Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**Звіт**

**про виконання лабораторних та практичних робіт блоку № 5**

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

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

до:

ВНС Лабораторної Роботи № 6

ВНС Лабораторної Роботи № 8

ВНС Лабораторної Роботи № 9

Алготестер Лабораторної Роботи №4

Алготестер Лабораторної Роботи №6

Практичних Робіт №5

**Виконав:**

Студент групи ШІ-12

Заставний Денис Миколайович

# **Тема роботи:**

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

# **Мета роботи:**

Ознайомитись з методами роботи з текстовими та бінарними файлами. Ознайомитись з створенням та використання бібліотек.

**Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Бінарні та текстові файли.
* Тема №2: Символи і рядкові змінні
* Тема №3: Стандартна бібліотека.створенням та використання бібліотек.
* Індивідуальний план опрацювання теорії:
* Тема №1: Бінарні та текстові файли.
  + Джерела Інформації.
    - [C++ Теорія ⦁ Урок 166 ⦁ Робота з файлами (стиль мови C)](https://www.youtube.com/watch?v=FeNqHytI0fA)
    - <https://purecodecpp.com/uk/archives/2751>
    - <https://replace.org.ua/topic/7901/>
  + Що опрацьовано:
    - Опрацьовано види файлів, їх особливості.
    - Статус: Ознайомлений
  + Початок опрацювання теми: 01.12.2023
  + Звершення опрацювання теми: 07.12.2023
* Тема №2: Символи і рядкові змінні
  + Джерела Інформації:
    - <http://www.kievoit.ippo.kubg.edu.ua/kievoit/2016/73_C++/index.html>
    - <http://cpp.dp.ua/vykorystannya-ryadkiv-typu-string/>
    - <https://acode.com.ua/urok-38-symvolnyj-typ-danyh-char/>
  + Що опрацьовано:
    - Опрацьовано поняття символів і рядкових зміних, корисні методи рядкових змінних у С++.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.11.2023
  + Звершення опрацювання теми: 07.12.2023
* Тема №3: Стандартна бібліотека. Cтворенням та використання бібліотек.
  + Джерела Інформації:
    - <https://acode.com.ua/urok-101-algorytmy-v-standartnij-bibliotetsi-s/>
    - <http://www.kievoit.ippo.kubg.edu.ua/kievoit/2016/66_C++/index.html>
  + Що опрацьовано:
    - Опрацьовано складові частини стандартної бібліотеки – конейнери(вектори, списки, сети та мапи), алгоритми(сортування та пошуку), функції, ітератори.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.11.2023
  + Звершення опрацювання теми: 07.12.2023

# **Виконання роботи:**

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

Завдання №1 **VNS Lab 6 - Task 1-17V**

* Варіант завдання 17
* 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. Всі слова рядка, які починаються з букви, відсортувати за абеткою.

Завдання №2 **VNS Lab 8 - Task 1-17V**

* Варіант завдання 17
* 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 "Фільм":

- назва;

- режисер;

- рік випуску;

- вартість.

Знищити всі елементи, у яких вартість перевищує задану, додати елемент на

початок файлу.

Завдання №3 **VNS Lab 9 - Task 1-17**

* Варіант завдання 1
* Деталі завдання:

Створити текстовий файл F1 не менше, ніж з 10 рядків і записати в нього

інформацію

Виконати завдання. 1) Скопіювати з файлу F1 у файл F2 всі рядки, що починаються на букву «А»

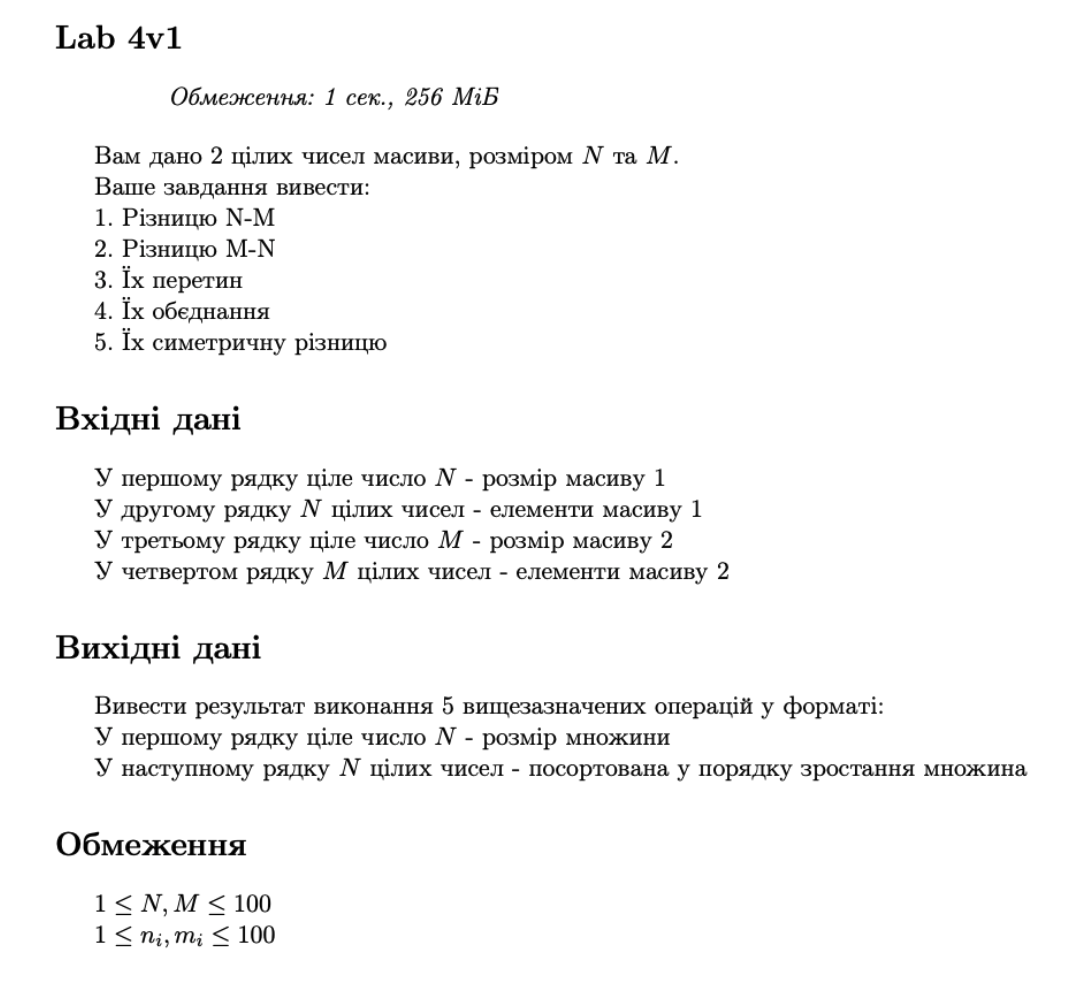
розташовані між рядками з номерами N1 й N2, а потім всі рядки від N2+3 і до

останнього.

2) Визначити кількість слів в останньому рядку файлу F2.

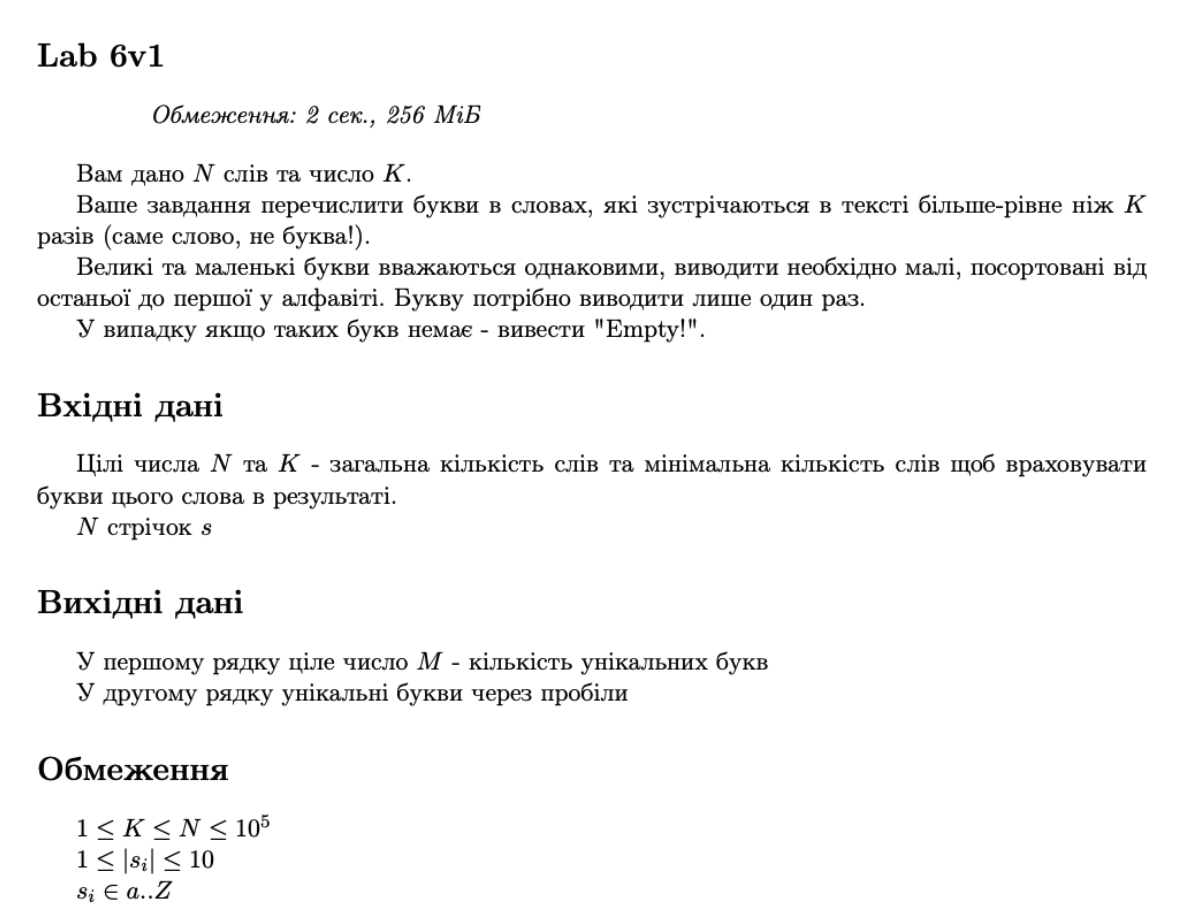
Завдання №4 **Algotester Lab 4v1**

Варіант завданння 1

* Деталі завдання
* 

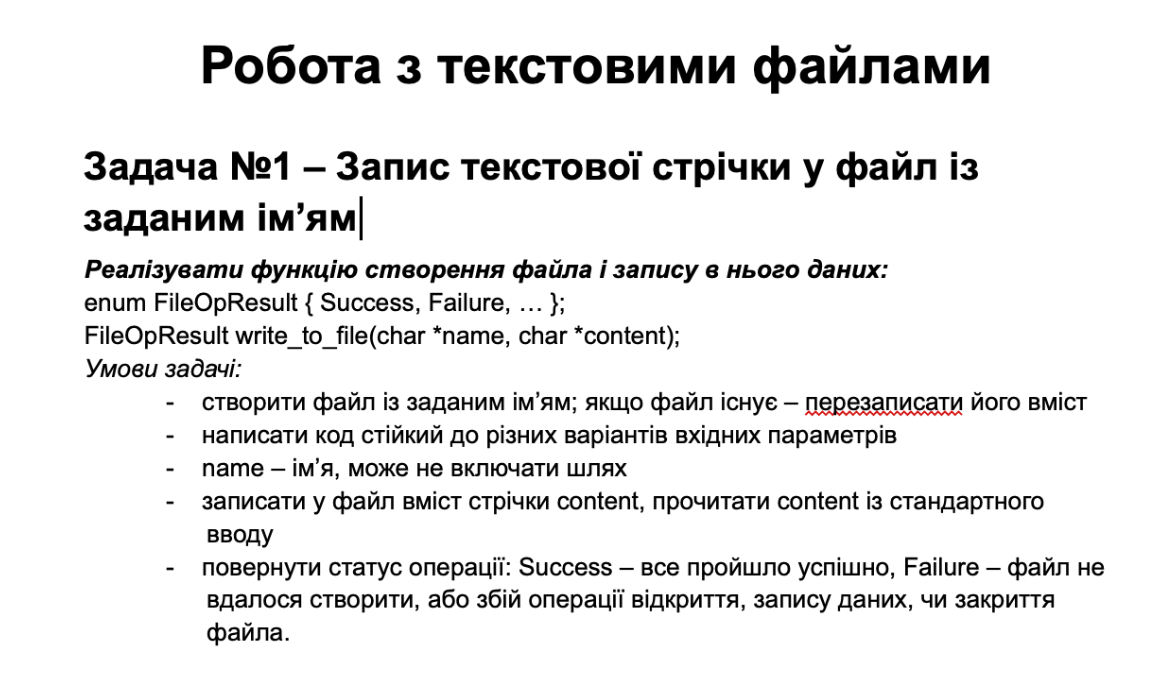
*Figure 1 Умова до завдання №4*

Завдання №5 **Algotester Lab 6v1**

* Деталі завдання
* 

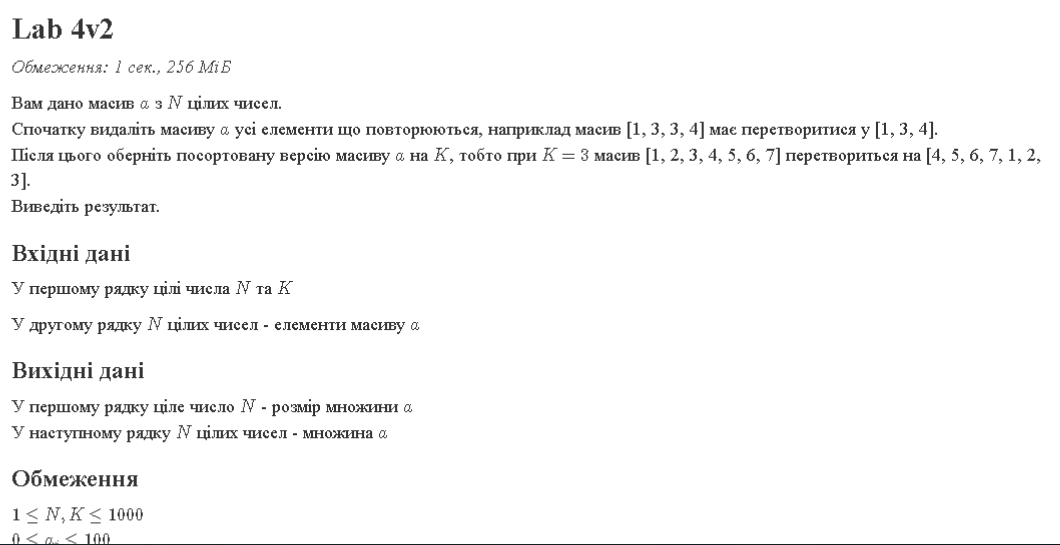
*Figure 2 Умова до завдання №5*

Завдання №6 **Class Practice Work**

* Деталі завдання
* 

*Figure 3 Умова до завдання №6*

# **Завдання №7 Self Practice Work – Algotester Lab 4v2**

* Деталі завдання
* 

*Figure 4 Умова до завдання №7*

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

Програма №1 **VNS Lab 6 - Task 1-17**

* Планований час на реалізацію: 30хв
* Важливі деталі для врахування в імплементації: необхідно посортувати незалежно від регістру першої букви, тому пишемо власну функцію сортування.

Програма №2 **VNS VNS Lab 8 - Task 1-17**

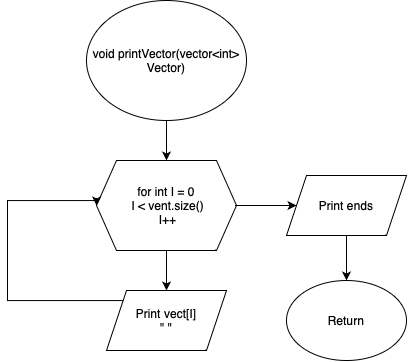
* Планований час на реалізацію: 120хв.

Програма №3 **VNS Lab 9 - Task 1-17**

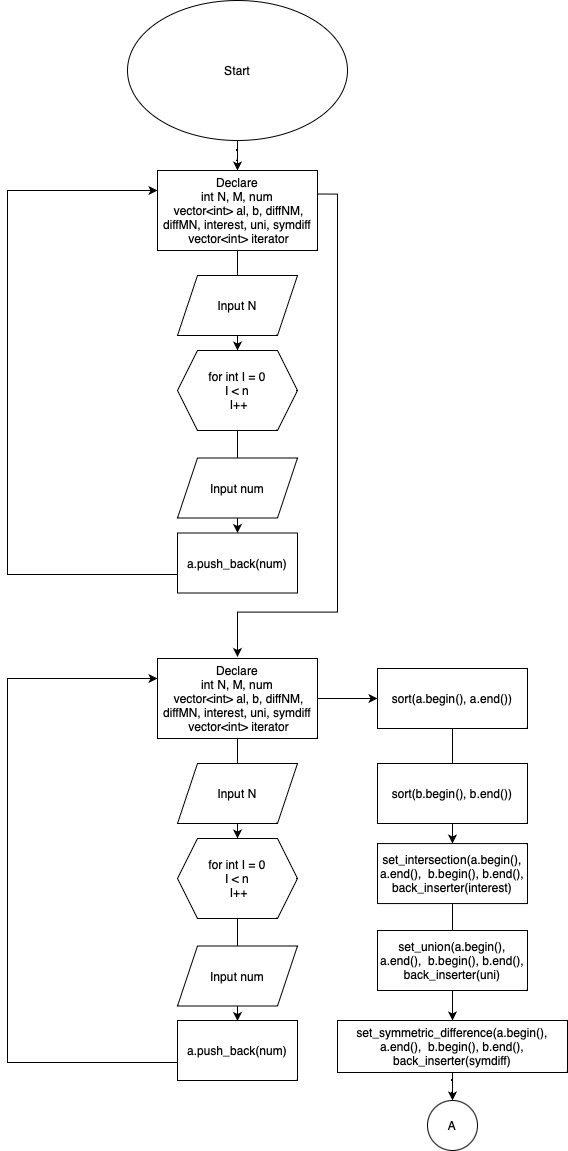
Планований час на реалізацію: 60хв.

Програма №4 **Algotester Lab 4v1**

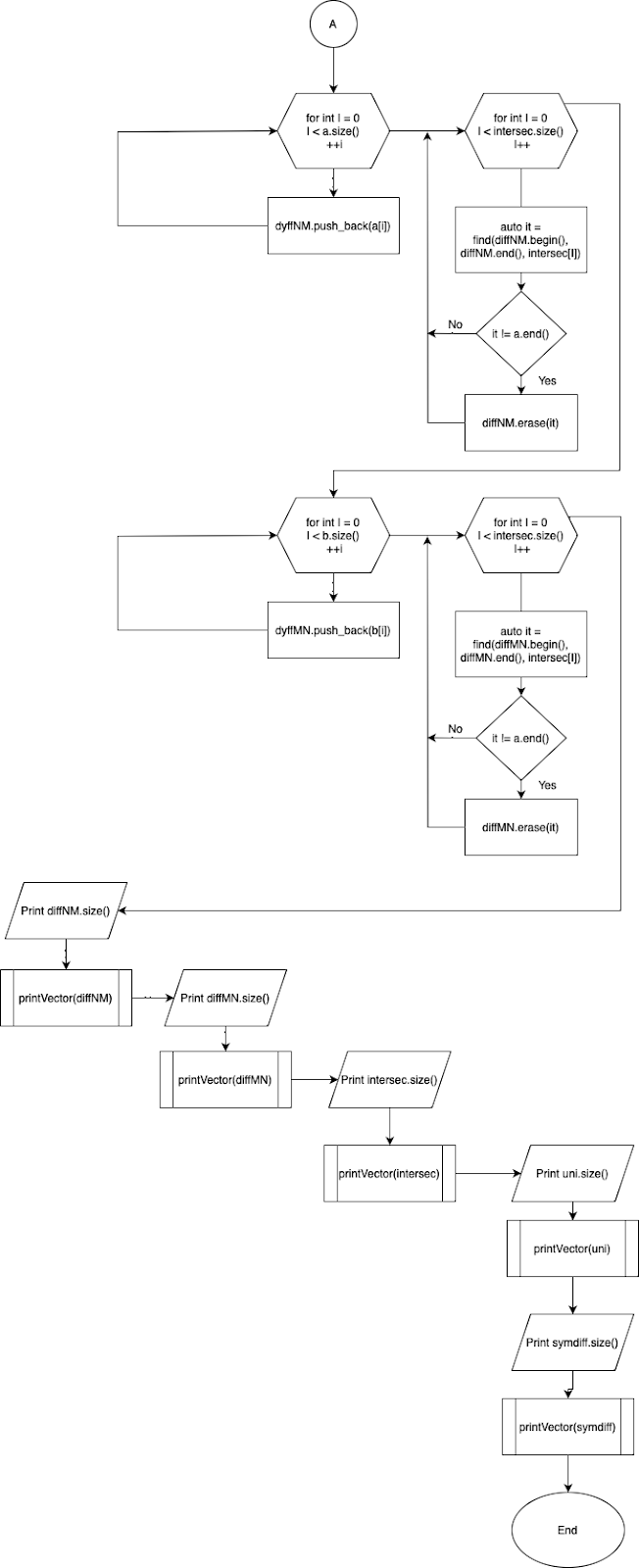
**Блок-схема:**

****

*Figure 5 Блок-схема до програми №4 - функція printVector*



*Figure 6 Блок-схема до програми №4 – функція main 1*



* Планований час на реалізацію 20 хв.

Програма №5 **Algotester Lab 6**

* Планований час на реалізацію: 30хв.

Програма №6 **Class Practice Work**

* Планований час на реалізацію: 40хв.

Програма №7 **Self Practice Work – Algotester Lab 4v2**

* Планований час на реалізацію: 40хв.

**3. Конфігурація середовища до виконання завдань:**

відсутня

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

Завдання №1 **VNS Lab 6 - Task 1-17v**

#include <iostream>

#include <cstring>

#include <algorithm>

int compareStrings(const void \*a, const void \*b) {

return strcmp(\*(const char \*\*)a, \*(const char \*\*)b);

}

int main() {

using namespace std;

char input[256];

char \*words[256];

char \*token;

cout << "Enter a string: \n";

cin.getline(input, sizeof(input));

int wordCount = 0;

token = strtok(input, " ");

while (token != NULL) {

words[wordCount] = new char[strlen(token) + 1];

strcpy(words[wordCount], token);

wordCount++;

token = strtok(NULL, " ");

}

qsort(words, wordCount, sizeof(char \*), compareStrings);

cout << "Sorted words: ";

for (int i = 0; i < wordCount; i++) {

if (isalpha(words[i][0])) {

cout << words[i] << " ";

}

delete[] words[i];

}

return 0;

}

Завдання №2 **VNS Lab 8 - Task 1-17v**

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

using namespace std;

struct Film {

string title;

string director;

int year;

double cost;

};

void createBinaryFile(const char \*fileName, vector<Film> &films) {

ofstream file(fileName, ios::binary | ios::trunc);

if (!file.is\_open()) {

cerr << "Помилка при відкритті файлу для запису." << endl;

return;

}

file.write(reinterpret\_cast<char \*>(films.data()), sizeof(Film) \* films.size());

file.close();

}

void printBinaryFile(const char \*fileName) {

ifstream file(fileName, ios::binary);

if (!file.is\_open()) {

cerr << "Помилка при відкритті файлу для читання." << endl;

return;

}

Film film;

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

cout << "Title: " << film.title << ", Director: " << film.director

<< ", Year: " << film.year << ", Cost: " << film.cost << endl;

}

file.close();

}

void deleteFilmsByCost(const char \*fileName, double maxCost) {

ifstream inFile(fileName, ios::binary);

if (!inFile.is\_open()) {

cerr << "Помилка при відкритті файлу для читання." << endl;

return;

}

ofstream outFile("temp.dat", ios::binary | ios::trunc);

if (!outFile.is\_open()) {

cerr << "Помилка при відкритті файлу для запису." << endl;

inFile.close();

return;

}

Film film;

while (inFile.read(reinterpret\_cast<char \*>(&film), sizeof(Film))) {

if (film.cost <= maxCost) {

outFile.write(reinterpret\_cast<char \*>(&film), sizeof(Film));

}

}

inFile.close();

outFile.close();

remove(fileName);

rename("temp.dat", fileName);

}

void addFilm(const char \*fileName, Film newFilm) {

ifstream inFile(fileName, ios::binary);

if (!inFile.is\_open()) {

cerr << "Помилка при відкритті файлу для читання." << endl;

return;

}

ofstream outFile("temp.dat", ios::binary | ios::trunc);

if (!outFile.is\_open()) {

cerr << "Помилка при відкритті файлу для запису." << endl;

inFile.close();

return;

}

outFile.write(reinterpret\_cast<char \*>(&newFilm), sizeof(Film));

Film film;

while (inFile.read(reinterpret\_cast<char \*>(&film), sizeof(Film))) {

outFile.write(reinterpret\_cast<char \*>(&film), sizeof(Film));

}

inFile.close();

outFile.close();

remove(fileName);

rename("temp.dat", fileName);

}

int main() {

vector<Film> films = {

{"Inception", "Christopher Nolan", 2010, 25.99},

{"The Shawshank Redemption", "Frank Darabont", 1994, 20.50},

{"Forrest Gump", "Robert Zemeckis", 1994, 15.75}

};

const char \*fileName = "films.bin";

createBinaryFile(fileName, films);

cout << "Contents of the binary file:" << endl;

printBinaryFile(fileName);

double maxCostToDelete = 15.0;

cout << "\nDeleting films with cost more than " << maxCostToDelete << endl;

deleteFilmsByCost(fileName, maxCostToDelete);

cout << "\nContents of the binary file after deletion:" << endl;

printBinaryFile(fileName);

Film newFilm = {"Titanic", "James Cameron", 1997, 18.25};

cout << "\nAdding a new film:" << endl;

addFilm(fileName, newFilm);

cout << "\nContents of the binary file after addition:" << endl;

printBinaryFile(fileName);

return 0;

}

Завдання №3 **VNS Lab 9**

#include <iostream>

#include <fstream>

#include <sstream>

#include <string>

using namespace std;

// Функція для копіювання рядків з файлу F1 у файл F2

void copyLines(const char \*inputFileName, const char \*outputFileName, int start1, int end1, int start2) {

ifstream inputFile(inputFileName);

if (!inputFile.is\_open()) {

cerr << "Помилка при відкритті файлу для читання." << endl;

return;

}

ofstream outputFile(outputFileName);

if (!outputFile.is\_open()) {

cerr << "Помилка при відкритті файлу для запису." << endl;

inputFile.close();

return;

}

string line;

int currentLine = 0;

// Копіюємо рядки, які починаються на "А" між n1 та n2

while (getline(inputFile, line)) {

currentLine++;

if (currentLine >= start1 && currentLine <= end1 && !line.empty() && line[0] == 'A') {

outputFile << line << endl;

}

}

inputFile.clear();

inputFile.seekg(0, ios::beg);

currentLine = 0;

while (currentLine < start2 && getline(inputFile, line)) {

currentLine++;

}

while (getline(inputFile, line)) {

if (!line.empty()) {

outputFile << line << endl;

}

}

inputFile.close();

outputFile.close();

}

int countWordsInLine(const string &line) {

int wordCount = 0;

istringstream iss(line);

string word;

while (iss >> word) {

wordCount++;

}

return wordCount;

}

int main() {

ofstream f1("F1.txt");

f1 << "Apple pie\n";

f1 << "Banana split with chocolate\n";

f1 << "Alphabet soup and crackers\n";

f1 << "Apricot jam on toast\n";

f1 << "Avocado and tomato toast\n";

f1 << "Aardvark goes on an exciting adventure\n";

f1 << "Artichoke dip with spinach\n";

f1 << "Cat takes a long nap under the sun\n";

f1 << "Doghouse in the backyard\n";

f1 << "Elephants march in a grand parade\n";

f1.close();

copyLines("F1.txt", "F2.txt", 1, 2, 5);

ifstream f2("F2.txt");

string lastLine;

while (getline(f2, lastLine)) {

if (!lastLine.empty()) {

}

}

f2.close();

int lastLineWordCount = countWordsInLine(lastLine);

cout << "Contents of F2.txt after copying:" << endl;

ifstream f2Print("F2.txt");

cout << f2Print.rdbuf() << endl;

f2Print.close();

cout << "\nWord count in the last line of F2.txt: " << lastLineWordCount << endl;

return 0;

}

Завдання №4 **Algotester Lab 4v1**

*#include <iostream>*

*#include <vector>*

*#include <algorithm>*

*using namespace std;*

*void printVector(vector<int> vect) {*

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

*cout << vect[i] << " ";*

*}*

*cout << endl;*

*}*

*int main() {*

*int N, M, num;*

*vector<int> a, b, diffNM, diffMN, intersec, uni, symdiff;*

*vector<int>::iterator it;*

*cin >> N;*

*for (int i = 0; i < N; i++) {*

*cin >> num;*

*a.push\_back(num);*

*}*

*cin >> M;*

*for (int i = 0; i < M; i++) {*

*cin >> num;*

*b.push\_back(num);*

*}*

*sort(a.begin(), a.end());*

*sort(b.begin(), b.end());*

*set\_intersection(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(intersec));*

*set\_union(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(uni));*

*set\_symmetric\_difference(a.begin(), a.end(), b.begin(), b.end(), back\_inserter(symdiff));*

*// Визначення різниці N-M*

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

*diffNM.push\_back(a[i]);*

*}*

*// Видалення спільних елементів*

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

*auto it = find(diffNM.begin(), diffNM.end(), intersec[i]);*

*if (it != a.end()) {*

*diffNM.erase(it);*

*}*

*}*

*// Визначення різниці M-N*

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

*diffMN.push\_back(b[i]);*

*}*

*// Видалення спільних елементів*

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

*auto it = find(diffMN.begin(), diffMN.end(), intersec[i]);*

*if (it != b.end()) {*

*diffMN.erase(it);*

*}*

*}*

*cout << diffNM.size() << endl;*

*printVector(diffNM);*

*cout << diffMN.size() << endl;*

*printVector(diffMN);*

*cout << intersec.size() << endl;*

*printVector(intersec);*

*cout << uni.size() << endl;*

*printVector(uni);*

*cout << symdiff.size() << endl;*

*printVector(symdiff);*

*return 0;*

*}*

Програма №5 **Algotester Lab6v1**

#include <algorithm>

#include <string>

#include <vector>

#include <iostream>

#include <unordered\_map>

using namespace std;

bool sortZA(char a, char b) {

return a > b;

}

int main() {

int wordCount, minWordCount, totalOccurrence = 0;

cin >> wordCount >> minWordCount;

string currentWord, result;

vector<string> words;

unordered\_map<string, int> wordOccurrences;

for (int i = 0; i < wordCount; i++) {

cin >> currentWord;

transform(currentWord.begin(), currentWord.end(), currentWord.begin(), ::tolower);

words.push\_back(currentWord);

wordOccurrences[currentWord]++;

}

for (int i = 0; i < wordCount; i++) {

int count = wordOccurrences[words[i]];

if (count >= minWordCount) {

totalOccurrence += count;

result += words[i];

}

}

sort(result.begin(), result.end());

auto last = unique(result.begin(), result.end());

result.erase(last, result.end());

sort(result.begin(), result.end(), sortZA);

if (totalOccurrence >= minWordCount) {

cout << result.size() << endl;

for (auto character : result) {

cout << character << " ";

}

} else {

cout << "Empty!";

}

return 0;

}

Програма №6 **Practise team task p.1**

*#include <iostream>*

*#include <fstream>*

*enum FileOpResult { Success, Failure };*

*FileOpResult write\_to\_file(const char \*name, const char \*content) {*

*std::ofstream file(name);*

*if (!file.is\_open()) {*

*std::cerr << "Error opening file for writing: " << name << std::endl;*

*return Failure;*

*}*

*file << content;*

*if (!file.good()) {*

*std::cerr << "Error writing to file: " << name << std::endl;*

*return Failure;*

*}*

*file.close();*

*return Success;*

*}*

*int main() {*

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

*const char \*content = "Hello, world!";*

*FileOpResult result = write\_to\_file(filename, content);*

*if (result == Success) {*

*std::cout << "File created and written successfully." << std::endl;*

*} else {*

*std::cerr << "Failed to create or write to file." << std::endl;*

*}*

*return 0;*

*}*

Програма №7 **Practise team task p.2**

**#include <iostream>**

**#include <fstream>**

**#include <string>**

**enum FileOpResult { Success, Failure };**

**FileOpResult write\_to\_file(const char \*name, const char \*content) {**

**std::ofstream file(name);**

**if (!file.is\_open()) {**

**std::cerr << "Error: Cannot create or open file." << std::endl;**

**return Failure;**

**}**

**file << content;**

**file.close();**

**return Success;**

**}**

**FileOpResult copy\_file(const char \*file\_from, const char \*file\_to) {**

**std::ifstream source\_file(file\_from, std::ios::binary);**

**if (!source\_file.is\_open()) {**

**std::cerr << "Error: Source file does not exist or cannot be opened." << std::endl;**

**return Failure;**

**}**

**std::ofstream destination\_file(file\_to, std::ios::binary);**

**if (!destination\_file.is\_open()) {**

**std::cerr << "Error: Cannot open or create destination file." << std::endl;**

**source\_file.close();**

**return Failure;**

**}**

**destination\_file << source\_file.rdbuf();**

**if (!destination\_file.good()) {**

**std::cerr << "Error: Failed to copy content between files." << std::endl;**

**source\_file.close();**

**destination\_file.close();**

**return Failure;**

**}**

**source\_file.close();**

**destination\_file.close();**

**return Success;**

**}**

**int main() {**

**const char \*source\_filename = "source.txt";**

**const char \*destination\_filename = "destination.txt";**

**const char \*content = "Hello, this is the content of the source file.";**

**FileOpResult write\_result = write\_to\_file(source\_filename, content);**

**if (write\_result != Success) {**

**std::cerr << "Failed to write content to the source file." << std::endl;**

**return 1;**

**}**

**FileOpResult copy\_result = copy\_file(source\_filename, destination\_filename);**

**if (copy\_result == Success) {**

**std::cout << "Files created and copied successfully." << std::endl;**

**} else {**

**std::cerr << "Failed to copy files." << std::endl;**

**}**

**return 0;**

**}**

Програма №8 **Selfpractice task Algotester lav 4v2**

**#include <iostream>**

**#include <vector>**

**#include <algorithm>**

**int main() {**

**using namespace std;**

**int n, k, b;**

**cin >> n >> k;**

**vector<int> uniqueSet;**

**for (int i = 0; i < n; i++) {**

**cin >> b;**

**uniqueSet.push\_back(b);**

**}**

**sort(uniqueSet.begin(), uniqueSet.end());**

**auto x = unique(uniqueSet.begin(), uniqueSet.end());**

**uniqueSet.erase(x, uniqueSet.end());**

**if (k > uniqueSet.size()) {**

**k = k % uniqueSet.size();**

**}**

**rotate(uniqueSet.begin(), uniqueSet.begin() + k % uniqueSet.size(), uniqueSet.end());**

**cout << uniqueSet.size() << endl;**

**for (int j = 0; j < uniqueSet.size(); j++) {**

**cout << uniqueSet[j] << " ";**

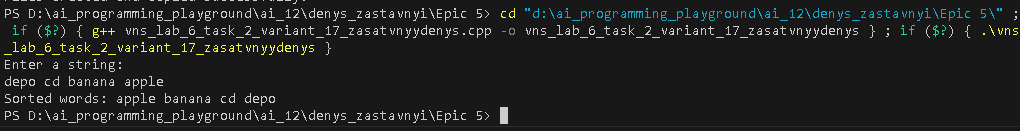
**}**

**return 0;**

**}**

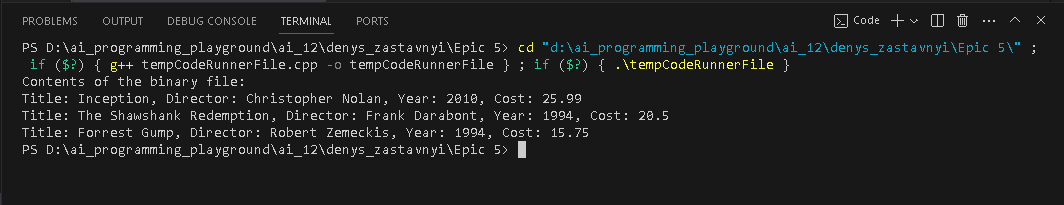
## **5. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 **VNS Lab 6 - Task 1-17**



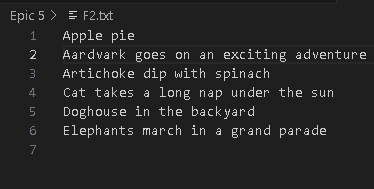
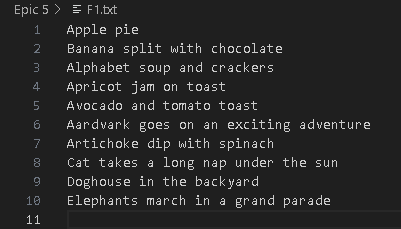
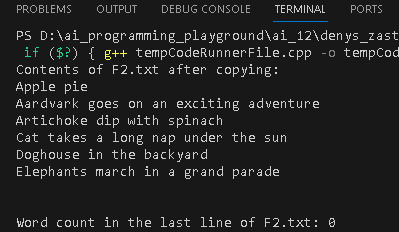
Час затрачений на виконання завдання: 30хв.

Завдання №2 **VNS Lab 8 - Task 1-17**



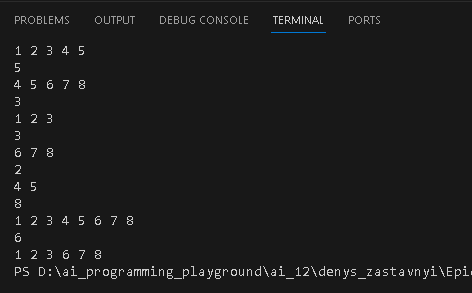
Час затрачений на виконання завдання: 80хв.

Завдання №3 **VNS Lab 9 - Task 1-17**

**

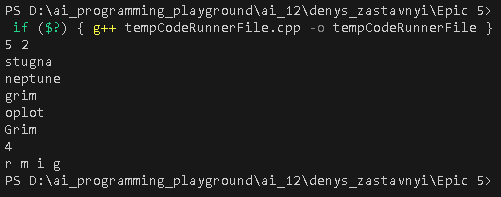
Час затрачений на виконання завдання: 60хв.

Завдання №4 **Algotester Lab 4**



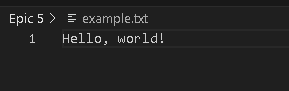
Час затрачений на виконання завдання: 20хв.

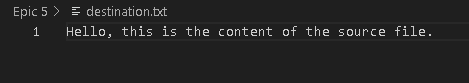
Програма №5 **Algotester Lab 6**



Час затрачений на виконання завдання: 60хв.

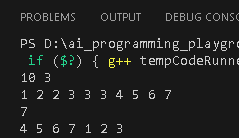
Програма №7,8 **Class Practice Work**





Час затрачений на виконання завдання: 45хв.

Програма №7 **Self Practice Work – Lab 4v2**



Час затрачений на виконання завдання: 40хв.

# **Висновки:**

Під час виконання епіку 5 я ознайомився з бінарними та текстовими файлами, символьними та рядковими змінними, стандартною бібліотекою та системами числення.