N;
    for (int i = 0; i < K; ++i) {
        int first = deleted[0];
        for (int j = 0; j < N - 1; ++j) {
            deleted[j] = deleted[j + 1];
        }
        deleted[N - 1] = first;
    }

    cout << N << endl;
    for (int s = 0; s < N; s++) {
        cout << deleted[s] << " ";
    }

    delete[] arr;
    delete[] deleted;
    return 0;
```

Завдання №6

```cpp
#include <iostream>
#include <cstring>
#include <cmath>
#include <set>
using namespace std;

struct ChessPiece {
    char type;
    int row, col;
};
const char PIECES[] = {'P', 'R', 'N', 'B', 'K', 'Q', 'X', 'O'};

int checkCell(ChessPiece &piece, int row, int col) {
    if (piece.row == row && piece.col == col)
        return 6;
    switch (piece.type) {
    case 'P':
        if (piece.row == row - 1 && abs(piece.col - col) == 1)
            return 0;
        break;
    case 'R':
        if (piece.row == row || piece.col == col)
            return 1;
        break;
    case 'N':
        if ((abs(piece.row - row) == 2 && abs(piece.col - col) == 1) || (abs(piece.row - row) == 1 && abs(piece.col - col)
            return 2;
        break;
    case 'B':
        if (abs(piece.row - row) == abs(piece.col - col))
            return 3;
        break;
    case 'K':
        if (abs(piece.row - row) <= 1 && abs(piece.col - col) <= 1)
            return 4;
        break;
    case 'Q':
        if (piece.row == row || piece.col == col || abs(piece.row - row) == abs(piece.col - col))
            return 5;
        break;
    default:
        break;
    }
    return -1;
}
```

```cpp
int main() {
    ChessPiece board[64];
    int pieceCount = 0;
    for (int i = 0; i < 8; i++) {
        string row;
        cin >> row;
        for (int j = 0; j < row.size(); j++) {
            if (row[j] != PIECES[7]) {
                board[pieceCount++] = {row[j], i + 1, j + 1};
            }
        }
    }
    int queries;
    cin >> queries;
    string* output = new string[queries];
    for (int i = 0; i < queries; i++) {
        set<char> attackingPieces;
        int row, col;
        cin >> row >> col;
        bool isEmpty = true;
        bool isPieceOnCell = false;
        for (int j = 0; j < pieceCount; j++) {
            int result = checkCell(board[j], row, col);
            if (result >= 0 && result <= 5) {
                attackingPieces.insert(PIECES[result]);
                isEmpty = false;
            } else if (result == 6) {
                isPieceOnCell = true;
                isEmpty = false;
            }
        }
        if (isPieceOnCell) {
            output[i] = "X";
        } else if (isEmpty) {
            output[i] = "O";
        } else {
            for (auto it = attackingPieces.begin(); it != attackingPieces.end(); ++it) {
                output[i] += *it;
            }
        }
    }
    for (int i = 0; i < queries; i++) {
        cout << output[i] << endl;
    }
    delete[] output;
```

Завдання №7

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

using namespace std;

bool isValidGenome(const string& genome) {
    if (genome.empty() || genome.size() > 30) {
        return false;
    }
    for (char c : genome) {
        if (c != 'C' && c != 'G' && c != 'U' && c != 'A') {
            return false;
        }
    }
    return true;
}

string findOriginalGenome(const string& genome1, const string& genome2) {
    int n = genome1.size(), m = genome2.size();
    vector<vector<int>> dp(n + 1, vector<int>(m + 1, 0));

    for (int i = 1; i <= n; ++i) {
        for (int j = 1; j <= m; ++j) {
            if (genome1[i - 1] == genome2[j - 1]) {
                dp[i][j] = dp[i - 1][j - 1] + 1;
            } else {
                dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            }
        }
    }

    string lcs;
    int i = n, j = m;
    while (i > 0 && j > 0) {
        if (genome1[i - 1] == genome2[j - 1]) {
            lcs += genome1[i - 1];
            --i;
            --j;
        } else if (dp[i - 1][j] > dp[i][j - 1]) {
            --i;
        } else {
            --j;
        }
```

```
    reverse(lcs.begin(), lcs.end());
    return lcs.empty() ? "No original genome" : lcs;
}

int main() {
    string genome1, genome2;
    cin >> genome1 >> genome2;

    if (!isValidGenome(genome1) || !isValidGenome(genome2)) {
        return 0;
    }

    cout << findOriginalGenome(genome1, genome2) << endl;
    return 0;
}
```

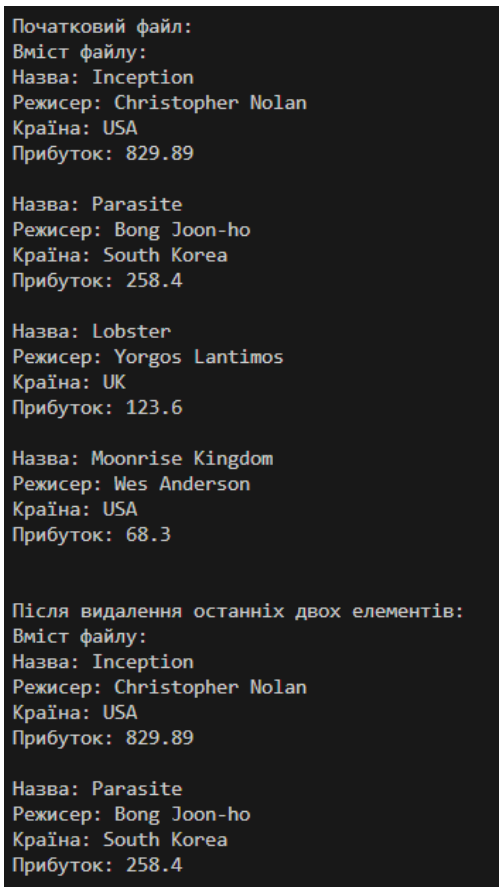
2) Результати виконання завдань, тестування та фактично затрачений час

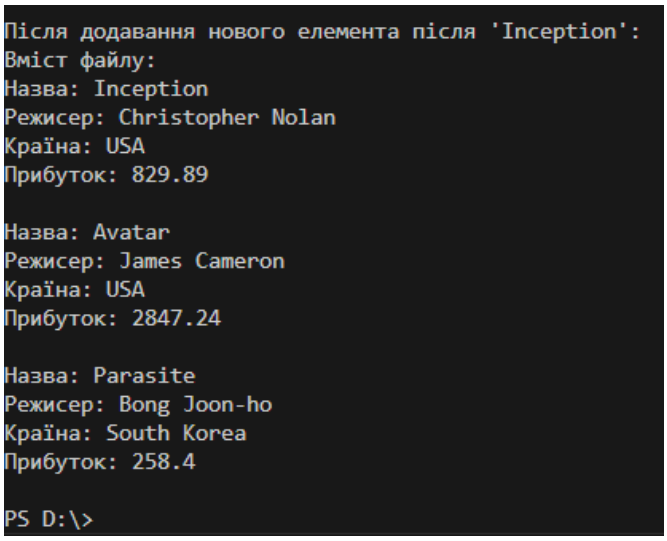
Завдання №1

```
Enter the text: duck inch_while float
Identifiers in the text: duck inch_while
PS D:\>
```

Витратив 40 хвилин.

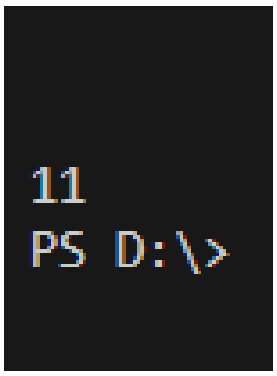
Завдання №2





Витратив на завдання близько 1 години.

Завдання №3



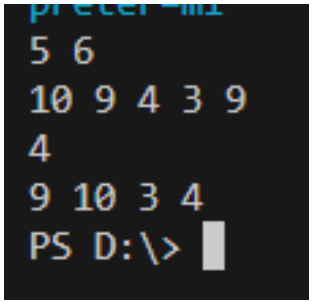
На це завдання пішло 25 хвилин.

Завдання №4

```
Введіть ім'я файлу для запису: pract1.txt
Введіть текст для запису у файл: dear diary, i cannot tell how much grie
f and hopelesness i feel
Файл успішно створено і записано.
Введіть ім'я файлу-джерела для копіювання: pract1.txt
Введіть ім'я файлу-призначення: pract2.txt
Введіть ім'я файлу-джерела для копіювання: pract1.txt
Введіть ім'я файлу-призначення: pract2.txt
Файл успішно скопійовано.
PS D:\>
```

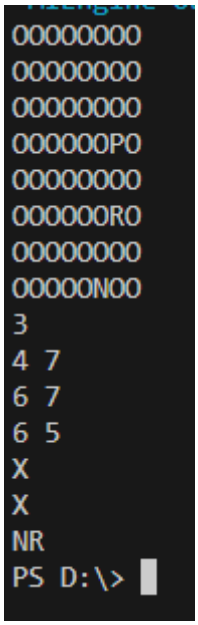
Витратив на завдання приблизно 60 хвилин.

Завдання №5



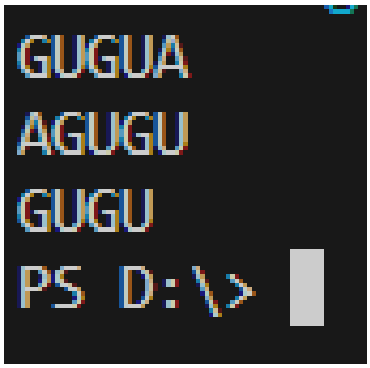
Опрацьовував завдання 2 години.

Завдання №6



На завдання пішло 5 годин.

Завдання №7



На це завдання пішло 40 хвилин.

Висновки:

У ході виконання лабораторної роботи ми ознайомилися з основами роботи з файлами у програмуванні мовою C++. Було розглянуто особливості використання текстових та бінарних файлів, а також роботи з символами й рядковими змінними. Ми навчилися використовувати можливості стандартної бібліотеки

C++ для маніпуляції файлами, зокрема методи відкриття, читання, запису та закриття файлів. Практично застосували принципи обробки даних у текстовому та бінарному форматах.

Крім того, у процесі роботи було розроблено власну бібліотеку, що дозволило отримати досвід її створення, структуризації та інтеграції у програмні проекти. Це сприяє підвищенню модульності та повторного використання коду.

Таким чином, мету лабораторної роботи досягнуто. Отримані знання та навички є корисними для створення ефективних і масштабованих програмних продуктів, які працюють із файлами та використовують власні бібліотеки.