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

Звіт

про виконання лабораторних та практичних робіт блоку № 5

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

з дисципліни: «Основи програмування»

до:

ВНС Лабораторної Роботи № 6
ВНС Лабораторної Роботи № 8
ВНС Лабораторної Роботи № 9
Алготестер Лабораторної Роботи №4
Алготестер Лабораторної Роботи №6
Практичних Робіт до блоку №5

Виконав:
Студент групи ШІ-12
Сирватка Олександр

Тема роботи: Файлова система в C++. Робота з бінарними файлами та текстовими файлами, маніпуляції символами й рядковими змінними, як типу std::string, так і char*. Ознайомлення з можливостями стандартної бібліотеки C++ для роботи з файлами та створенням власних бібліотек для розширення функціональності.

Мета роботи: Опанувати практичні навички роботи з файлами в мові C++: створення, зчитування та запис даних у бінарні й текстові файли. Засвоїти принципи роботи з рядковими змінними різних типів (std::string і char*), вивчити використання стандартних методів та функцій для маніпуляцій з ними. Дослідити основи створення та застосування власних бібліотек для зручності повторного використання коду й розширення можливостей стандартної бібліотеки C++.

Джерела:

CS50 course
University lectures
Google + chatGPT: string functions and memory allocation

Виконання роботи:

Lab# programming: VNS Lab 6

Time expected: 30 min

Time spent: 30 min

```cpp
#include <iostream>
#include <cstring>

using namespace std;

int main() {
    char s[256];

    cout << "Введіть рядок (до 255 символів): ";
    gets(s);

    char* longest = nullptr;
    char* shortest = nullptr;

    char* word = strtok(s, " ");

    while (word != nullptr) {

        int len = strlen(word);
        if (word[len - 1] == '.') {
            word[len - 1] = '\0';
        }

        if (longest == nullptr || strlen(word) > strlen(longest)) {
            longest = word;
        }
        if (shortest == nullptr || strlen(word) < strlen(shortest)) {
            shortest = word;
        }

        word = strtok(nullptr, " ");
    }

    cout << "Найдовше слово: " << longest << endl;
    cout << "Найкоротше слово: " << shortest << endl;

    return 0;
}
```

Lab# programming: VNS Lab 8

Time expected: 1 h

Time spent: 1.5 h – 2 h

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

struct Abiturient {
    char surname[50];
    char name[50];
    char patronymic[50];
    int birthYear;
    int examGrades[3];
    double avgDiplomaScore;
};

// Масив для 5 абітурієнтів, ініціалізованих в коді
Abiturient abiturients[5] = {
    {"Ivanov", "Ivan", "Ivanovich", 2001, {180, 175, 190}, 4.5},
    {"Petrov", "Petro", "Petrovich", 2000, {185, 190, 180}, 4.6},
    {"Sidorov", "Sidir", "Sidorovich", 1999, {160, 170, 165}, 4.3},
    {"Kovalenko", "Kostyantyn", "Anatoliyovych", 2002, {190, 185, 200}, 4.8},
    {"Melnyk", "Mykola", "Volodymyrovych", 2003, {175, 165, 180}, 4.2}
};

// Функція для запису абітурієнтів у файл
void writeToFile(const char* filename, Abiturient* abits, size_t count) {
    FILE* file = fopen(filename, "wb");
    if (!file) {
        cerr << "Error opening file for writing!" << endl;
        return;
    }

    fwrite(abits, sizeof(Abiturient), count, file);  // Записуємо абітурієнтів у файл

    fclose(file);
}

// Функція для читання абітурієнтів з файлу
void readFromFile(const char* filename) {
    FILE* file = fopen(filename, "rb");
    if (!file) {
        cerr << "Error opening file for reading!" << endl;
        return;
    }

    Abiturient ab;
    while (fread(&ab, sizeof(Abiturient), 1, file)) {
        cout << "Surname: " << ab.surname << ", Name: " << ab.name << ", Patronymic: " << ab.patronymic << endl;
        cout << "Birth Year: " << ab.birthYear << endl;
        cout << "Entrance Exam Grades: ";
        for (int i = 0; i < 3; ++i) {
            cout << ab.examGrades[i] << " ";
        }
        cout << "\nAverage Diploma Score: " << ab.avgDiplomaScore << endl << endl;
    }

    fclose(file);
}
```

```cpp
// Функція для знищення абітурієнта за номером
void deleteAbiturient(const char* filename, int indexToDelete) {
    FILE* file = fopen(filename, "rb");
    if (!file) {
        cerr << "Error opening file for reading!" << endl;
        return;
    }

    FILE* tempFile = fopen("temp.dat", "wb");
    if (!tempFile) {
        cerr << "Error opening temporary file for writing!" << endl;
        fclose(file);
        return;
    }

    Abiturient ab;
    int index = 0;
    while (fread(&ab, sizeof(Abiturient), 1, file)) {
        if (index != indexToDelete) {
            fwrite(&ab, sizeof(Abiturient), 1, tempFile);  // Копіюємо всі елементи, окрім того, який треба видалити
        }
        index++;
    }

    fclose(file);
    fclose(tempFile);

    // Видаляємо старий файл і перейменовуємо тимчасовий файл в основний
    remove(filename);
    rename("temp.dat", filename);

    cout << "Abiturient deleted successfully!" << endl;
}

// Функція для додавання абітурієнта після зазначеного прізвища
void addAbiturientAfterSurname(const char* filename, const char* surname) {
    FILE* file = fopen(filename, "rb");
    if (!file) {
        cerr << "Error opening file for reading!" << endl;
        return;
    }

    FILE* tempFile = fopen("temp.dat", "wb");
    if (!tempFile) {
        cerr << "Error opening temporary file for writing!" << endl;
        fclose(file);
        return;
    }

    Abiturient ab;
    bool surnameFound = false;
    while (fread(&ab, sizeof(Abiturient), 1, file)) {
        fwrite(&ab, sizeof(Abiturient), 1, tempFile);  // Копіюємо елемент в тимчасовий файл

        // Якщо знайдений абітурієнт з таким прізвищем, додаємо нового
        if (strcmp(ab.surname, surname) == 0) {
            Abiturient newAbiturient;
            cout << "Enter surname: ";
            cin >> newAbiturient.surname;
            cout << "Enter name: ";
            cin >> newAbiturient.name;
            cout << "Enter patronymic: ";
            cin >> newAbiturient.patronymic;
            cout << "Enter birth year: ";
            cin >> newAbiturient.birthYear;
            cout << "Enter entrance exam grades (3): ";
            for (int i = 0; i < 3; ++i) {
                cin >> newAbiturient.examGrades[i];
            }
            cout << "Enter average diploma score: ";
            cin >> newAbiturient.avgDiplomaScore;

            fwrite(&newAbiturient, sizeof(Abiturient), 1, tempFile);  // Додаємо нового абітурієнта після знайденого
            surnameFound = true;
        }
    }

    fclose(file);
    fclose(tempFile);

    if (!surnameFound) {
        cout << "No abiturient found with the surname " << surname << endl;
        return;
    }

    // Видаляємо старий файл і перейменовуємо тимчасовий файл в основний
    remove(filename);
    rename("temp.dat", filename);

    cout << "Abiturient added successfully after " << surname << "!" << endl;
}
```

```cpp

int main() {
    const char* filename = "abiturients.dat";

    // Спочатку записуємо 5 абітурієнтів у файл
    writeToFile(filename, abiturients, 5);

    // Читаємо вміст файлу і виводимо його
    cout << "File contents before any operation:\n";
    readFromFile(filename);

    // Запит про знищення абітурієнта
    int indexToDelete;
    cout << "Enter the index of the abiturient to delete: ";
    cin >> indexToDelete;
    deleteAbiturient(filename, indexToDelete);

    // Запит про додавання абітурієнта після певного прізвища
    char surname[50];
    cout << "Enter the surname after which you want to add a new abiturient: ";
    cin >> surname;
    addAbiturientAfterSurname(filename, surname);

    // Після всіх операцій виводимо вміст файлу
    cout << "\nFile contents after all operations:\n";
    readFromFile(filename);

    return 0;
}

```

Lab# programming: VNS Lab 9

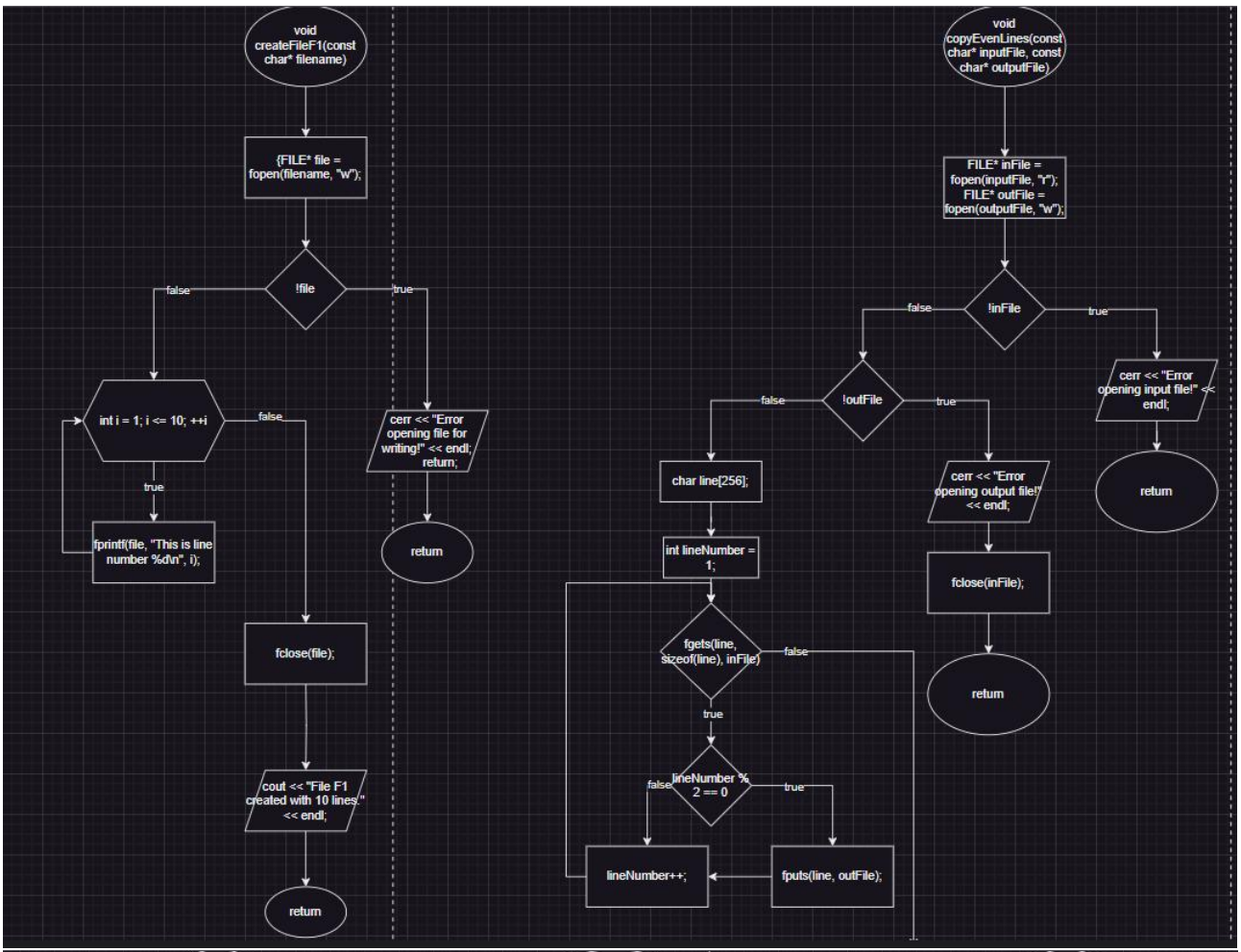
Time expected: 1.5 h

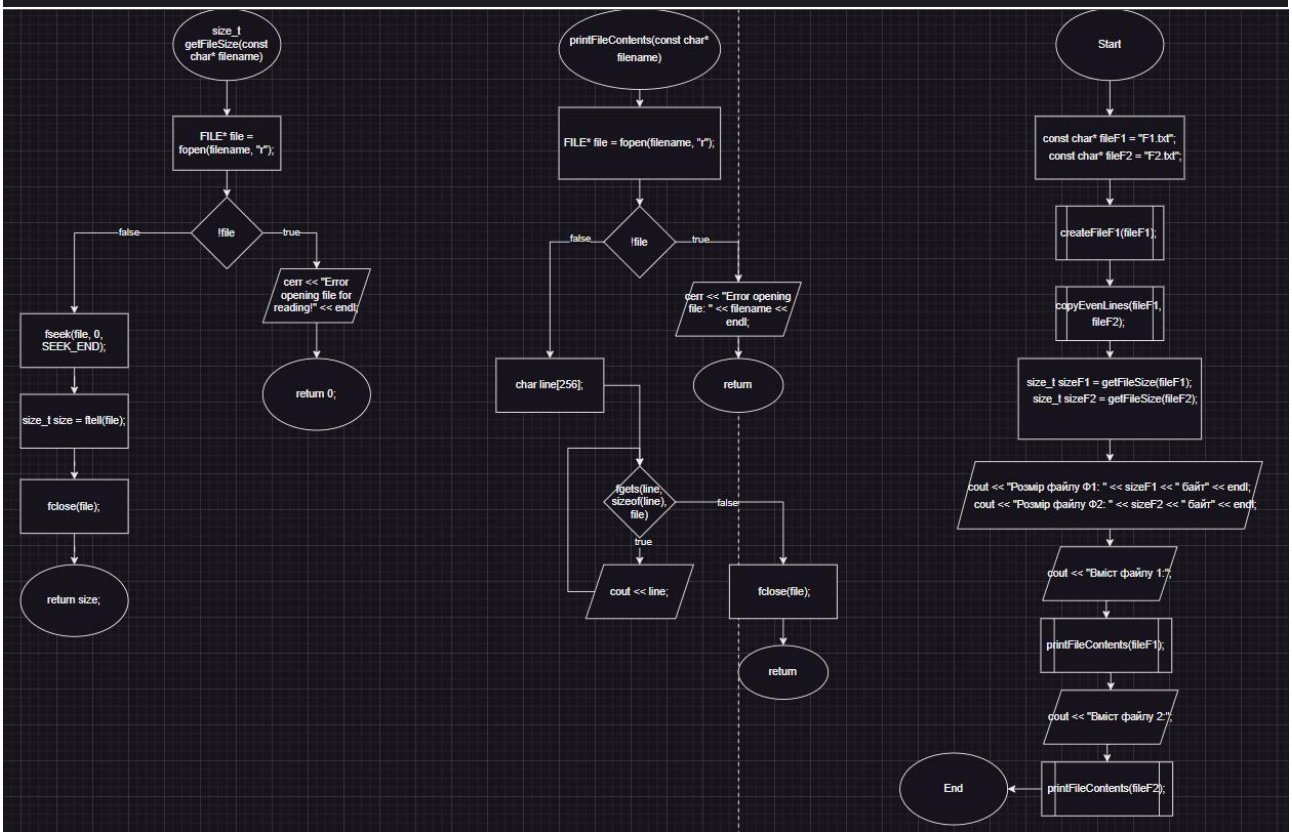
Time spent: 2.5 h

```cpp
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

// Функція для створення файлу F1 з 10 рядками
void createFileF1(const char* filename) {
    FILE* file = fopen(filename, "w");
    if (!file) {
        cerr << "Error opening file for writing!" << endl;
        return;
    }

    // Записуємо 10 рядків у файл F1
    for (int i = 1; i <= 10; ++i) {
        fprintf(file, "This is line number %d\n", i);
    }

    fclose(file);
    cout << "File F1 created with 10 lines." << endl;
}

// Функція для копіювання парних рядків з F1 у F2
void copyEvenLines(const char* inputFile, const char* outputFile) {
    FILE* inFile = fopen(inputFile, "r");
    FILE* outFile = fopen(outputFile, "w");
    if (!inFile) {
        cerr << "Error opening input file!" << endl;
        return;
    }
    if (!outFile) {
        cerr << "Error opening output file!" << endl;
        fclose(inFile);
        return;
    }

    char line[256];  // Буфер для зчитування рядків
    int lineNumber = 1;

    // Читання і запис кожного рядка з F1
    while (fgets(line, sizeof(line), inFile)) {
        if (lineNumber % 2 == 0) {
            fputs(line, outFile);  // Записуємо парний рядок у F2
        }
        lineNumber++;
    }

    fclose(inFile);
    fclose(outFile);
    cout << "Even lines copied to file F2." << endl;
}

// Функція для підрахунку розміру файлу
size_t getFileSize(const char* filename) {
    FILE* file = fopen(filename, "r");
    if (!file) {
        cerr << "Error opening file for reading!" << endl;
        return 0;
    }

    fseek(file, 0, SEEK_END);  // Переходимо в кінець файлу
    size_t size = ftell(file); // Отримуємо розмір файлу
    fclose(file);

    return size;
}
```

```cpp
    return size;
}

// Функція для друку вмісту файлу
void printFileContents(const char* filename) {
    FILE* file = fopen(filename, "r");  // Відкриваємо файл для читання
    if (!file) {
        cerr << "Error opening file: " << filename << endl;
        return;
    }

    char line[256];  // Масив для зберігання рядка

    // Читаємо файл рядок за рядком
    while (fgets(line, sizeof(line), file)) {
        cout << line;  // Виводимо кожен рядок
    }

    fclose(file);  // Закриваємо файл після завершення роботи з ним
}

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

    // Створюємо файл F1 з 10 рядками
    createFileF1(fileF1);

    // Копіюємо парні рядки з F1 у F2
    copyEvenLines(fileF1, fileF2);

    // Виводимо розміри файлів F1 та F2
    size_t sizeF1 = getFileSize(fileF1);
    size_t sizeF2 = getFileSize(fileF2);

    cout << "Розмір файлу Ф1: " << sizeF1 << " байт" << endl;
    cout << "Розмір файлу Ф2: " << sizeF2 << " байт" << endl;

    cout << "Вміст файлу 1:";
    printFileContents(fileF1);
    cout << "Вміст файлу 2:";
    printFileContents(fileF2);

    return 0;
}
```

Flowchart:


void createFileF1(const char* filename)
{FILE* file = fopen(filename, "w");
!file
false
true
int i = 1; i <= 10; ++i
false
true
fprintf(file, "This is line number %d\n", i);
fclose(file);
cout << "File F1 created with 10 lines." << endl;
return
cerr << "Error opening file for writing!" << endl;
return
void copyEvenLines(const char* inputFile, const char* outputFile)
FILE* inFile = fopen(inputFile, "r");
FILE* outFile = fopen(outputFile, "w");
!inFile
false
true
cerr << "Error opening input file!" << endl;
return
!outFile
false
true
cerr << "Error opening output file!" << endl;
fclose(inFile);
return
char line[256];
int lineNumber = 1;
fgets(line, sizeof(line), inFile)
false
true
lineNumber % 2 == 0
false
true
fputs(line, outFile);
lineNumber++;


size_t getFileSize(const char* filename)
FILE* file = fopen(filename, "r");
!file
false
true
cerr << "Error opening file for reading!" << endl;
return 0;
fseek(file, 0, SEEK_END);
size_t size = ftell(file);
fclose(file);
return size;
printFileContents(const char* filename)
FILE* file = fopen(filename, "r");
!file
false
true
cerr << "Error opening file: " << filename << endl;
return
char line[256];
fgets(line, sizeof(line), file)
false
true
cout << line;
fclose(file);
return
Start
const char* fileF1 = "F1.txt";
const char* fileF2 = "F2.txt";
createFileF1(fileF1);
copyEvenLines(fileF1, fileF2);
size_t sizeF1 = getFileSize(fileF1);
size_t sizeF2 = getFileSize(fileF2);
cout << "Розмір файлу Ф1: " << sizeF1 << " байт" << endl;
cout << "Розмір файлу Ф2: " << sizeF2 << " байт" << endl;
cout << "Вміст файлу 1:";
printFileContents(fileF1);
cout << "Вміст файлу 2:";
printFileContents(fileF2);
End

Lab# programming: Algotester Lab 4

3.1

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

using namespace std;

int main() {
    int N;
    cin >> N;
    vector<int> arr(N);
    for (int i = 0; i < N; ++i) {
        cin >> arr[i];
    }

    // Розділяємо масив на три частини за остачею від ділення на 3
    vector<int> mod0, mod1, mod2;
    for (int num : arr) {
        if (num % 3 == 0) mod0.push_back(num);
        else if (num % 3 == 1) mod1.push_back(num);
        else mod2.push_back(num);
    }

    // Сортуємо відповідно до умов
    sort(mod0.begin(), mod0.end());  // для остачі 0 - за зростанням
    sort(mod1.rbegin(), mod1.rend());  // для остачі 1 - за спаданням
    sort(mod2.begin(), mod2.end());  // для остачі 2 - за зростанням

    // Об'єднуємо три масиви
    vector<int> result;
    result.insert(result.end(), mod0.begin(), mod0.end());
    result.insert(result.end(), mod1.begin(), mod1.end());
    result.insert(result.end(), mod2.begin(), mod2.end());

    // Видаляємо дублікати
    result.erase(unique(result.begin(), result.end()), result.end());

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

    return 0;
}
```

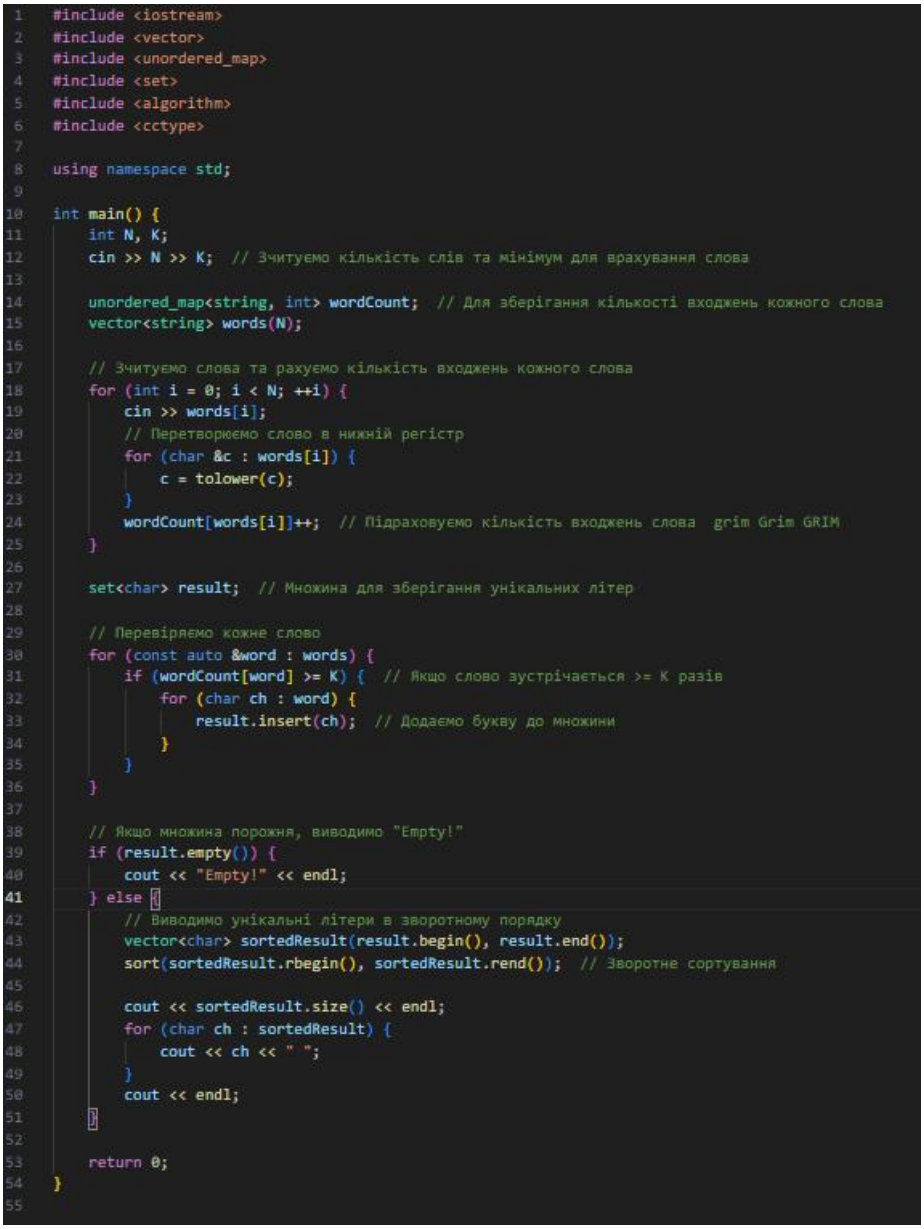
3.2

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

using namespace std;

// Об'єднана функція для сортування злиттям
void mergeSort(vector<int>& vec, int left, int right, bool ascending) {
    if (left >= right) return;  // Базовий випадок

    int mid = left + (right - left) / 2;

    // Рекурсивне сортування лівої та правої частин
    mergeSort(vec, left, mid, ascending);
    mergeSort(vec, mid + 1, right, ascending);

    // Об'єднання відсортованих підмасивів
    int n1 = mid - left + 1;
    int n2 = right - mid;

    vector<int> L(n1), R(n2);
    for (int i = 0; i < n1; ++i)
        L[i] = vec[left + i];
    for (int i = 0; i < n2; ++i)
        R[i] = vec[mid + 1 + i];

    int i = 0, j = 0, k = left;
    while (i < n1 && j < n2) {
        if ((ascending && L[i] <= R[j]) || (!ascending && L[i] >= R[j])) {
            vec[k] = L[i];
            i++;
        } else {
            vec[k] = R[j];
            j++;
        }
        k++;
    }

    while (i < n1) {
        vec[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        vec[k] = R[j];
        j++;
        k++;
    }
}

// Функція для видалення дублікатів з масиву
void removeDuplicates(vector<int>& vec) {
    vector<int> uniquearr;
    for (int i = 0; i < vec.size(); ++i) {
        if (i == 0 || vec[i] != vec[i - 1]) {
            uniquearr.push_back(vec[i]);
        }
    }
    vec = uniquearr;
}

int main() {
    int N;
    cin >> N;
    vector<int> arr(N);
    for (int i = 0; i < N; ++i) {
        cin >> arr[i];
    }

    // Розділяємо масив на три частини за остачею від ділення на 3
    vector<int> mod0, mod1, mod2;
    for (int num : arr) {
        if (num % 3 == 0) mod0.push_back(num);
        else if (num % 3 == 1) mod1.push_back(num);
        else mod2.push_back(num);
    }

    // Сортуємо відповідно до умов
    mergeSort(mod0, 0, mod0.size() - 1, true);   // За зростанням
    mergeSort(mod1, 0, mod1.size() - 1, false);  // За спаданням
    mergeSort(mod2, 0, mod2.size() - 1, true);   // За зростанням

    // Об'єднуємо три масиви
    vector<int> result;
    result.insert(result.end(), mod0.begin(), mod0.end());
    result.insert(result.end(), mod1.begin(), mod1.end());
    result.insert(result.end(), mod2.begin(), mod2.end());

    // Видаляємо дублікати
    removeDuplicates(result);

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

    return 0;
}

```

Time expected: 1 hour

Time spent: 2 hours

Lab# programming: Algotester Lab 6



```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <set>
#include <algorithm>
#include <cctype>

using namespace std;

int main() {
    int N, K;
    cin >> N >> K;  // Зчитуємо кількість слів та мінімум для врахування слова

    unordered_map<string, int> wordCount;  // Для зберігання кількості входжень кожного слова
    vector<string> words(N);

    // Зчитуємо слова та рахуємо кількість входжень кожного слова
    for (int i = 0; i < N; ++i) {
        cin >> words[i];
        // Перетворюємо слово в нижній регістр
        for (char &c : words[i]) {
            c = tolower(c);
        }
        wordCount[words[i]]++;  // Підраховуємо кількість входжень слова  grim Grim GRIM
    }

    set<char> result;  // Множина для зберігання унікальних літер

    // Перевіряємо кожне слово
    for (const auto &word : words) {
        if (wordCount[word] >= K) {  // Якщо слово зустрічається >= K разів
            for (char ch : word) {
                result.insert(ch);  // Додаємо букву до множини
            }
        }
    }

    // Якщо множина порожня, виводимо "Empty!"
    if (result.empty()) {
        cout << "Empty!" << endl;
    } else {
        // Виводимо унікальні літери в зворотному порядку
        vector<char> sortedResult(result.begin(), result.end());
        sort(sortedResult.rbegin(), sortedResult.rend());  // Зворотне сортування

        cout << sortedResult.size() << endl;
        for (char ch : sortedResult) {
            cout << ch << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Time expected: 4h

Time spent: 4h

Practice# programming: Class Practice Task

```cpp
#include<iostream>
#include<string>
#include<stdio.h>
using namespace std;

enum FileOpResult {Success, Failure};
FileOpResult write_to_file(const char *name, const char *content){
    // if (name == nullptr || strcmp(name, "") == 0 || !hasEnding(name, ".txt")) {
    //     return FileOpResult::Failure;
    // }
    FILE* fileStream;
    fileStream = fopen(name, "w");
    if (fileStream == nullptr){
        cerr << "Не вдалося створити файл";
        return FileOpResult::Failure;
    }
    fputs(content, fileStream);
    fclose(fileStream);
    return FileOpResult::Success;
}
FileOpResult copy_file(const char *file_from, const char *file_to){
    // if (name == nullptr || strcmp(name, "") == 0 || !hasEnding(name, ".txt")) {
    //     return FileOpResult::Failure;
    // }
    FILE* fileStream1;
    fileStream1 = fopen(file_from, "r");
    if (fileStream1 == nullptr){
        cerr << "Не вдалося створити файл";
        return FileOpResult::Failure;
    }
    FILE* fileStream2;
    fileStream2 = fopen(file_to, "w");
    if (fileStream2 == nullptr){
        cerr << "Не вдалося створити файл";
        fclose(fileStream1);
        return FileOpResult::Failure;
    }
    const int size= 64;
    char arr[size];
    while(fgets(arr, size, fileStream1)){
        fputs(arr, fileStream2);
    }
    fclose(fileStream1);
    fclose(fileStream2);
}


int main() {
    string name, content;
    cin >> name;
    cin >> content;
    const char* namec = name.c_str();
    const char* contentc = content.c_str();
    string file_from, file_to;
    cin >> file_from;
    cin >> file_to;
    const char* file_fromc = file_from.c_str();
    const char* file_toc = file_to.c_str();
   // write_to_file(namec, contentc);
    copy_file(file_fromc, file_toc);
    return 0;
}

```

Time expected: 1 h

Time spent: 40 min

Practice# programming: Self Practice Task

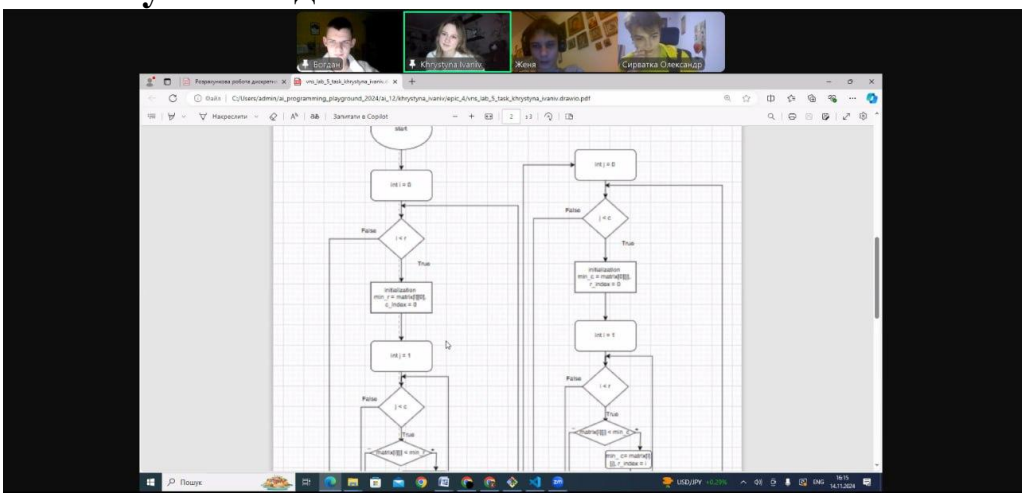
```cpp
#include <iostream>
#include <string>

using namespace std;

int main()
{
    int n;
    cin >> n;
    string ruad;
    cin >> ruad;
    int s_k = 0, s_v = 0, g_k = 0, g_v = 0;
    for(int i = 0; i < n; i++)
    {
        if(ruad[i] == 'V') s_v++;
        else s_k++;

        if(s_k >= 11 && s_k - s_v >= 2){
            g_k++;
            s_v = 0;
            s_k = 0;
        }
        if(s_v >= 11 && s_v - s_k >= 2){
            g_v++;
            s_k = 0;
            s_v = 0;
        }
    }
    cout << g_k << ":" << g_v << endl;
    if(s_v != 0 || s_k != 0){
        cout << s_k << ":" << s_v << endl;
    }
    return 0;
}

```

Time expected: 30 min

Time spent: 30 min

Pull request:https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/493

Робота у команді:



Висновок: У ході роботи було вивчено основи роботи з файловою системою в C++: опрацьовано принципи обробки текстових і бінарних файлів, включаючи процеси запису, зчитування й редагування даних. Завдяки використанню різних типів рядкових змінних (std::string та char*) вдалося ознайомитися з різними підходами до зберігання й обробки текстових даних. Використання стандартної бібліотеки значно спростило роботу з файлами, дозволяючи зосередитися на вирішенні основних завдань.