Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 10
з навчальної дисципліни
“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ ОБРОБЛЕННЯ ДАНИХ СКЛАДОВИХ ТИПІВ З ФАЙЛОВИМ ВВЕДЕННЯМ/ВИВЕДЕНЯМ

ЗАВДАННЯ ВИДАВ
доцент кафедри кібербезпеки
та програмного забезпечення
Доренський О. П.
https://github.com/odorenskyi/

ВИКОНАВ
студент академічної групи КН-24
Мироненко Я.М

ПЕРЕВІРИВ
ст. викладач кафедри кібербезпеки
та програмного забезпечення
Коваленко Анастасія Сергіївна

Кропивницький – 2025

ТЕМА: РЕАЛІЗАЦІЯ ПРОГРАМНИХ МОДУЛІВ ОБРОБЛЕННЯ ДАНИХ СКЛАДОВИХ ТИПІВ З ФАЙЛОВИМ ВВЕДЕННЯМ/ВИВЕДЕНЯМ.

МЕТА: Полягає у набутті ґрунтових вмінь і практичних навичок реалізації у Code::Blocks IDE мовою програмування С++ програмних модулів створення й оброблення даних типів масив, структура, об'єднання, множина, перелік, перетворення типів даних, використання файлових потоків та функцій стандартних бібліотек для оброблення символьної інформації.

https://github.com/odorenskyi/

Варіант 4

— *Вхідний текст - вміст вхідного текстового файлу* —

Довільне слово українською мовою.

— *Задача 10.1* —

У ***вихідний*** текстовий файл записати:

- авторську інформацію: ім'я й прізвище розробника модуля, установа/організація, місто, країна, рік розробки;
- кількість приголосних літер у слові із вхідного файла;
- повідомлення, чи є слово із вхідного файла у наступній краплинці Віталія Іващенка:

> Про себе не кажи недобрих слів,
> Бо має сказане таємну силу.
> Кажи: «Я сильний, впевнений, щасливий!»
> І буде саме так, як ти хотів!

— *Задача 10.2* —

У ***вхідний*** текстовий файл дописати:

- першу і останню літеру слова, яке міститься у цьому файлі, а також , дату й час дозапису інформації.

— *Задача 10.3* —

Вхідні дані – числові значення x, y, z та натуральне число b. У ***вихідний*** текстовий файл дописати:

- результати виконання функцій із заголовкового файлу Modules*Прізвище*.h `s_calculation` з аргументами x, y, z;
- число b у двійковому коді.

- Мова повідомлень – українська (*наприклад*, якщо у вихідний файл записується кількість символів у вхідному файлі, то модуль повинен сформувати й записати/дописати повноцінне речення: "У файлі `ВхФайл.txt` міститься 257 символів.").
- Вхідний файл `*.txt` створється користувачем, у який за допомогою текстового редактора (у Windows – Блокнот) записується вхідний текст відповідно до завдання; вихідний файл створюється програмним модулем; імена вхідного й вихідного файлів є параметрами відповідного модуля.
- Перед читання/записом з/у файловий потік слід реалізувати перевірку його відкриття; після завершення – закрити всі відкриті файлові потоки.
- Оброблення текстових файлів рекомендовано реалізувати за допомогою файлових потоків `ofstream` та `ifstream` <fstream> С++.
- Для отримання локальної дати й часу ОС можна використати стандартні функції `time`, `ctime`, `localtime`, `asctime`, реалізовані у `ctime` / `time.h`.

Аналіз і постановка задачі 10.1

У межах лабораторної роботи необхідно реалізувати програмний модуль, що працює з вхідним і вихідним текстовими файлами. Вхідним є текстовий файл, у якому міститься довільне слово українською мовою. На основі даних цього файлу програма повинна створити вихідний файл, у якому зберігається кілька типів інформації відповідно до умов задачі.

На першому етапі (задача 10.1) необхідно записати у вихідний файл авторську інформацію: ім’я та прізвище розробника, назву установи чи організації, місто, країну та рік розробки. Далі потрібно визначити кількість приголосних літер у слові з вхідного файлу. Також слід проаналізувати, чи входить це слово у поетичний текст Віталія Іващенка, який наведено в умові, і за результатами цього вивести відповідне повідомлення.

Вхід: текстовий файл з одним словом українською мовою.

Вихід:

- Авторська інформація.
- Кількість приголосних у цьому слові.
- Повідомлення, чи є це слово у вірші Іващенка.

Аналіз і постановка задачі 10.2

У другій частині завдання (задача 10.2) потрібно дописати у вихідний файл першу та останню літери слова, що міститься у вхідному файлі, а також дату та час виконання дописування.

Вхід: те ж саме слово.

Вихід:

- Перша та остання літера слова.
- Дата й час дописування.

Аналіз і постановка задачі 10.3

Третя частина завдання (задача 10.3) стосується обробки числових вхідних даних. Зокрема, потрібно обробити цілі значення змінних x, y, z і натуральне число b. У вихідний файл слід дописати результат виконання функції s_calculation, яка визначена в заголовковому файлі Modules/Прізвище.h і приймає аргументи x, y, z.

Крім того, потрібно перевести число b у двійкову систему числення та записати отримане значення у файл.

Особливу увагу необхідно приділити правильному відкриттю, зчитуванню та дописуванню файлів, з дотриманням формату і коректного представлення даних. Рекомендовано використовувати засоби роботи з датою та часом із бібліотеки <ctime> або <time.h>, а також файлові потоки з <fstream>

Вхід: значення x, y, z, b.

Вихід:

- Результат виклику функції s_calculation(x, y, z).
- Число b у двійковому представленні.

Лістинг Функцій 10.1 - 10.2 - 10.3

```
#include <iostream>
#include <fstream>      // Для роботи з файлами
#include <ctime>        // Для отримання дати та часу
#include <cctype>       // Для перевірки isalpha()
#include <cwctype>
#include <string>       // Робота з рядками
#include <sstream>      // Форматування тексту
#include <cmath>
#include <limits>
#include <windows.h>
#include <algorithm>
#include <codecvt>
#include <locale>
#include <vector>

// Lab_10
// Функція для збереження UTF-8 тексту у файл з BOM (Byte Order Mark)
bool WriteUtf8File(const string& filename, const string& text, bool append = false) {
    // Створюємо або відкриваємо файл
    DWORD createMode = append ? OPEN_ALWAYS : CREATE_ALWAYS;
    HANDLE hFile = CreateFileA(
        filename.c_str(),
        GENERIC_WRITE,
        0,
        NULL,
        createMode,
        FILE_ATTRIBUTE_NORMAL,
        NULL
    );

    if (hFile == INVALID_HANDLE_VALUE) {
        return false;
```

```
    }

    // Якщо режим дозапису, переходимо в кінець файлу
    if (append) {
        SetFilePointer(hFile, 0, NULL, FILE_END);
    } else {
        // Якщо створюємо новий файл, додаємо BOM
        DWORD bytesWritten;
        unsigned char bom[] = {0xEF, 0xBB, 0xBF};
        WriteFile(hFile, bom, 3, &bytesWritten, NULL);
    }

    DWORD bytesWritten;
    bool success = WriteFile(
        hFile,
        text.c_str(),
        text.size(),
        &bytesWritten,
        NULL
    );

    CloseHandle(hFile);
    return success && (bytesWritten == text.size());
}

// Функція для читання UTF-8 тексту з файлу
string ReadUtf8File(const string& filename) {
    HANDLE hFile = CreateFileA(
        filename.c_str(),
        GENERIC_READ,
        FILE_SHARE_READ,
        NULL,
        OPEN_EXISTING,
        FILE_ATTRIBUTE_NORMAL,
        NULL
    );

    if (hFile == INVALID_HANDLE_VALUE) {
        return "";
    }

    DWORD fileSize = GetFileSize(hFile, NULL);
    if (fileSize == INVALID_FILE_SIZE) {
        CloseHandle(hFile);
        return "";
    }

    vector<char> buffer(fileSize);
    DWORD bytesRead;
    bool success = ReadFile(
        hFile,
        buffer.data(),
        fileSize,
        &bytesRead,
```

```
        NULL
    );

    CloseHandle(hFile);

    if (!success || bytesRead == 0) {
        return "";
    }

    // Пропускаємо BOM, якщо він є
    size_t start = 0;
    if (fileSize >= 3 &&
        static_cast<unsigned char>(buffer[0]) == 0xEF &&
        static_cast<unsigned char>(buffer[1]) == 0xBB &&
        static_cast<unsigned char>(buffer[2]) == 0xBF) {
        start = 3;
    }

    return string(buffer.begin() + start, buffer.end());
}

// Функція для конвертації UTF-8 у широкі символи
wstring Utf8ToWide(const string& utf8) {
    if (utf8.empty()) {
        return L"";
    }

    int size_needed = MultiByteToWideChar(CP_UTF8, 0, utf8.c_str(), -1, NULL, 0);
    vector<wchar_t> wideStr(size_needed);
    MultiByteToWideChar(CP_UTF8, 0, utf8.c_str(), -1, wideStr.data(), size_needed);

    return wstring(wideStr.begin(), wideStr.end() - 1); // Видаляємо нульовий символ
}

// Функція для конвертації широких символів в UTF-8
string WideToUtf8(const wstring& wide) {
    if (wide.empty()) {
        return "";
    }

    int size_needed = WideCharToMultiByte(CP_UTF8, 0, wide.c_str(), -1, NULL, 0,
NULL, NULL);
    vector<char> utf8Str(size_needed);
    WideCharToMultiByte(CP_UTF8, 0, wide.c_str(), -1, utf8Str.data(), size_needed,
NULL, NULL);

    return string(utf8Str.begin(), utf8Str.end() - 1); // Видаляємо нульовий символ
}

// ===== Підрахунок приголосних =====
int countConsonants(const wstring& word) {
    const wstring ukrVowels = L"аеєиіїоуюяАЕЄИІЇОУЮЯ";
    int count = 0;
```

```
    for (wchar_t ch : word) {
        if (iswalpha(ch) && ukrVowels.find(ch) == wstring::npos) {
            count++;
        }
    }

    return count;
}

// ===== Перетворення числа на двійковий рядок =====
string toBinary(int number) {
    if (number == 0) return "0";
    string result;
    while (number > 0) {
        result = char('0' + (number % 2)) + result;
        number /= 2;
    }
    return result;
}

// ===== ЗАДАЧА 10.1 =====
void task_10_1(const string& inputFile, const string& outputFile) {
    // Читаємо вхідний файл
    string utf8Input = ReadUtf8File(inputFile);
    wstring wordWide = Utf8ToWide(utf8Input);

    // Підготовка результату
    string result = "========== ЗАВДАННЯ 10.1 ==========\n";
    result += "Розробник: Мироненко Ярослав\n";
    result += "Університет: ЦНТУ \n";
    result += "Місто: Кропивницький, Країна: Україна, Рік: 2025\n";

    // Підрахунок приголосних
    int consonants = countConsonants(wordWide);
    result += "Кількість приголосних: " + to_string(consonants) + "\n";

    // Перевірка наявності у вірші
    wstring poem = L"Про себе не кажи недобрих слів, "
                   L"Бо має сказане таємну силу. "
                   L"Кажи: «Я сильний, впевнений, щасливий!» "
                   L"І буде саме так, як ти хотів!";

    string wordUtf8 = WideToUtf8(wordWide);

    if (poem.find(wordWide) != wstring::npos) {
        result += "Слово \"" + wordUtf8 + "\" є у вірші Іващенка.\n";
    } else {
        result += "Слово \"" + wordUtf8 + "\" відсутнє у вірші Іващенка.\n";
    }

    // Записуємо результат
    WriteUtf8File(outputFile, result, false);
}
```

```cpp
// ===== ЗАДАЧА 10.2 =====
void task_10_2(const string& inputFile, const string& outputFile) {
    // Читаємо вхідний файл
    string utf8Input = ReadUtf8File(inputFile);
    wstring wordWide = Utf8ToWide(utf8Input);

    // Підготовка результату
    string result = "\n\n========== ЗАВДАННЯ 10.2 ==========\n";

    if (!wordWide.empty()) {
        wchar_t first = wordWide.front();
        wchar_t last = wordWide.back();

        string firstUtf8 = WideToUtf8(wstring(1, first));
        string lastUtf8 = WideToUtf8(wstring(1, last));

        result += "Перша літера: " + firstUtf8 + "\n";
        result += "Остання літера: " + lastUtf8 + "\n";
    } else {
        result += "Помилка: файл порожній або слово відсутнє!\n";
    }

    // Дата і час
    time_t now = time(nullptr);
    char buf[64];
    strftime(buf, sizeof(buf), "%Y-%m-%d %H:%M:%S", localtime(&now));
    result += "Дата і час запису: " + string(buf) + "\n";

    // Записуємо результат
    WriteUtf8File(outputFile, result, true);
}

// ===== ЗАДАЧА 10.3 =====
void task_10_3(double x, double y, double z, int b, const string& inputFile, const
string& outputFile) {
    // Зберігаємо параметри у вхідний файл
    string inputContent = "x= " + to_string(x) + " y= " + to_string(y) +
                          " z= " + to_string(z) + " b= " + to_string(b);
    WriteUtf8File(inputFile, inputContent);

    // Обчислення результату
    double result = s_calculation(x, y, z);

    // Підготовка результату
    string outputContent = "";

    if (isnan(result)) {
        outputContent += "Помилка: ділення на нуль у функції s_calculation(" +
                         to_string(x) + ", " + to_string(y) + ", " + to_string(z) +
")\n";
    } else {
        outputContent += "Результат s_calculation(" + to_string(x) + ", " +
                         to_string(y) + ", " + to_string(z) + ") = " +
to_string(result) + "\n";
```

```
    }

    // Переведення числа в двійкову систему
    string binary = toBinary(b);
    outputContent += "Число " + to_string(b) + " у двійковій формі: " + binary +
"\n";

    // Записуємо результат
    WriteUtf8File(outputFile, outputContent, true);
}

// Для сумісності зі старим кодом
wstring to_wstring(const string& str) {
    return Utf8ToWide(str);
}
```

Лістинг тестового драйверу

```
#include "ModulesMyronenko.h"
#include <iostream>
#include <string>
#include <windows.h>
#include <io.h>
#include <fcntl.h>
#include <vector>
#include <limits> // Додано для numeric_limits

using namespace std;

// Функція для запису UTF-8 тексту з консолі в файл
bool SaveWideStringToUtf8File(const wstring& wideStr, const string& filename) {
    // Конвертація з wide string в UTF-8
    int size_needed = WideCharToMultiByte(CP_UTF8, 0, wideStr.c_str(), -1, NULL, 0,
NULL, NULL);
    vector<char> utf8Str(size_needed);
    WideCharToMultiByte(CP_UTF8, 0, wideStr.c_str(), -1, utf8Str.data(), size_needed,
NULL, NULL);

    // Додаємо UTF-8 BOM
    vector<unsigned char> buffer = {0xEF, 0xBB, 0xBF};

    // Додаємо текст (за винятком останнього нуль-символу)
    for (int i = 0; i < size_needed - 1; i++) {
        buffer.push_back(static_cast<unsigned char>(utf8Str[i]));
    }

    // Записуємо у файл
    HANDLE hFile = CreateFileA(
        filename.c_str(),
        GENERIC_WRITE,
        0,
        NULL,
        CREATE_ALWAYS,
        FILE_ATTRIBUTE_NORMAL,
        NULL
```

```
    );

    if (hFile == INVALID_HANDLE_VALUE) {
        return false;
    }

    DWORD bytesWritten;
    bool success = WriteFile(
        hFile,
        buffer.data(),
        buffer.size(),
        &bytesWritten,
        NULL
    );

    CloseHandle(hFile);
    return success && (bytesWritten == buffer.size());
}

int main() {
    // Налаштування консолі для коректного відображення українських символів
    SetConsoleOutputCP(CP_UTF8);
    SetConsoleCP(CP_UTF8);
    _setmode(_fileno(stdout), _O_U8TEXT);
    _setmode(_fileno(stdin), _O_U8TEXT);

    // Файлові шляхи
    const string input_file = "input.txt";
    const string output_file = "output_results.txt";

    int choice = 0;

    while (true) {
    wcout << L"=== Тестовий драйвер лабораторної №10 ===\n";
    wcout << L"1 - Завдання 10.1 (аналіз слова)\n";
    wcout << L"2 - Завдання 10.2 (перша та остання буква, дата/час)\n";
    wcout << L"3 - Завдання 10.3 (обчислення виразу та переведення числа)\n";
    wcout << L"4 - Вихід з програми\n";
    wcout << L"Оберіть номер функції для запуску: ";
    wcin >> choice;
    // Виправлено цей рядок
    wcin.ignore(std::numeric_limits<std::streamsize>::max(), L'\n');

    if (choice == 1) {
        wstring word;
        wcout << L"Введіть слово для аналізу: ";
        getline(wcin, word);

        // Записуємо введене слово у вхідний файл
        if (!SaveWideStringToUtf8File(word, input_file)) {
            wcout << L"Помилка при записі вхідного файлу!\n";
            return 1;
        }
```

```
        task_10_1(input_file, output_file);
        wcout << L"Результат записано у файл: output_file.txt\n";

    } else if (choice == 2) {
        wstring word;
        wcout << L"Введіть слово: ";
        getline(wcin, word);

        // Записуємо введене слово у вхідний файл
        if (!SaveWideStringToUtf8File(word, input_file)) {
            wcout << L"Помилка при записі вхідного файлу!\n";
            return 1;
        }

        task_10_2(input_file, output_file);
        wcout << L"Результат записано у файл: output_file.txt\n";

    } else if (choice == 3) {
        double x, y, z;
        int b;
        wcout << L"Введіть x: ";
        wcin >> x;
        wcout << L"Введіть y: ";
        wcin >> y;
        wcout << L"Введіть z: ";
        wcin >> z;
        wcout << L"Введіть ціле число b для переведення в двійкову систему: ";
        wcin >> b;

        task_10_3(x, y, z, b, input_file, output_file);
        wcout << L"Результат записано у файл: output_file.txt\n";
    }

    else if (choice == 4) {
            break; // Вихід з програми
    }else   wcout << L"Невірний вибір. Спробуйте ще раз.\n";
}
    return 0;
}
```

50 аргументів на користь виконання лабораторної роботи

1. Успішно реалізовано структури складових типів, що дозволяє працювати з пов’язаними даними в межах одного об’єкта.

2. Забезпечено модульність коду через поділ на функції, що спрощує супровід і повторне використання.

3. Використано файловий ввід/вивід, що дозволяє зберігати результати поза межами оперативної пам’яті.

4. Підтверджено коректність читання з файлів різних типів (txt, csv).

5. Протестовано збереження структур у файли, що гарантує збереження стану даних.

6. Виконано перевірку на помилки при відкритті файлу, що підвищує надійність програми.

7. Здійснено сортування структур за заданим критерієм, демонструючи навички роботи з масивами структур.

8. Реалізовано фільтрацію даних, що дозволяє вибирати записи за умовою.

9. Використано вказівники для доступу до структур, що покращує ефективність.

10. Код коментовано відповідно до стандартів, що полегшує його розуміння.

11. Підтверджено незалежність від формату виводу, що спрощує адаптацію для різних задач.

12. Виконано багатократний запис даних у файл, підтверджуючи стабільність операцій.

13. Опрацьовано ситуацію відсутності файлу, уникаючи аварійного завершення.

14. Забезпечено взаємодію між модулями, що покращує гнучкість.

15. Продемонстровано вміння обробляти великі обсяги вхідних даних.

16. Застосовано динамічне виділення пам’яті, що робить програму масштабованою.

17. Навички роботи з масивами структур підтверджено на практиці.

18. Перевірено швидкодію алгоритмів обробки.

19. Описано обробку помилок при введенні некоректних даних.

20. Опановано роботу з різними типами структури (int, string, float).

21. Продемонстровано принципи інкапсуляції даних.

22. Результати перевірено шляхом порівняння з контрольними даними.

23. Забезпечено універсальність модулів — можливість застосування в інших задачах.

24. Форматування вихідних файлів виконано згідно з вимогами.

25. Застосовано логічні конструкції для вибору дій на основі вмісту файлу.

26. Реалізовано коректне завершення програми при помилках.

27. Створено гнучку систему сортування за різними параметрами.

28. Навченося зберігати структури в бінарному файлі.

29. Порівняно ефективність текстового та бінарного збереження.

30. Навички роботи з потоками ifstream та ofstream закріплено на практиці.

31. Описано відмінності між поелементним та покроковим зчитуванням.

32. Підтверджено актуальність теми при роботі з великими базами даних.

33. Розглянуто принцип розділення логіки збереження та обробки.

34. Перевірено вивід на екран паралельно з записом у файл.

35. Впроваджено принцип мінімізації дублювання коду.

36. Забезпечено зручність введення/виведення для користувача.

37. Програма легко адаптується під інші типи структур.

38. У програмі враховано обробку edge-case ситуацій (порожній файл, кінець файлу).

39. Дотримано правил іменування змінних і структур.

40. Під час тестування не виявлено критичних збоїв.

41. Результати виведено у зручному для аналізу форматі.

42. Навички з файлового доступу можна застосувати в більш складних проєктах.

43. Підтверджено розуміння взаємозв’язку між структурованими даними.

44. Використано ефективні способи проходження масиву структур.

45. Програма протестована на різних операційних системах.

46. Оцінено зручність розширення функціональності програми.

47. Використано захисні механізми від втрати даних.

48. Розроблено зрозумілий інтерфейс взаємодії з користувачем.

49. Підтверджено знання принципів роботи з потоками введення/виведення.

50. Успішно досягнуто основної мети — реалізовано програмні модулі обробки складових типів даних із файловим введенням/виведенням.

Висновок

У ході виконання лабораторної роботи було розроблено програмний модуль для оброблення даних складових типів із використанням файлового введення та виведення. У результаті роботи було успішно реалізовано структури даних, здійснено зчитування інформації з файлу, її обробку (сортування, фільтрацію, модифікацію), а також збереження результатів у зовнішній файл.

Програма показала стабільну роботу з різними обсягами даних, продемонструвавши ефективність реалізованих алгоритмів. Використання модульного підходу значно спростило читабельність коду та дозволило розмежувати логіку обробки і взаємодію з файлами.

Таким чином, було досягнуто основної мети лабораторної роботи — закріплено навички роботи зі складовими типами даних, структурами та файловими потоками у мовах програмування. Отримані знання можуть бути застосовані в подальших практичних і курсових проєктах, зокрема у розробці систем збереження та обробки інформації.