МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА МЕХАНІКО-МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ Кафедра обчислювальної математики

ЗВІТ з лабораторної роботи з дисципліни «Програмування»

на тему: «Розробка бібліотеки на C та класу на C++ для роботи з форматом CSV» (Завдання №16)

Виконав: студент 2 курсу групи «Комп’ютерна математика» Ульянов Максим

Київ – 2025

1. Опис формату CSV

CSV (Comma-Separated Values — значення, розділені комою) — це текстовий формат, призначений для представлення табличних даних. Цей формат є одним з найпоширеніших засобів обміну даними між різними програмами, оскільки він простий для читання як людиною, так і комп'ютером.

Основні правила формату:

1. **Структура рядків:** Кожен рядок файлу відповідає одному рядку таблиці. Розділення рядків здійснюється стандартними символами перенесення рядка (CRLF або LF).
2. **Роздільники:** Значення в межах одного рядка розділяються спеціальним символом. Хоча назва формату передбачає кому (`,`), на практиці часто використовуються крапка з комою (`;`) або табуляція (`\t`). Це особливо актуально для локалізацій, де кома використовується як десятковий роздільник у числах.
3. **Заголовки:** Перший рядок файлу часто містить заголовки стовпців, що дозволяє ідентифікувати зміст полів.
4. **Екранування:** Якщо значення містить зарезервовані символи (кому, лапки або перенесення рядка), воно повинно бути взяте в подвійні лапки.

У рамках даної роботи реалізовано підтримку базового формату CSV з можливістю динамічної зміни роздільника.

2. Шляхи роботи з CSV у мові C++

Стандартна бібліотека C++ (STL) не надає спеціалізованих класів для автоматичного парсингу CSV тому використовують:

A. Використання потоків введення-виведення (`<fstream>`) Це базовий метод, який використовується у даній роботі.

- Файл відкривається через `std::ifstream`.
- Читання відбувається порядково за допомогою функції `std::getline`, яка дозволяє вказати символ-роздільник рядків.

- Отриманий рядок передається у `std::stringstream`, звідки за допомогою `std::getline` (з вказанням роздільника полів, наприклад, коми) витягуються окремі значення клітинок.

Перевагою цього методу є відсутність залежностей від сторонніх бібліотек.

Б. C-style I/O (`<cstdio>`) Використання функцій `fopen`, `fgets`, `strtok` або `scanf`. Цей метод є швидшим, але менш безпечним з точки зору управління пам'яттю та обробки рядків (`std::string`).

В. Сторонні бібліотеки Існують готові бібліотеки (наприклад, *RapidCSV* або *csv-parser*), які забезпечують високу швидкість та обробку складних випадків (наприклад, вкладених лапок).

3. Опис розробленої програми

Програмний комплекс складається з двох частин: бібліотеки функцій на мові C та класу на мові C++. Обидві частини реалізують ідентичний функціонал та перевіряються крос-тестом.

3.1. Бібліотека на C (CSV_C)

Реалізація базується на структурі `CSVTable`, яка містить:

- `char ***data` — тривимірний масив для зберігання значень клітинок.
- `rows`, `cols` — розмірність таблиці.
- `delimiter` — символ роздільника.
- `headers` — масив заголовків (якщо вони увімкнені).

Управління пам'яттю здійснюється вручну:

- Функція `csv_create` виділяє пам'ять під структуру.
- `csv_read_file` читає файл, динамічно розширюючи масив за допомогою `realloc`.
- Функції `csv_add_row` та `csv_add_column` дозволяють змінювати розмір таблиці.
- `csv_free` коректно звільняє всю виділену пам'ять, щоб уникнути витоків (memory leaks).

3.2. Клас на C++ (CSV_CPP)

Реалізація на C++ використовує об'єктно-орієнтований підхід. Клас `CSVHandler` інкапсулює дані у контейнері `std::vector<std::vector<std::string>>`, що автоматизує управління пам'яттю.

Ключові особливості реалізації:

- **Ітератор:** Згідно з завданням, реалізовано власний вкладений клас ітератора `CSVIterator`. Це дозволяє проходити по рядках таблиці у зручному циклі `for`.
- **Методи маніпуляції:** Реалізовано методи `setField`, `deleteRow`, `addRow` та `addColumn`, які дозволяють редагувати завантажені дані.
- **Поліморфізм:** Метод `getField` перевантажено для доступу до даних як за числовим індексом стовпця, так і за його назвою.

3.3. Тестування

Для перевірки коректності роботи реалізовано:

1. **Драйверні тести:** Файли `CSV_test.c` та `CSV_test.cpp`, які містять інтерактивне меню для перевірки всіх функцій (читання, запис, редагування, зміна роздільника).
2. **Крос-тест:** Файл `CSV_cross_test.cpp`, який одночасно використовує C-бібліотеку та C++ клас для читання одного файлу, порівнюючи результати. Це гарантує ідентичність алгоритмів в обох реалізаціях.