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. 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-11

Іванов Олексій Олександрович

# **Тема роботи:**

**Робота з файлами в мовах С та С++. Використання С-методів fopen, fwrite, fclose, та С++ бібліотеки fstream.**

# **Мета роботи:**

* Ознайомитися з основними поняттями та методами роботи з файлами в мові С++.
* Вивчити способи створення, відкриття, закриття, читання та запису даних у файли.
* Набути практичних навичок роботи з файлами в мові С++.

# **Виконання роботи:**

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

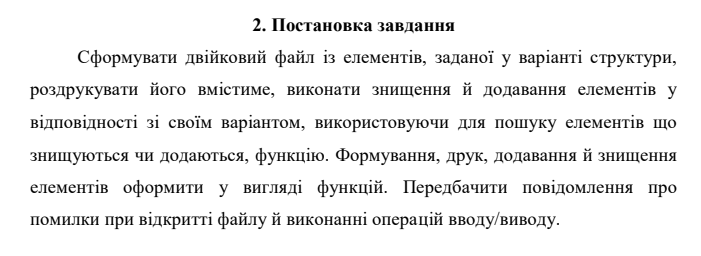
Завдання № 1 VNS Lab 6

* Варіант завдання - 24
* Деталі завдання

Завдання № 2 VNS Lab 8

Рисунок 1: VNS Lab 6

* Варіант завдання - 24
* Деталі завдання
* Важливі деталі для врахування в імплементації програми

Рисунок 2: VNS Lab 8

Структура "Студент":

1) прізвище, ім'я, по батькові;

2) номер телефону;

3) група;

4) оцінки по 3 основних предметах.

Знищити всі елементи із групи із зазначеним номером, у яких середнє

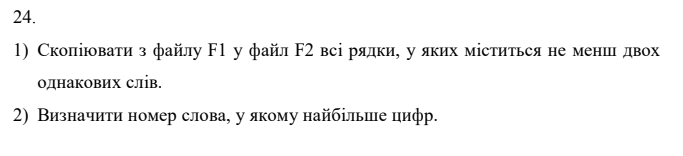
арифметичне оцінок менше заданого, додати елемент після елемента із

заданим прізвищем.

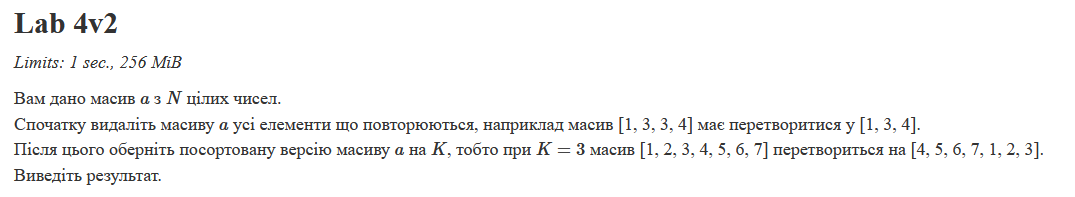
Завдання № 3 VNS Lab 9

* Варіант завдання - 24
* Деталі завдання

Завдання № 4 Algotester Lab 4

Рисунок 3: VNS Lab 9

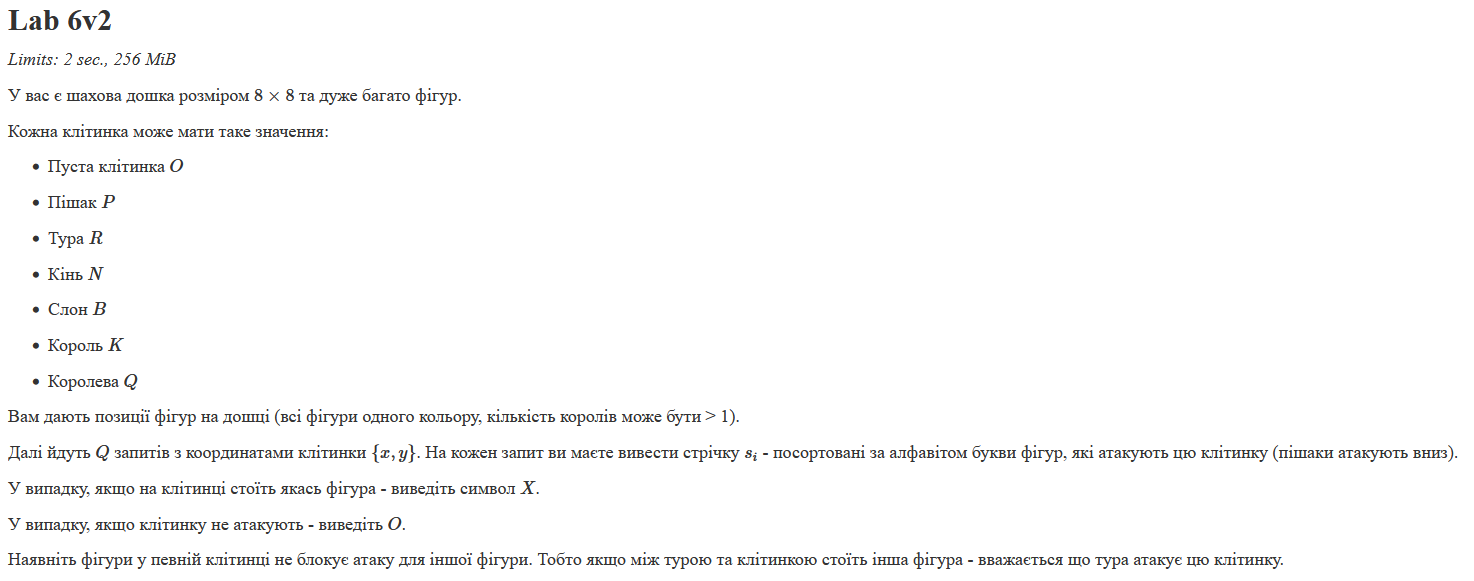
* Варіант завдання - 2
* Деталі завдання
* Важливі деталі для врахування в імплементації програми

Рисунок 4: Algotester Lab 4

Написати два варіанти розв’язку: власний та за допомогою бібліотеки algorithm

Завдання № 5 Algotester Lab 6

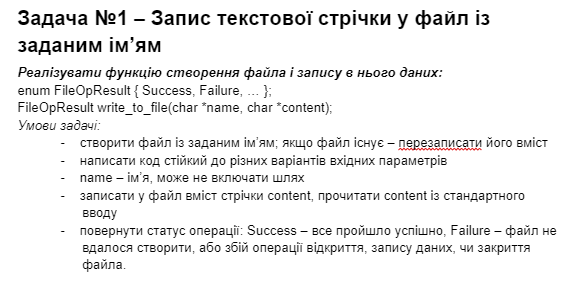
* Варіант завдання - 2
* Деталі завдання
* Важливі деталі для врахування в імплементації програми

Рисунок 5: Algotester Lab 6

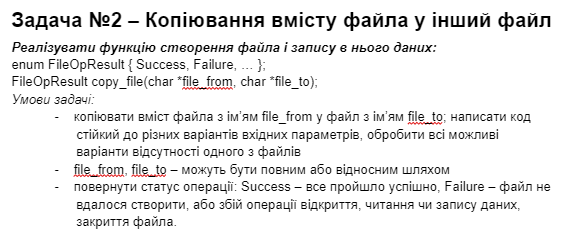
Варто розділяти процедури на підпроцедури заради більшої модульності досить громіздкого коду

Завдання № 6 Class Practice Work

* Деталі завдання

Рисунок 6: Class Practice Work Task 1

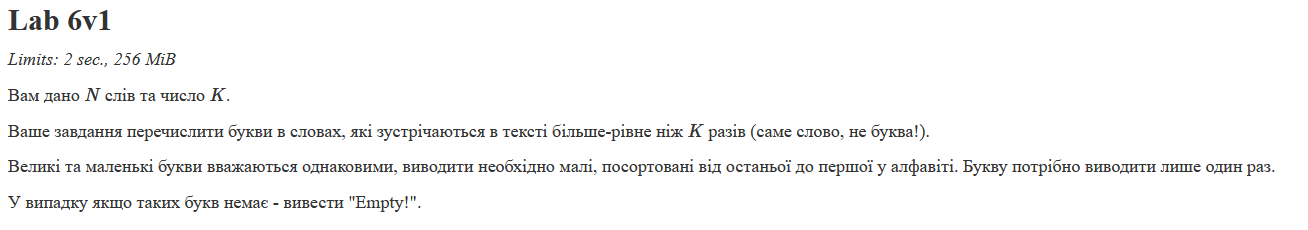
* Важливі деталі для врахування в імплементації програми

Рисунок 7: Class Practice Work Task 2

Варто використати enum для повернення стану виконання операції та виводити помилку відкриття файлу.

Завдання № 7 Self Practice Work

* Деталі завдання
* Важливі деталі для врахування в імплементації програми

Рисунок 8: Self Practice Work

Варто використати хеш-таблиці та словники для спрощення алгоритму пошуку

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

Програма № 1 "Рядки"

* Планований час на реалізацію — 10 хвилин
* Важливі деталі для врахування в імплементації

Врахувати, що процедура strtok - небезпечна

Програма № 2 "Блоковий ввід-вивід"

* Планований час на реалізацію — 30 хвилин
* Важливі деталі для врахування в імплементації

Використати структуру для організації даних

Програма № 3 "Ввід-вивід рядків"

* Планований час на реалізацію — 20 хвилин
* Важливі деталі для врахування в імплементації

Варто видати помилку, якщо файл F1 відсутній

Програма № 4 “Unique-sort-rotate”

* Планований час на реалізацію — 30 хвилин
* Важливі деталі для врахування в імплементації

Використати бібліотеку algorithm, а потім написати власні методи

Програма № 5 “Шахова дошка”

* Блок-схема

[Посилання на блок-схему](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679/files#diff-52def8a7aba8c9563833af6e99d67aa8ad80a16ae2035a7cc3f49b7c6935acf8) (надто велика для вставки в документ)

* Планований час на реалізацію — 40 хвилин
* Важливі деталі для врахування в імплементації

Розбити програму на підпрограми, які конкретно шукають, чи атакується клітинка якоюсь із фігур

Програма № 6 “Запис/клонування файлів”

* Планований час на реалізацію — 20 хвилин
* Важливі деталі для врахування в імплементації

Необхідно використати enum для повернення стану читання/запису

Програма № 7 “Букви”

* Планований час на реалізацію — 30 хвилин
* Важливі деталі для врахування в імплементації

Варто використати словники та дерева

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

Завдання № 1 [VNS Lab 6](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679/files#diff-04b3fef98d1ef9df7b0900701f26fa50662879ffe91f26b4aa4f6e1f41de2142)

#include <stdio.h>  
#include <string.h>  
  
int count\_words(char\* str, char\* separators) {  
 int count = 0;  
  
 char\* token = strtok(str, separators);  
 while (token != NULL) {  
 count++;  
 token = strtok(NULL, separators);  
 }  
  
 return count;  
}  
  
int main() {  
 char\* seps = " ,.";  
  
 char str[100];  
 printf("Enter a string: ");  
 fgets(str, 100, stdin);  
  
 int count = count\_words(str, seps);  
 printf("Number of words: %d\n", count);  
  
 return 0;  
}

Завдання № 2 [VNS Lab 8](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679/files#diff-f7ff01ee18e68f9f5fcdfa8795e85321f8a8c4a30ab2c0511b5ebbc2bff56f22)

#include <iostream>  
#include <fstream>  
#include <vector>  
  
using namespace std;  
  
constexpr size\_t GRADES\_COUNT = 3;  
struct Student {  
 string name;  
 string phone\_num;  
 string group;  
 double main\_grades[GRADES\_COUNT]{};  
};  
  
void print\_student(const Student& student);  
void read(vector<Student>& students, ifstream& file);  
void menu(vector<Student>& students);  
void write(const vector<Student>& students, ofstream& file);  
Student create\_student();  
  
constexpr string filename = "students.bin";  
  
int main() {  
 vector<Student> students;  
  
 ifstream read\_stream(filename, ios::binary);  
 if (read\_stream.is\_open()) {  
 read(students, read\_stream);  
 read\_stream.close();  
 } else cout << "Starting with empty list." << endl << endl;  
  
 menu(students);  
  
 ofstream output\_stream(filename, ios::binary | ios::trunc);  
 if (!output\_stream.is\_open()) {  
 cerr << "Error happened when tried to open file." << endl;  
 return 1;  
 }  
  
 write(students, output\_stream);  
 output\_stream.close();  
  
 return 0;  
}  
  
void print\_student(const Student& student) {  
 cout << "Name: " << student.name << "\n";  
 cout << "Phone number: " << student.phone\_num << "\n";  
 cout << "Group: " << student.group << "\n";  
 cout << "Main grades: ";  
 for (const double& grade : student.main\_grades) {  
 cout << grade << " ";  
 }  
 cout << endl;  
}  
  
void read(vector<Student>& students, ifstream& file) {  
 Student student;  
 while (file >> student.name >> student.phone\_num >> student.group) {  
 for (double& grade : student.main\_grades) {  
 file >> grade;  
 }  
 students.push\_back(student);  
 }  
}  
  
void write(const vector<Student>& students, ofstream& file) {  
 for (const Student& student : students) {  
 file << student.name << "\n" << student.phone\_num << "\n" << student.group << "\n";  
 for (const double& grade : student.main\_grades) {  
 file << grade << " ";  
 }  
 file << endl;  
 }  
}  
  
Student create\_student() {  
 string input\_buf;  
  
 Student new\_student;  
 cout << "Enter name: ";  
 getline(cin, new\_student.name);  
 cout << "Enter phone number: ";  
 getline(cin, new\_student.phone\_num);  
 cout << "Enter group: ";  
 getline(cin, new\_student.group);  
 cout << "Enter main grades: ";  
 for (double& grade : new\_student.main\_grades) {  
 getline(cin, input\_buf);  
 grade = stod(input\_buf);  
 }  
  
 return new\_student;  
}  
  
void menu(vector<Student>& students) {  
 int choice;  
 string input;  
  
 do {  
 cout << "1. Add student\n"  
 "2. Print all students\n"  
 "3. Erase student\n"  
 "4. Remove students by group\n"  
 "5. Add student after another student\n"  
 "0. Exit\n"  
 "Your choice: ";  
  
 getline(cin, input);  
 choice = stoi(input);  
  
 switch (choice) {  
 case 1: {  
 students.push\_back(create\_student());  
 break;  
 }  
 case 2: {  
 if (students.empty()) {  
 cout << "List is empty!" << endl;  
 break;  
 }  
  
 for (int i = 0; i < students.size(); ++i) {  
 cout << i + 1 << "." << endl;  
 print\_student(students[i]);  
 }  
 break;  
 }  
 case 3: {  
 int index;  
 cout << "Enter index: ";  
 getline(cin, input);  
 index = stoi(input);  
  
 if (index < 0 || index >= students.size()) {  
 cout << "Wrong index!\n";  
 break;  
 }  
 students.erase(students.begin() + index);  
  
 break;  
 }  
 case 4: {  
 string group;  
 cout << "Enter group: ";  
 getline(cin, group);  
  
 for (auto it = students.begin(); it != students.end(); it++)  
 if (it->group == group) it = students.erase(it);  
  
 break;  
 }  
 case 5: {  
 string name;  
 cout << "Enter the name of student in database: ";  
 getline(cin, name);  
  
 auto it = students.begin();  
 for (; it != students.end(); it++)  
 if (it->name == name) break;  
  
 if (it == students.end()) {  
 cout << "Wrong name!" << endl;  
 break;  
 }  
 students.insert(it + 1, create\_student());  
  
 break;  
 }  
 case 0:  
 break;  
 default: {  
 cout << "Wrong choice!" << endl;  
 break;  
 }  
 }  
 } while (choice != 0);  
}

Завдання № 3 [VNS Lab](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679/files#diff-b63fd9884e5d9d88a7937150c7082f6f4f907c989c2929a6f59c99b1ab0f17d1) 9

#include <iostream>  
#include <fstream>  
#include <vector>  
#include <sstream>  
  
using namespace std;  
  
constexpr const char\* FILE1 = "file1.txt";  
constexpr const char\* FILE2 = "file2.txt";  
  
int longestWordPosition(const vector<string>& words) {  
 if (words.empty())  
 return -1;  
  
 int longestPosition = 0;  
 for (int i = 0; i < words.size(); ++i) {  
 if (words[i].length() > words[longestPosition].length()) {  
 longestPosition = i;  
 }  
 }  
  
 return longestPosition;  
}  
  
vector<string> split(const string& line) {  
 vector<string> words;  
 string word;  
 istringstream iss(line);  
  
 while (iss >> word) {  
 words.push\_back(word);  
 }  
  
 return words;  
}  
  
bool writeLnIfHasDuplicate(const string& line, const string& filename) {  
 vector<string> words = split(line);  
 for (int i = 0; i < words.size(); ++i) {  
 for (int j = 0; j < words[i].length(); ++j) {  
 if (words[i] == words[j]) {  
 ofstream outfile(filename, ios::app);  
 outfile << line << endl;  
 return true;  
 }  
 }  
 }  
  
 return false;  
}  
  
int main() {  
 ifstream infile1(FILE1);  
 ofstream outfile2(FILE2);  
  
 if (!infile1.is\_open() || !outfile2.is\_open()) {  
 cerr << "Error opening files" << endl;  
 return 1;  
 }  
  
 vector<string> lines;  
 vector<string> words;  
  
 string line1;  
 while (getline(infile1, line1)) {  
 lines.push\_back(line1);  
  
 vector<string> s = split(line1);  
 words.insert(words.end(), s.begin(), s.end());  
  
 writeLnIfHasDuplicate(line1, FILE2);  
 }  
  
 infile1.close();  
 outfile2.close();  
  
 int longest\_pos = longestWordPosition(words);  
 cout << "Longest word: " << words[longest\_pos]  
 << " (Position: " << longest\_pos << ")" << endl;  
  
 return 0;  
}

Завдання № 4 [Algotester Lab 4](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679/files#diff-88aa77a0bfe82f09bbb4a2bc05e37f9720ce7aefffa7f08ba8a2856176d24b25)

#include <iostream>  
#include <vector>  
using namespace std;  
  
void mergeSort(vector<int>& arr);  
void mergeSort(vector<int>& arr, int left, int right);  
void merge(vector<int>& arr, int left, int mid, int right);  
  
vector<int> unique(const vector<int>& arr);  
  
vector<int> rotate(const vector<int>& arr, int k);  
  
int main() {  
 int n, k;  
 cin >> n >> k;  
  
 vector<int> arr(n);  
 for (int &el: arr) cin >> el;  
  
 mergeSort(arr);  
 arr = unique(arr);  
 arr = rotate(arr, k);  
  
 cout << arr.size() << endl;  
 for (int el: arr) cout << el << " ";  
  
 return 0;  
}  
  
void merge(vector<int>& arr, int left, int mid, int right) {  
 int n1 = mid - left + 1;  
 int n2 = right - mid;  
  
 vector<int> leftArr(n1);  
 vector<int> rightArr(n2);  
  
 for (int i = 0; i < n1; i++) {  
 leftArr[i] = arr[left + i];  
 }  
  
 for (int i = 0; i < n2; i++) {  
 rightArr[i] = arr[mid + 1 + i];  
 }  
  
 int i = 0, j = 0, k = left;  
  
 while (i < n1 && j < n2) {  
 if (leftArr[i] <= rightArr[j]) {  
 arr[k] = leftArr[i];  
 i++;  
 } else {  
 arr[k] = rightArr[j];  
 j++;  
 }  
 k++;  
 }  
  
 while (i < n1) {  
 arr[k] = leftArr[i];  
 i++;  
 k++;  
 }  
  
 while (j < n2) {  
 arr[k] = rightArr[j];  
 j++;  
 k++;  
 }  
}  
  
void mergeSort(vector<int>& arr, int left, int right) {  
 if (left < right) {  
 int mid = left + (right - left) / 2;  
  
 mergeSort(arr, left, mid);  
 mergeSort(arr, mid + 1, right);  
  
 merge(arr, left, mid, right);  
 }  
}  
  
void mergeSort(vector<int>& arr) {  
 mergeSort(arr, 0, arr.size() - 1);  
}  
  
vector<int> unique(const vector<int>& arr) {  
 vector<int> result;  
 for (int i = 0; i < arr.size(); i++) {  
 if (i == 0 || arr[i] != arr[i - 1]) {  
 result.push\_back(arr[i]);  
 }  
 }  
 return result;  
}  
  
vector<int> rotate(const vector<int>& arr, int k) {  
 k %= arr.size();  
 if (k == 0) return { arr };  
  
 vector<int> result(arr.size());  
 for (int i = k; i < arr.size(); i++) {  
 result[i - k] = arr[i];  
 }  
 for (int i = 0; i < k; i++) {  
 result[arr.size() - k + i] = arr[i];  
 }  
  
 return result;  
}

Завдання № 5 [Algotester Lab](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679/files#diff-6cb1e03f542ab82dbad3ff58de25e91ef73f2ce482b0b0b451625922e171da12) 6

#include <iostream>  
#include <vector>  
#include <algorithm>  
#include <set>  
using namespace std;  
  
bool diagonalToken(char token, const vector<string>& board, int x, int y);  
bool horizontalToken(char token, const vector<string>& board, int x, int y);  
bool verticalToken(char token, const vector<string>& board, int x, int y);  
  
bool isBishopAttacking(const vector<string>& board, int x, int y);  
bool isRookAttacking(const vector<string>& board, int x, int y);  
bool isQueenAttacking(const vector<string>& board, int x, int y);  
bool isPawnAttacking(const vector<string>& board, int x, int y);  
bool isKingAttacking(const vector<string>& board, int x, int y);  
bool isKnightAttacking(const vector<string>& board, int x, int y);  
  
bool checkOffsets(int x, int y, int m, int p);  
  
set<char> attacking(const vector<string>& board, int x, int y);  
  
constexpr int SIZE = 8;  
  
int main() {  
 vector<string> v(SIZE);  
 for(string& s : v) cin >> s;  
  
 int Q;  
 cin >> Q;  
  
 int x, y;  
 set<char> result;  
 for(int i = 0; i < Q; i++) {  
 cin >> x >> y;  
  
 result = attacking(v, x - 1, y - 1);  
  
 for (char c : result) cout << c;  
 cout << endl;  
 }  
  
 return 0;  
}  
  
bool checkOffsets(int x, int y, int m, int p) {  
 return x + m >= 0 && x + m <= 7 && y + p >= 0 && y + p <= 7;  
}  
  
bool diagonalToken(char token, const vector<string>& board, int x, int y) {  
 for (int i = 1; i < SIZE; ++i) {  
 // Check for token in left top diagonal part  
 if (x - i >= 0 && y - i >= 0 && (board[x - i][y - i] == token))  
 return true;  
  
 // Check for token in right top diagonal part  
 if (x - i >= 0 && y + i < SIZE && (board[x - i][y + i] == token))  
 return true;  
  
 // Check for token in left bottom diagonal part  
 if (x + i < SIZE && y - i >= 0 && (board[x + i][y - i] == token))  
 return true;  
  
 // Check for token in right bottom diagonal part  
 if (x + i < SIZE && y + i < SIZE && (board[x + i][y + i] == token))  
 return true;  
 }  
  
 return false;  
}  
bool horizontalToken(char token, const vector<string>& board, int x, int y) {  
 for (int i = 0; i < SIZE; ++i) {  
 // Check for token in left part  
 if (y - i >= 0 && (board[x][y - i] == token))  
 return true;  
  
 // Check for token in right part  
 if (y + i < SIZE && (board[x][y + i] == token))  
 return true;  
 }  
  
 return false;  
}  
bool verticalToken(char token, const vector<string>& board, int x, int y) {  
 for (int i = 0; i < SIZE; ++i) {  
 // Check for token in top part  
 if (x - i >= 0 && (board[x - i][y] == token))  
 return true;  
  
 // Check for token in bottom part  
 if (x + i < SIZE && (board[x + i][y] == token))  
 return true;  
 }  
  
 return false;  
}  
  
bool isBishopAttacking(const vector<string>& board, int x, int y) {  
 return diagonalToken('B', board, x, y);  
}  
bool isRookAttacking(const vector<string>& board, int x, int y) {  
 return horizontalToken('R', board, x, y) || verticalToken('R', board, x, y);  
}  
bool isQueenAttacking(const vector<string>& board, int x, int y) {  
 return diagonalToken('Q', board, x, y) ||  
 horizontalToken('Q', board, x, y) ||  
 verticalToken('Q', board, x, y);  
}  
bool isPawnAttacking(const vector<string>& board, int x, int y) {  
 return (x - 1 >= 0 && y - 1 >= 0 && board[x - 1][y - 1] == 'P') ||  
 (x - 1 >= 0 && y + 1 < SIZE && board[x - 1][y + 1] == 'P');  
}  
bool isKingAttacking(const vector<string>& board, int x, int y) {  
 for (int m = -1; m <= 1; m++)  
 for (int p = -1; p <= 1; p++)  
 if (checkOffsets(x, y, m, p) && board[x + m][y + p] == 'K')  
 return true;  
  
 return false;  
}  
bool isKnightAttacking(const vector<string>& board, int x, int y) {  
 for (int m = -2; m <= 2; m += 4)  
 for (int p = -1; p <= 1; p += 2)  
 if ((checkOffsets(x, y, m, p) && board[x + m][y + p] == 'N') ||  
 (checkOffsets(x, y, p, m) && board[x + p][y + m] == 'N'))  
 return true;  
  
 return false;  
}  
  
set<char> attacking(const vector<string>& board, int x, int y) {  
 set<char> result;  
  
 // If is not empty, return 'X'  
 if (board[x][y] != 'O') {  
 result.insert('X');  
 return result;  
 }  
  
 if (isPawnAttacking(board, x, y))  
 result.insert('P');  
  
 if (isRookAttacking(board, x, y))  
 result.insert('R');  
  
 if (isKnightAttacking(board, x, y))  
 result.insert('N');  
  
 if (isBishopAttacking(board, x, y))  
 result.insert('B');  
  
 if (isKingAttacking(board, x, y))  
 result.insert('K');  
  
 if (isQueenAttacking(board, x, y))  
 result.insert('Q');  
  
 if (result.empty())  
 result.insert('O');  
  
 return result;  
}

Завдання № 6 [Class Practice Work](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679/files#diff-17097abf0f37c18958c3ee8dc789a381495977d853e603c8b824ebf986726625)

#include <iostream>  
#include <cstring>  
using namespace std;  
  
enum FileOpResult {  
 *INDETERMINATE* = 0,  
 *SUCCESS*,  
 *FAILURE*,  
 *FILE\_NOT\_FOUND*,  
};  
  
FileOpResult write\_to\_file(const char \*name, const char \*content);  
FileOpResult copy(const char \*src\_path, const char \*dest\_path);  
  
int main() {  
 int choice = 0;  
  
 do {  
 cout << "1. Write to file\n"  
 "2. Copy file\n"  
 "3. Exit\n"  
 "Enter choice: ";  
 cin >> choice;  
  
 switch (choice) {  
 case 1: {  
 char name[256];  
 string content;  
  
 cout << "Enter file name: ";  
 cin >> name;  
  
 cout << "Enter content: ";  
 cin.ignore(INT\_MAX, '\n');  
 getline(cin, content);  
  
 if (write\_to\_file(name, content.c\_str()) == *SUCCESS*) {  
 cout << "Successfully written to file" << endl;  
 } else {  
 cout << "Failed to write to file" << endl;  
 }  
  
 break;  
 }  
 case 2: {  
 char src\_path[256];  
 char dest\_path[256];  
  
 cout << "Enter source file path: ";  
 cin >> src\_path;  
  
 cout << "Enter destination file path: ";  
 cin >> dest\_path;  
  
 if (copy(src\_path, dest\_path) == *SUCCESS*) {  
 cout << "Successfully copied file" << endl;  
 } else {  
 cout << "Failed to copy file" << endl;  
 }  
  
 break;  
 }  
 case 3: {  
 cout << "Exiting..." << endl;  
 break;  
 }  
 default: {  
 cout << "Invalid choice" << endl;  
 break;  
 }  
 }  
 } while (choice != 3);  
  
 return 0;  
}  
  
FileOpResult write\_to\_file(const char \*name, const char \*content) {  
 FILE\* file = fopen(name, "w+");  
  
 if (file == nullptr) {  
 return *FILE\_NOT\_FOUND*;  
 }  
  
 size\_t len = strlen(content);  
 fwrite(content, sizeof(char), len, file);  
  
 return fclose(file) == 0 ? *SUCCESS* : *FAILURE*;  
}  
  
FileOpResult copy(const char \*src\_path, const char \*dest\_path) {  
 FILE \*src\_file = fopen(src\_path, "r");  
 FILE \*dest\_file = fopen(dest\_path, "w+");  
  
 if (src\_file == nullptr || dest\_file == nullptr) {  
 return *FILE\_NOT\_FOUND*;  
 }  
  
 char buffer[1024];  
 size\_t bytes\_read;  
  
 while ((bytes\_read = fread(buffer, 1, sizeof(buffer), src\_file)) > 0)  
 fwrite(buffer, sizeof(char), bytes\_read, dest\_file);  
  
 return fclose(src\_file) == 0 && fclose(dest\_file) == 0  
 ? *SUCCESS* : *FAILURE*;  
}

Завдання № 7 [Self Practice Work](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679/files#diff-108401fca1e221e0b329d5a3c50725357da5113bfb28b9e55d52c1ff6682757a)

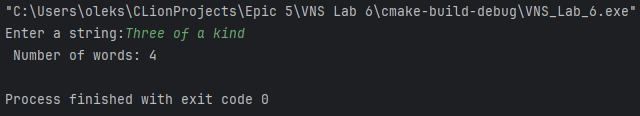
#include <iostream>  
#include <vector>  
#include <map>  
#include <set>  
#include <algorithm>  
  
using namespace std;  
  
string to\_lower(string str) {  
 for (char& c : str)  
 c = tolower(c);  
 return str;  
}  
  
int main() {  
 int n, k;  
 cin >> n >> k;  
  
 vector<string> words(n);  
  
 // first value in pair is a word  
 // second value in pair is the count of word occurrences  
 map<string, int> words\_count;  
  
 set<char> charset;  
  
 for (int i = 0; i < n; ++i) {  
 cin >> words[i];  
 // normalize word (to upper case would be fine too)  
 words[i] = to\_lower(words[i]);  
 words\_count[words[i]]++;  
 }  
  
 for (const auto& pair : words\_count)  
 if (pair.second >= k)  
 for (char c : pair.first)  
 charset.insert(c);  
  
 if (charset.empty()) {  
 cout << "Empty!";  
 return 0;  
 }  
  
 cout << charset.size() << endl;  
 for(auto it = charset.rbegin(); it != charset.rend(); ++it)  
 cout << \*it << " ";  
  
 return 0;  
}

[**Посилання на pull request**](https://github.com/artificial-intelligence-department/ai_programming_playground/pull/679)

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

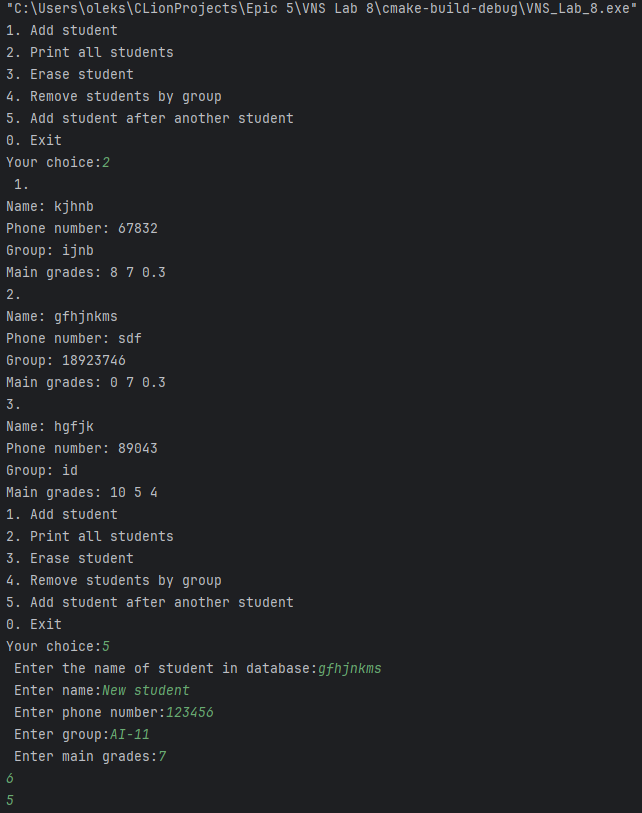
Завдання № 1 VNS Lab 6

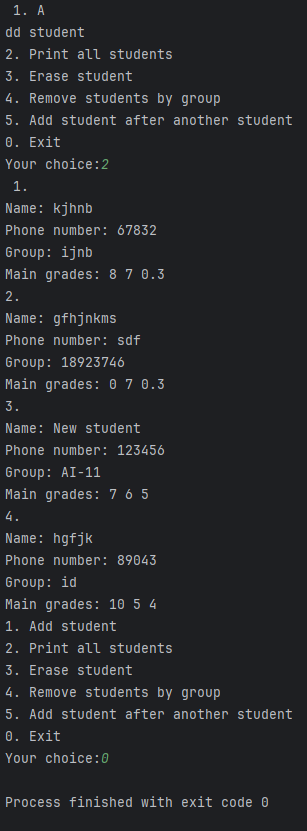
Час, витрачений на виконання завдання - 15 хвилин

Рисунок 9: Результати виконання VNS Lab 6

Завдання № 2 VNS Lab 8

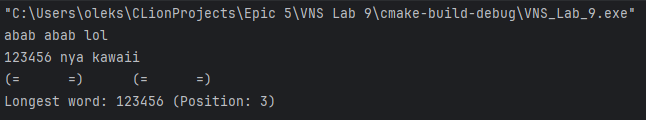
Час, витрачений на виконання завдання — 25 хвилин

Рисунок 11: Результати виконання VNS Lab 8 - Частина 1

Рисунок 10: Результати виконання VNS Lab 8 - Частина 2

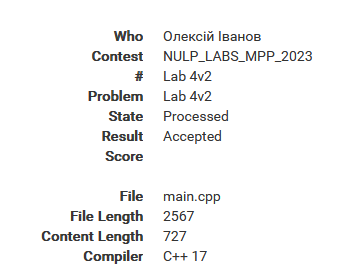
Завдання № 3 VNS Lab 9

Час, витрачений на виконання завдання — 15 хвилин

Рисунок 12: Результати виконання VNS Lab 9

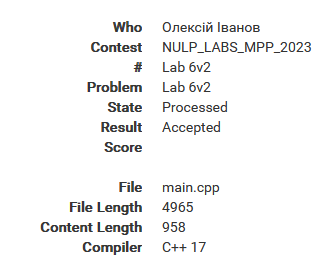
Завдання № 4 Algotester Lab 4

Час, витрачений на виконання завдання — 40 хвилин

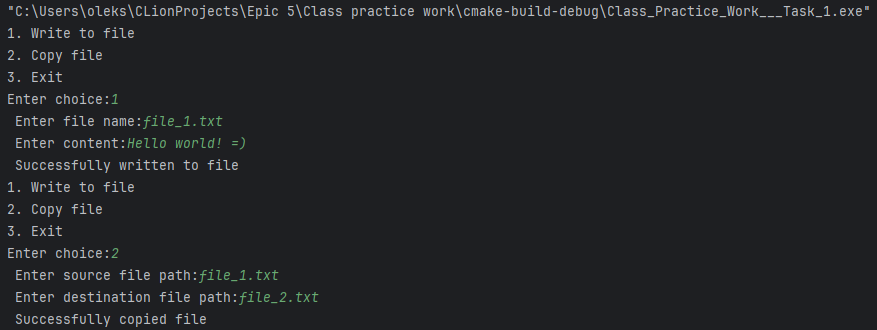
Рисунок 13: Результати виконання Algotester Lab 4

Завдання № 5 Algotester Lab 6

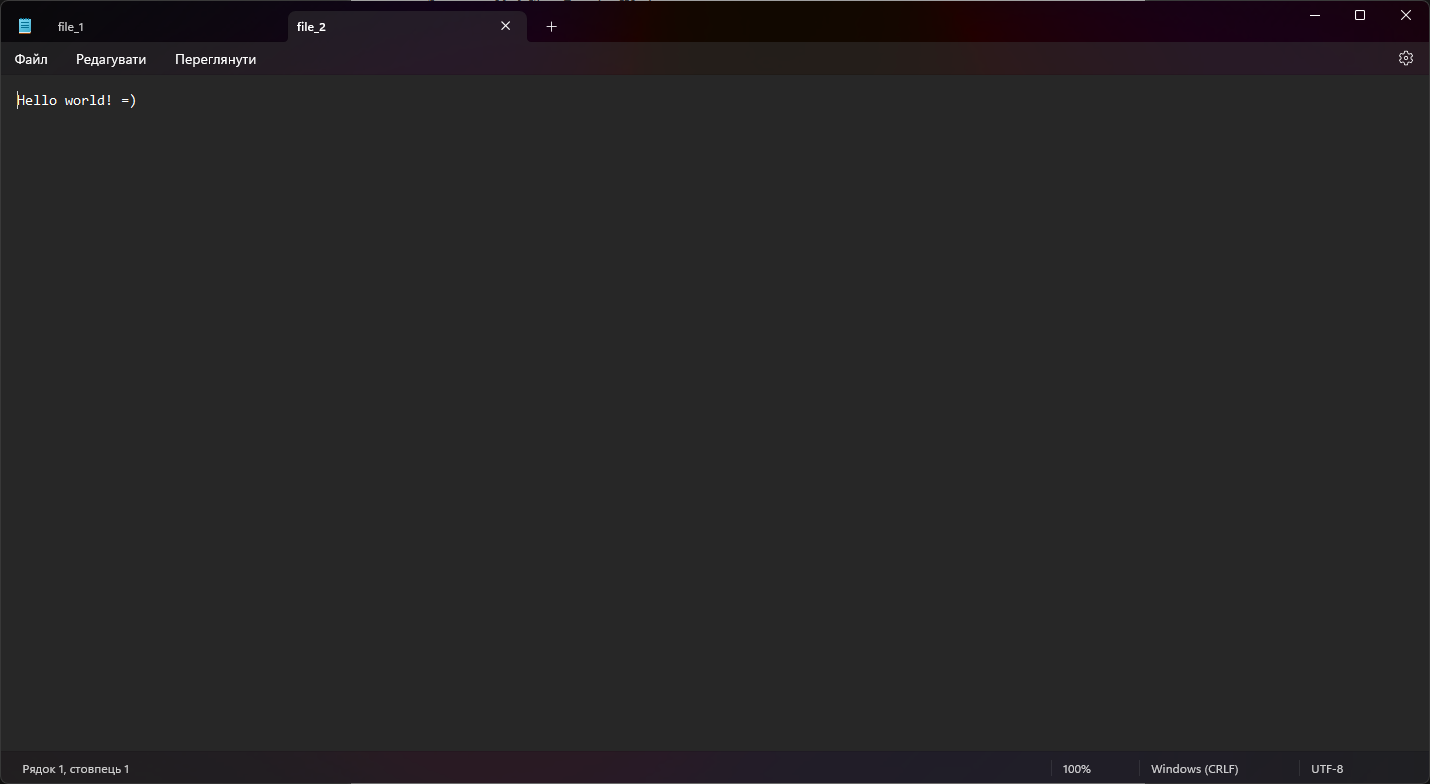
Час, витрачений на виконання завдання — 1 година

Рисунок 14: Результати виконання Algotester Lab 6

Завдання № 6 Class Practice Work

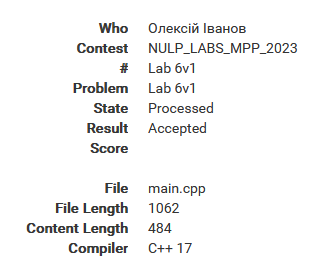
Рисунок 15: Результати виконання Class practice work

Час, витрачений на виконання завдання — 20 хвилин

Рисунок 16: Результати виконання Class practice work

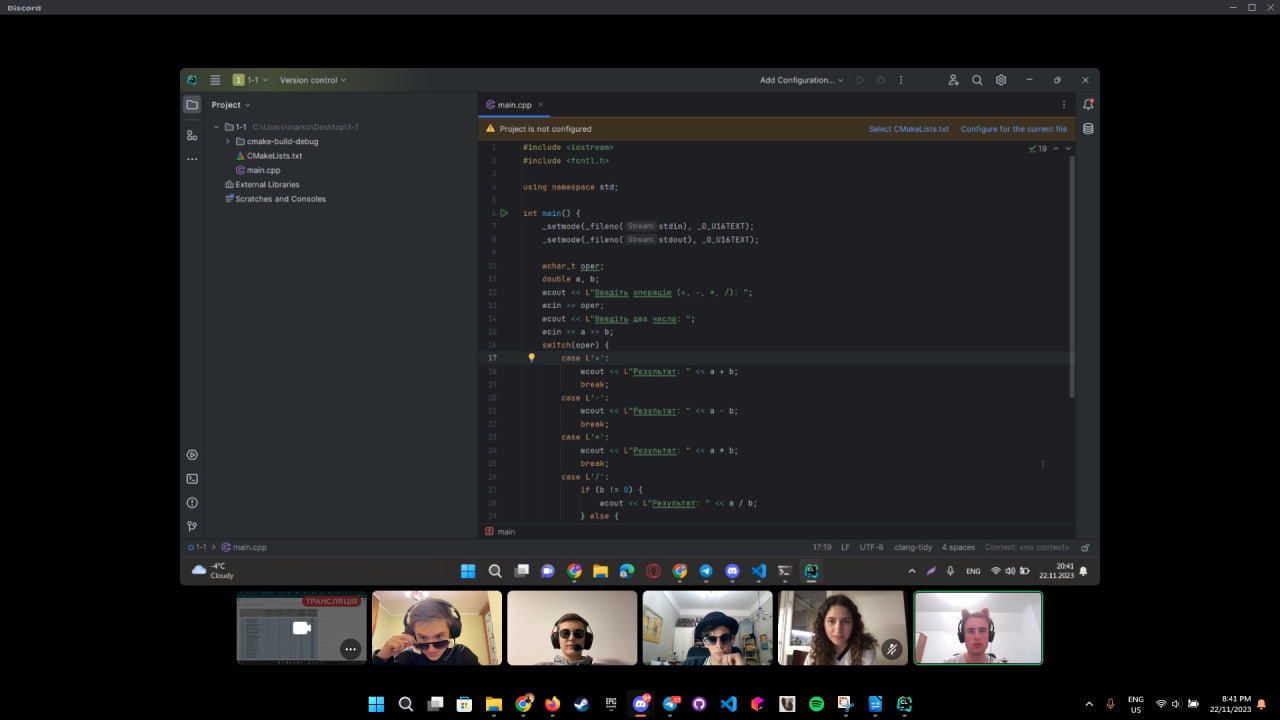
Завдання № 7 Self Practice Work

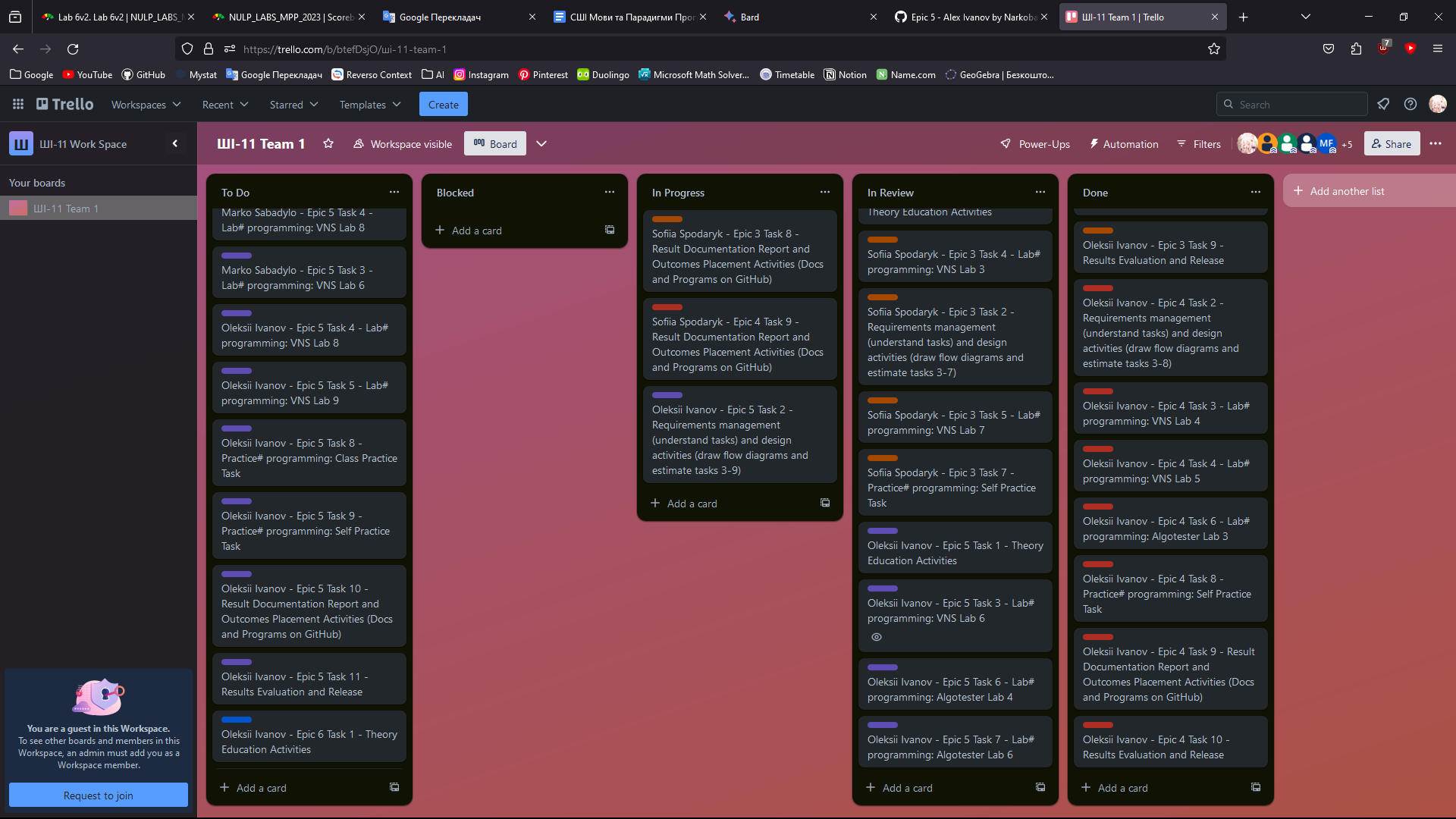
Час, витрачений на виконання завдання - 20 хвилин

Рисунок 17: Результати виконання Self practice work

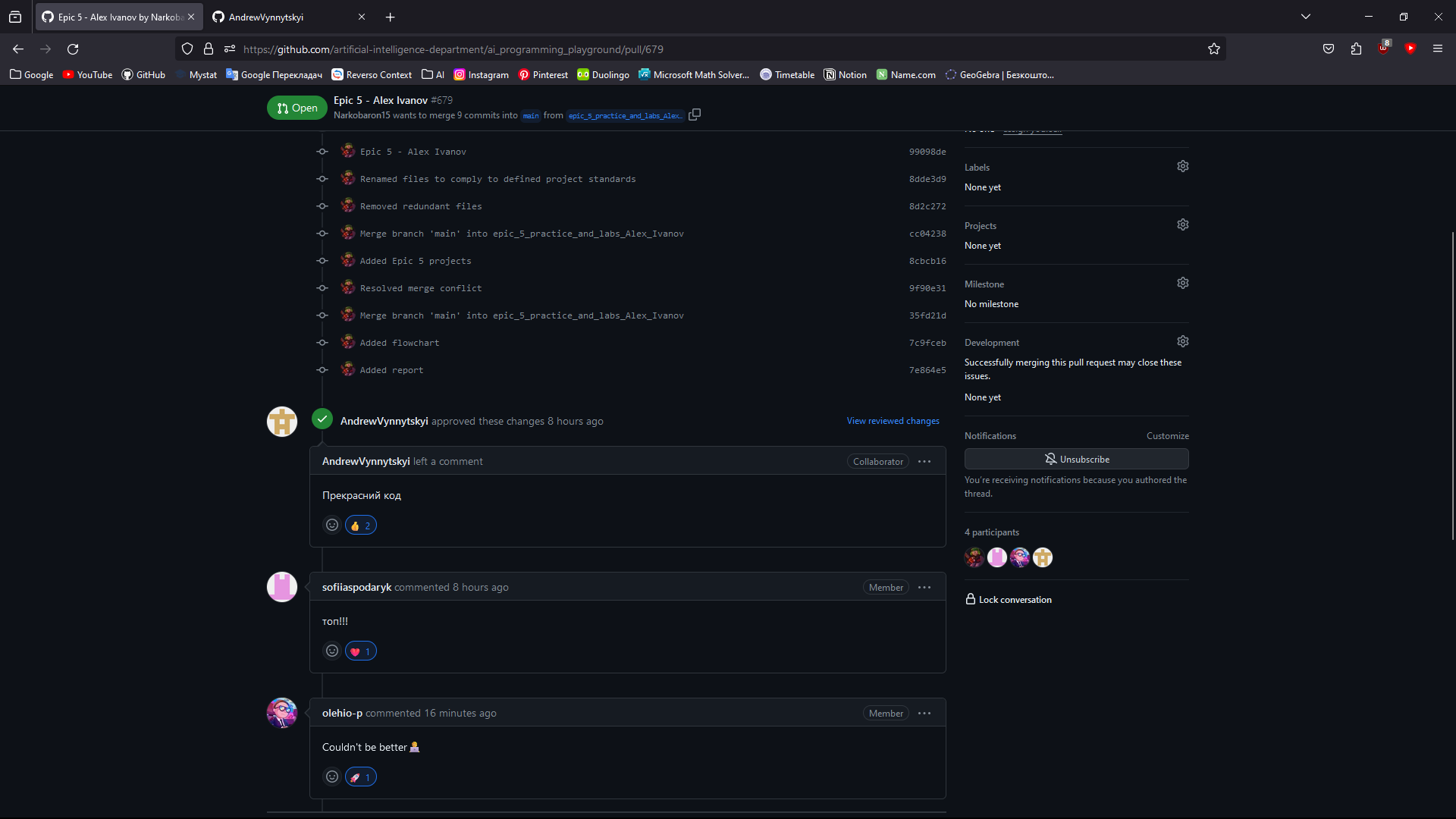
## **5. Кооперація з командою:**

* Скрін із зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло

Рисунок 18: Скріншот зустрічі з командою

Рисунок 19: Скріншот з прогресом по Trello

* Скрін з 2-му коментарями від учасників команди на пул реквесті з Ревю Роботи

Рисунок : Скріншот з коментарями команди на пул реквест

# **Висновки:**

На даній лабораторній роботі я ознайомився з основними поняттями та методами роботи з файлами в мові С++, повторив способи, як створювати, відкривати, закривати, читати та записувати дані у файли у С-стилі та С++ стилі. Також я поглибив знання в області роботи зі С-стрічками за допомогою С-методів, таких як **strtok**. В результаті виконання лабораторної роботи я поглибив практичні навички роботи з файлами в мові С++.