Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 5**

***з дисципліни:*** «Мови та парадигми програмування»

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

***Виконав:***

Студент групи ШІ-13

Мельников Дмитро

# **Тема роботи:**

Робота із двійковими файлами. Ввід-вивід рядків.

# **Мета роботи:**

Ознайомитись з організацією вводу-виводу структурованої інформації із зовнішніх носіїв. Навчитися працювати з текстовими файлами, ввід-вивід текстової інформації і її зберігання на зовнішніх носіях.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Потоковий ввід-вивід
* Тема №2: Бінарні файли
* Тема №3: Робота з файлами

1. Індивідуальний план опрацювання теорії:

* Тема №1: Потоковий ввід-вивід.
  + Джерела Інформації
    - <http://cpp.dp.ua/potokove-vvedennya-vyvedennya/>
    - <https://acode.com.ua/urok-215-potoky-vvodu-i-vyvodu/>
  + Що опрацьовано:
    - Ввід/вивід в C++
    - Стандартні потоки С++
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12.23
  + Звершення опрацювання теми: 11.12.23
* Тема №2: Блоковий ввід-вивід
  + Джерела Інформації:
  + https://purecodecpp.com/uk/archives/2751Що опрацьовано:
    - Функції, що використовуються для блочного вводу та виводу
  + Статус: Ознайомлений
  + Початок опрацювання теми: 05.12.23
  + Звершення опрацювання теми: 10.11.23

* Тема №3: Робота з файлами
  + Джерела Інформації:
    - http://www.kytok.org.ua/post/vector-u-cplusplus
    - https://www.bestprog.net/uk/2021/10/12/c-the-vector-class-methods-that-define-and-modify-the-general-characteristics-of-array-ua/
    - https://acode.com.ua/urok-99-vvedennya-v-std-vector/
  + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.23
  + Звершення опрацювання теми: 12.12.23

**Виконання роботи:**

## **1. Опрацювання завдання та вимог до програм та середовища:**

**Завдання №1 Опрацювання теорії**

* Деталі завдання:

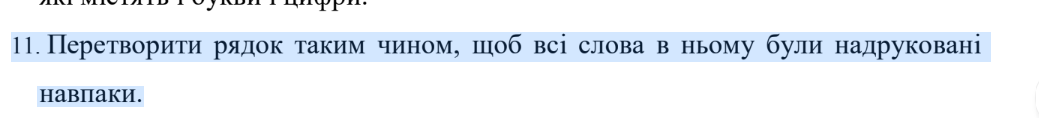
Опрацювання різноманітних матеріалів, аналіз та дослідження відео, статей, книг на задані теми. Організація та структурування отриманих даних для можливості ефективніше засвоювати отримані знання на практиці. Вивчення найважливіших моментів.

**Завдання №2 Перегляд вимог та проектування**

* Деталі завдання

Проектування блок-схем в Draw.io.

**Завдання №3 VNS Lab 6**

* Варіант завдання: 11
* Деталі 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 №4 VNS Lab 8**

* Варіант завдання: 11
* Деталі завдання:
* **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 "Відеокасета":**

**- назва фільму;**

**- режисер;**

**- тривалість;**

**- ціна.**

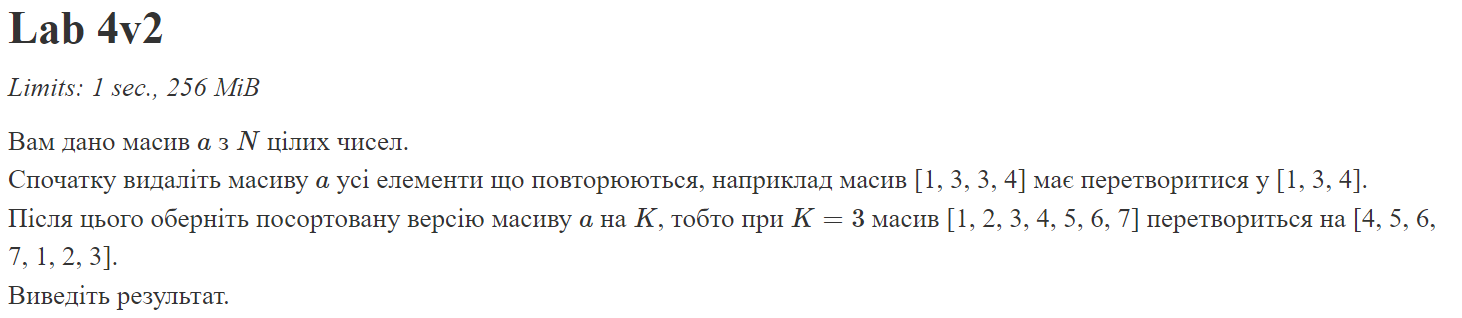
**Завдання №5 VNS Lab 9**

1) **Скопіювати з файлу F1 у файл F2.** **всі рядки, крім того рядка, що містить**

**найкоротше слово.**

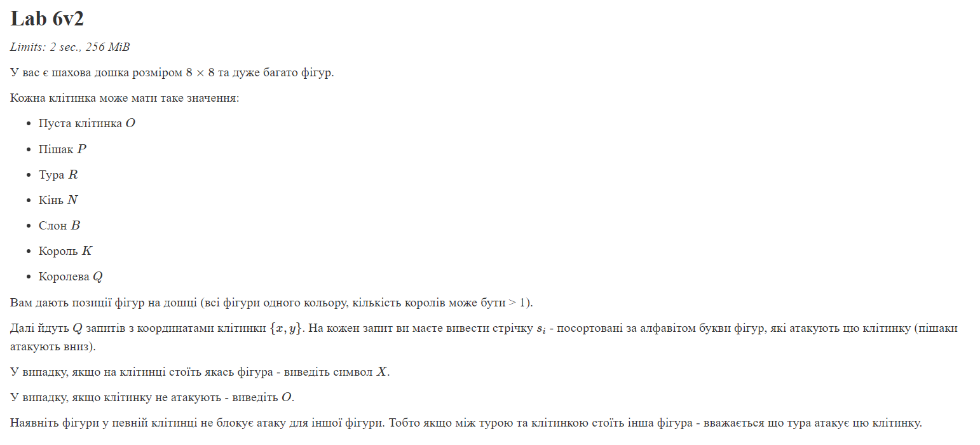
**2) Надрукувати номер цього рядка.**

**Завдання №6 Algotester Lab 4**

* **Варіант завдання: 2**
* ****

**Завдання №7 Algotester Lab 6**

* **Варіант завдання: 2**



**Завдання №8 Class Practice Task**

* Деталі завдання :

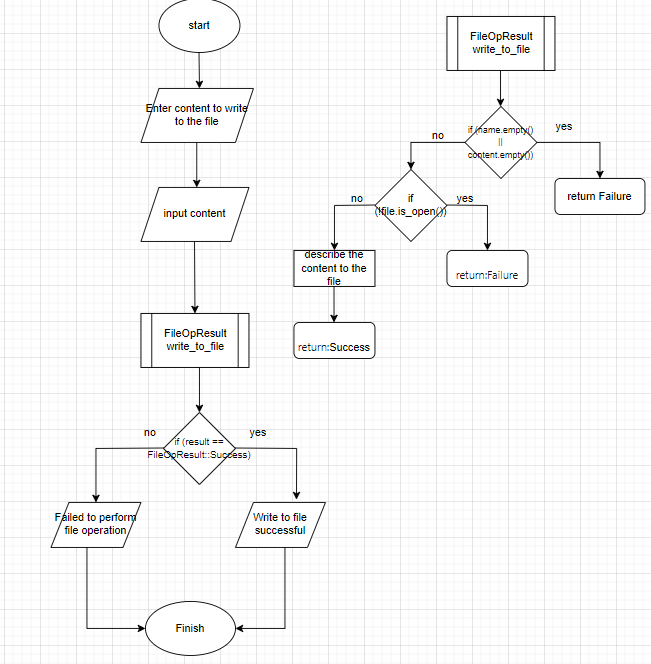
**1) Реалізувати функцію створення файла і запису в нього даних**

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

## **2. Дизайн та планована оцінка часу виконання завдань:**

**Програма №1 VNS Lab 8**

* Блок-схема:



* Планований час на реалізацію: 2 години

**Програма №2 VNS Lab 9**

* **Планований час на реалізацію: 3 години**

**Програма №3 Algotester Lab 4**

* **Планований час на реалізацію: 3 години**

**Програма №4 VNS Lab 6**

* **Планований час на реалізацію: 5 годин**

**Програма №5 Algotester Lab 6**

* **Планований час на реалізацію: 12 годин**

**- Важливі деталі для врахування в імплементації:**

**Програма №6 Class Practice Task**

* Планований час на реалізацію: 3 години

## **3. Код програм з посиланням на зовнішні ресурси:**

**Програма №6 Class Practice Task**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

enum class fileresult { Success, Failure }; // клас для встановлення саксес або фейл

fileresult text\_to\_file(const string& name, const string& content) { //функція для вмісту

    if (name.empty() || content.empty()) {

        return fileresult::Failure;

    }

    ofstream file(name, ios::out); //якщо файл не відкр то помилка

    if (!file.is\_open()) {

        return fileresult::Failure;

    }

    file << content; //виводить зміст у файл

    return fileresult::Success;

}

int main() {

    string filename = "classpractice.txt";

    string content;

    cout << "Enter content to write to the file: ";

    getline(cin, content);

    fileresult result = text\_to\_file(filename, content);

    if (result == fileresult::Success) {

        cout << "Write to file successful." << endl;

    } else {

        cout << "Failed to perform file operation." << endl;

    }

    return 0;

}

**Завдання №2 VNS Lab 9**

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <limits>

using namespace std;

int shortestword(const vector<string>& lines) {//номер рядка з найкоротшим словом

    int shortestRow = -1;

    size\_t shortestlength = numeric\_limits<size\_t>::max(); //початкове значення перед найменшим

    for (int i = 0; i < lines.size(); ++i) {

        istringstream s(lines[i]);

        string word;

        while (s >> word) {

            if (word.length() < shortestlength) { //  довжинa кожного слова та знаходимо найкоротше

                shortestlength = word.length();

                shortestRow = i;

            }

        }

    }

    return shortestRow;

}

void copylines(const string& inputFile, const string& outputFile) { //копіюємо рядки з F1  F2 крім shortrow

    ifstream in(inputFile);

    ofstream out(outputFile);

    if (!in.is\_open()) {

        cout << "Помилка відкриття файлу " << inputFile << endl;// для читання

        return;

    }

    if (!out.is\_open()) {

        cout << "Помилка відкриття файлу " << outputFile << " для запису." << endl;

        in.close();

        return;

    }

    vector<string> lines;

    string line;

    while (getline(in, line)) { //рядки з файлу F1

        lines.push\_back(line);

    }

    int shortestRow = shortestword(lines); // номер рядка з найкоротшим словом

    for (int i = 0; i < lines.size(); ++i) {// Копіюємо рядки F2

        if (i != shortestRow) {

            out << lines[i] << endl;

        }

    }

    cout << "Номер рядка з найкоротшим словом: " << shortestRow + 1 << endl;

    in.close();

    out.close();// закриває файлові потоки

}

int main() {

    //  функція для копіювання рядків та виведення номера рядка з найкоротшим словом

    copylines("F1.txt", "F2.txt");

    return 0;

}

**Завдання №3 Algotester Lab 4**

#include <iostream>

#include <vector>

#include <algorithm>

#include <set>

using namespace std;

int main() {

  int N, K;

  cin >> N >> K;

  // видалити з масиву a всі дублікати за допомогою set`.

  vector<int> a(N);

  for (int i = 0; i < N; ++i) {

    cin >> a[i];

  }

  set<int> uniqueSet(a.begin(), a.end());

  a.assign(uniqueSet.begin(), uniqueSet.end());

  sort(a.begin(), a.end());  // Сортувати масив

  // Обернути масив на кількість K

  swap\_ranges(a.begin(),a.begin(),a.end());

  rotate(a.begin(), a.begin() + K % a.size(), a.end());

  // Вивести результат

  cout << a.size() << endl;

  for (int i : a) { //вивід кожного елементу а

    cout << i << " ";

  }

  cout << endl;

  return 0;

}

**Завдання №4 VNS Lab 6**

#include <iostream>

#include <vector>

#include <bitset>

#include <algorithm>

using namespace std;

const int BOARD\_SIZE = 8;

enum Figure { EMPTY, PAWN, ROOK, KNIGHT, BISHOP, KING, QUEEN };

bitset<6> hasFigure[BOARD\_SIZE][BOARD\_SIZE];

bool canMoveInDirection(int x, int y, const string board[], int dx, int dy, char figure) {

    // Функція перевіряє, чи може фігура рухатись у певному напрямку

    for (int i = x + dx, j = y + dy; i >= 0 && i < BOARD\_SIZE && j >= 0 && j < BOARD\_SIZE; i += dx, j += dy) {

        if (board[i][j] == figure) {

            return true;

        }

    }

    return false;

}

bool canMoveInDirections(int x, int y, const string board[], int dx[], int dy[], int n, char figure) {

    // Функція перевіряє, чи може фігура рухатись у будь-якому з декількох напрямків

    for (int i = 0; i < n; ++i) {

        int nx = x + dx[i];

        int ny = y + dy[i];

        if (nx >= 0 && nx < BOARD\_SIZE && ny >= 0 && ny < BOARD\_SIZE && board[nx][ny] == figure) {

            return true;

        }

    }

    return false;

}

bool canMovePawn(int x, int y, const string board[]) {

    // Функція перевіряє, чи може пішак рухатись

    if ((x - 1 >= 0 && y - 1 >= 0 && board[x - 1][y - 1] == 'P') || (x - 1 >= 0 && y + 1 < BOARD\_SIZE && board[x - 1][y + 1] == 'P')) {

        return true;

    }

    return false;

}

bool canMoveRook(int x, int y, const string board[]) {

    // Функція перевіряє, чи може тура рухатись

    return canMoveInDirection(x, y, board, -1, 0, 'R') ||

           canMoveInDirection(x, y, board, 1, 0, 'R') ||

           canMoveInDirection(x, y, board, 0, -1, 'R') ||

           canMoveInDirection(x, y, board, 0, 1, 'R');

}

bool canMoveKnight(int x, int y, const string board[]) {

    // Функція перевіряє, чи може конь рухатись

    int knight\_dx[] = {-2, -1, 1, 2, 2, 1, -1, -2};

    int knight\_dy[] = {1, 2, 2, 1, -1, -2, -2, -1};

    return canMoveInDirections(x, y, board, knight\_dx, knight\_dy, 8, 'N');

}

bool canMoveBishop(int x, int y, const string board[]) {

    // Функція перевіряє, чи може слон рухатись

    return canMoveInDirection(x, y, board, -1, -1, 'B') ||

           canMoveInDirection(x, y, board, -1, 1, 'B') ||

           canMoveInDirection(x, y, board, 1, -1, 'B') ||

           canMoveInDirection(x, y, board, 1, 1, 'B');

}

bool canMoveKing(int x, int y, const string board[]) {

    // Функція перевіряє, чи може король рухатись

    int king\_dx[] = {-1, -1, -1, 0, 0, 1, 1, 1};

    int king\_dy[] = {-1, 0, 1, -1, 1, -1, 0, 1};

    return canMoveInDirections(x, y, board, king\_dx, king\_dy, 8, 'K');

}

bool canMoveQueen(int x, int y, const string board[]) {

    // Функція перевіряє, чи може ферзь рухатись

    return canMoveInDirection(x, y, board, -1, -1, 'Q') ||

           canMoveInDirection(x, y, board, -1, 1, 'Q') ||

           canMoveInDirection(x, y, board, 1, -1, 'Q') ||

           canMoveInDirection(x, y, board, 1, 1, 'Q') ||

           canMoveInDirection(x, y, board, -1, 0, 'Q') ||

           canMoveInDirection(x, y, board, 1, 0, 'Q') ||

           canMoveInDirection(x, y, board, 0, -1, 'Q') ||

           canMoveInDirection(x, y, board, 0, 1, 'Q');

}

string processQuery(int x, int y, const string board[]) {

    // Функція обробляє запит та повертає відповідь

    char figure = board[x][y];

    if (figure != 'O') {

        return "X";

    } else {

        string answer;

        const string figures = "BKNPQR"; // Змінено порядок фігур для зручності

        for (char figure : figures) {

            switch (figure) {

                case 'B':

                    if (canMoveBishop(x, y, board)) answer += figure;

                    break;

                case 'K':

                    if (canMoveKing(x, y, board)) answer += figure;

                    break;

                case 'N':

                    if (canMoveKnight(x, y, board)) answer += figure;

                    break;

                case 'P':

                    if (canMovePawn(x, y, board)) answer += figure;

                    break;

                case 'Q':

                    if (canMoveQueen(x, y, board)) answer += figure;

                    break;

                case 'R':

                    if (canMoveRook(x, y, board)) answer += figure;

                    break;

                default:

                    break;

            }

        }

        return answer.empty() ? "O" : answer;

    }

}

int main() {

    // Основна функція для вводу та обробки запитів

    string board[BOARD\_SIZE];

    for (int i = 0; i < BOARD\_SIZE; i++) {

        cin >> board[i];

        for (int j = 0; j < BOARD\_SIZE; j++) {

            hasFigure[i][j] = (board[i][j] != '.');

        }

    }

    int Q;

    cin >> Q;

    vector<pair<int, int>> queries;

    for (int m = 0; m < Q; m++) {

        int a, b;

        cin >> a >> b;

        queries.push\_back({a - 1, b - 1});

    }

    for (const auto &query : queries) {

        int x = query.first;

        int y = query.second;

        string answer = processQuery(x, y, board);

        cout << answer << endl;

    }

    return 0;

}

**Завдання №5 Algotester Lab 6**

#include <iostream>

#include <fstream>

#include <sstream>

#include <vector>

#include <limits>

using namespace std;

int shortestword(const vector<string>& lines) {//номер рядка з найкоротшим словом

    int shortestRow = -1;

    size\_t shortestlength = numeric\_limits<size\_t>::max(); //початкове значення перед найменшим

    for (int i = 0; i < lines.size(); ++i) {

        istringstream s(lines[i]);

        string word;

        while (s >> word) {

            if (word.length() < shortestlength) { //  довжинa кожного слова та знаходимо найкоротше

                shortestlength = word.length();

                shortestRow = i;

            }

        }

    }

    return shortestRow;

}

void copylines(const string& inputFile, const string& outputFile) { //копіюємо рядки з F1  F2 крім shortrow

    ifstream in(inputFile);

    ofstream out(outputFile);

    if (!in.is\_open()) {

        cout << "Помилка відкриття файлу " << inputFile << endl;// для читання

        return;

    }

    if (!out.is\_open()) {

        cout << "Помилка відкриття файлу " << outputFile << " для запису." << endl;

        in.close();

        return;

    }

    vector<string> lines;

    string line;

    while (getline(in, line)) { //рядки з файлу F1

        lines.push\_back(line);

    }

    int shortestRow = shortestword(lines); // номер рядка з найкоротшим словом

    for (int i = 0; i < lines.size(); ++i) {// Копіюємо рядки F2

        if (i != shortestRow) {

            out << lines[i] << endl;

        }

    }

    cout << "Номер рядка з найкоротшим словом: " << shortestRow + 1 << endl;

    in.close();

    out.close();// закриває файлові потоки

}

int main() {

    //  функція для копіювання рядків та виведення номера рядка з найкоротшим словом

    copylines("F1.txt", "F2.txt");

    return 0;

}

**Програма №1 VNS Lab 8**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

struct Video { //Структура

    string title;

    string director;

    int duration;

    float price;

};

void createFile(const char\* filename, Video\* data, int size) { //створюємо файл

    ofstream file(filename, ios::binary);//створюємо

    if (!file) { //якщо нема змоги відкрити файл помилка

        cerr << "Помилка: не вдалося відкрити файл для запису." << endl; //вивід помилки

        return;

    }

    file.write(reinterpret\_cast<char\*>(data), size \* sizeof(Video));//записуємо файл та переводити масив структ у чар

    file.close();//закриваємо файл задля збереження файлів

}

void printFile(const char\* filename, int size) {// друк вмісту файлу

    ifstream file(filename, ios::binary);// записуємо в файл

    if (!file) {

        cerr << "Помилка: не вдалося відкрити файл для читання." << endl;

        return;

    }

    Video\* data = new Video[size];

    file.read(reinterpret\_cast<char\*>(data), size \* sizeof(Video));

    file.close();

    for (int i = 0; i < size; ++i) {

        cout << "Element " << i + 1 << ":\n";

        cout << "Name of Movie: " << data[i].title << endl;

        cout << "Producer: " << data[i].director << endl;

        cout << "Lenght: " << data[i].duration << " minuts\n";

        cout << "Price: " << data[i].price << " grn\n\n";

    }

    delete[] data;

}

//додавання елементів в кінець файлу

void addElements(const char\* filename, Video\* newData, int newSize) {

    ofstream file(filename, ios::binary | ios::app);

    if (!file) {

        cerr << "Помилка: не вдалося відкрити файл для запису." << endl;

        return;

    }

    file.write(reinterpret\_cast<char\*>(newData), newSize \* sizeof(Video));

    file.close();

}

//знищення елементів з ціною вищою заданої

void deleteElements(const char\* filename, float maxPrice) {

    ifstream file(filename, ios::binary);

    if (!file) {

        cerr << "Помилка: не вдалося відкрити файл для читання." << endl;

        return;

    }

    ofstream tempFile("temp.txt", ios::binary);

    if (!tempFile) {

        cerr << "Помилка: не вдалося відкрити тимчасовий файл для запису." << endl;

        file.close();

        return;

    }

    Video temp;

    while (file.read(reinterpret\_cast<char\*>(&temp), sizeof(Video))) {

        if (temp.price <= maxPrice) { // перевіряє чи ціна не перевищує вказану

            tempFile.write(reinterpret\_cast<char\*>(&temp), sizeof(Video)); //додає до тимчасового файлу

        }

    }

    file.close();

    tempFile.close();

    remove(filename);//видаляє орігінальний файл

    rename("temp.txt", filename); //перейменовує тимчасовий

}

int main() {

    const char\* filename = "casetta.txt";

    const int initialSize = 3; // Початковий розмір файлу

    const float maxprice = 50.0; // Максимальна ціна для знищення елементів

    Video initialData[initialSize] = {

        {"Movie 1", "Producer 1", 120, 40.0},

        {"Movie 2", "Producer 2", 90, 35.0},

        {"Movie 3", "Producer 3", 110, 55.0}

    };

    createFile(filename, initialData, initialSize);//формування файлу з даними

    cout << "Початковий вміст файлу:\n";//друк вмісту файл

    printFile(filename, initialSize);

    deleteElements(filename, maxprice);// видалення з ціною вищою

    cout << "\nВміст файлу після знищення елементів з ціною вищою за " << maxprice << ":\n";//  вміст файлу після видалення

    printFile(filename, initialSize);

    // Додавання 3 елементів в кінець файлу

    const int newSize = 3;

    Video newElements[newSize] = {

        {"Movie 4", "Producer 4", 100, 45.0},

        {"Movie 5", "Producer 5", 80, 30.0},

        {"Movie 6", "Producer 6", 95, 50.0}

    };

    addElements(filename, newElements, newSize);

    cout << "\nВміст файлу після додавання 3 елементів:\n";  //  вміст файлу після додавання

    printFile(filename, initialSize + newSize);

    return 0;

}

## **4. Результати виконання завдань, тестування та фактично затрачений час:**

**Завдання №1** **VNS Lab 9**

   Номер рядка з найкоротшим словом: 3

**Завдання №3 Algotester Lab 4**



**Завдання №4 VNS Lab 6**

      ведіть рядок: hello world

Результат: world hello

PS J:\Epic 5>

**Завдання №6 Class Practice Task**

      Enter content to write to the file: hello im Dmytro

Write to file successful.

PS J:\Epic 5>

**Завдання №7 Algotester lab6**

****

# **Висновки:**

В ході виконання лабораторних та практичних завдань, що стосувалися взаємодії з двійковими файлами та операцій введення-виведення рядків, мені вдалося успішно ознайомитися з організацією обміну структурованою інформацією через зовнішні носії. Робота над цими етапами дозволила мені отримати практичні навички роботи з текстовими файлами, виконанням операцій введення-виведення текстової інформації, а також зберіганням даних на зовнішніх носіях. Цей досвід виявився надзвичайно корисним, допомагаючи зрозуміти процес зберігання та обробки інформації в різних форматах, зокрема, вивчити особливості роботи з двійковими файлами та текстовими рядками.