1. **Опис формату CSV.**

CSV (Comma Separated Values) - це файл значень, розділених комами, який дозволяє зберігати дані в табличному форматі. Ці файли іноді можуть називатися файлами з роздільниками-символами або файлами з роздільниками-комами. В основному вони використовують кому для поділу даних, але іноді використовують інші символи, наприклад, крапки з комою (в нашому випадку). Ідея полягає в тому, що ви можете експортувати складні дані з однієї програми до файлу .csv, а потім імпортувати дані з цього файлу .csv в іншу програму.

Щоб переглянути вміст CSV у Блокноті, клацніть на файл правою кнопкою миші та виберіть команду «Змінити» («Edit», «Изменить»). У Блокнота можуть виникнути проблеми з відкриттям файлу CSV, якщо він дуже великий. Тому також можна відкривати файли CSV у програмах для роботи з електронними таблицями, що спрощує їх читання. Наприклад, якщо на комп’ютері встановлено Microsoft Excel, ви можете двічі клацнути файл .csv, щоб відкрити його в Excel за замовчуванням. Якщо він не відкривається в Excel, ви можете клацнути файл .csv правою кнопкою миші і вибрати «Відкрити з допомогою» > («Open With», «Открыть с помощью») «Excel».

Зберігати CSV файли відносно просто: лише потрібно знати, де змінити тип файлу. В «Excel» можна зробити так: «Файл» → «Зберегти як» → змінити тип файлу на .csv (з роздільниками-комами).

1. **CSV та C++. Способи відкриття на C++**

Читання CSV файлу по суті аналогічно читанню текстових файлів на C++. Різниця полягає у способі вилучення значень із рядків.

Відкрити файл можна за допомогою бібліотеки «fstream». Потім ми читаємо файл рядок за рядком, використовуючи getline() метод, оскільки кожен рядок закінчується символом нового рядка. Метод getline() також приймає необов’язковий аргумент delimitor. Якщо ми передаємо роздільник як вхідний аргумент, він читатиме файл доти, доки роздільник не буде знайдений. Після вилучення рядка з файлу ми будемо використовувати stringstream для вилучення слів, розділених у рядку коми. Третім параметром в getline() поставимо крапку з комою ‘;’. Тому ми зможемо вилучити всі слова із рядка та зберегти їх значення у векторі чи масиві.

1. **Опис власної програми на C та C++**

* ***Програма на C***

На С реалізована бібліотека “CSV.h” для використання та редагування файлів формату .csv. При бажанні використати цю бібліотеку спочатку обов'язково треба викликати функцію start(), в аргументи якої треба передати файл з яким хочете працювати. Для зручного користування можна використати функцію writeToAnotherFile(), яка дозволяє переписати дані з одного файлу в інший. Оскільки введений в функцію start() файл буде змінено, для повторного тестування/перевірки основний файл краще не змінювати, а перезаписувати в допоміжний. Всі дані з файлу записуються в масив рядків ARR\_CSV. Це зроблено для того, щоб не відкривати файл кожен раз при читанні. Для будь-якої функції, що змінює якимось чином вміст файлу, після виконання основної задачі викликається функція file\_write(), яка записує дані з масиву в файл.

Далі розглянемо функції, які безпосередньо будуть використовуватися для роботи із файлами.

Функції для роботи з рядками:

* changeRow(char filename[], char string[], number) – змінює рядок під номером number (від 1 до кількості рядків) на заданий рядок string. Сам рядок, на який замінюємо, повинен містити слова із роздільником ‘;’ після кожного слова, окрім останнього.
* removeRow(char filename[], int number) – видаляє рядок під номером number. При цьому видалити рядок, якщо файл пустий, буде неможливо.
* rowToEOF(char filename[], char string[]) – додає рядок до кінця файлу. Вхідний параметр string має такий самий вид, як і для функції changeRow
* printRowByNumber(int n) – виводить значення рядка в консоль.

Функції для роботи із колонками:

* changeColumn(char filename[], char \*\*columns, int n) – замінює колонку під номером n на колонку, що міститься в масиві рядків columns.
* changeColumnByName(char filename[], char \*columns[], char name[]) – замінює колонку під іменем name на колонку columns. Під іменем колонки мається на увазі слово з відповідною назвою з першого рядка файлу.
* removeColumnByNumber(char filename[], int n) – видаляє колонку за номером. Видалити колонку, якщо файл буде пустим також неможливо.
* removeColumnByName(char filename[], char name[]) – видаляє колонку за назвою name.
* changeGeneralFieldName(char filename[], char name[], int n) – змінює назву колонки під номером number. Під значенням назви/імені колонки мається на увазі те ж саме, що і для функції changeColumnByName()
* appendColumn(char filename[], char \*column[]) – додає колонку в кінець файлу.
* printFieldByNumber(int n) – виводить значення всієї колонки під номером n

Функції для роботи з полями (під полем мається на увазі будь-яка клітинка, АЛЕ не з першого рядка):

* changeSquareName(char filename[], char name[], int row, int col) – змінює назву поля під номером (row, col), якщо рахувати від лівого верхнього кута.
* printSquareByNumber(int n, int m) – виводить значення поля рядка n під номером m

Допоміжні функції(викликаються тількі в файлі CSV.c. Не використовувати):

* file\_write(char filename[]) – записує масив ARR\_CSV в файл
* fill\_arr(char filename[]) – заповнює масив з файлу
* get\_size(char filename[]) – знаходить розмір файлу (кількість колонок та рядків)
* getfield(char\* line, int num) – знаходить слово під номером num в рядку
* concat(char \*s1, char \*s2) – з’єднує два рядки
* replace\_word(const char\* s, const char\* oldW, const char\* newW) – замінює входження слова oldW на newW в рядку s
* remove\_substring(char \*str, char \*sub) – видаляє входження слова в рядку

НЕ зробив можливість редагувати тип роздільника. За домовленістю із самим собою роздільник крапка з комою.

* ***Програма на С++***

На С++ виконано дві варіації програми: в першій використовується динамічний масив для зберігання вмісту файлу, а в другій – контейнер vector. Функціонал не відрізняється, окрім методів роботи із масивами та векторами відповідно.

Створений клас csvFile також робить все те, що і програма на С. Також для зручності в конструктор передається два файли: один з інформацією, а інший порожній. Вся інформація перезаписується в порожній; надалі всі зміни будуть проходити лише в цьому новому файлі.

Методи для роботи з рядками:

* row\_by\_number(int number) – виводить в консоль рядок під номером number
* delete\_row(int number) – видаляє рядок
* change\_row(string line, int number) – змінює рядок під відповідним номером на рядок string. Рядок string повинен мати такий самий вигляд як і на С. (Слова з роздільником крапка з комою. Після останнього слова роздільника немає)
* append\_row(string line) – додає рядок в кінець файлу.

Методи для роботи з колонками:

* read\_field\_by\_name(string name) – виводить колонку даних за іменем. Під ім’ям мається на увазі значення окремих полів першого рядку.
* change\_field(string \*column, int num) – змінює колонку за номером на нову зі значеннями з масиву column
* append\_field(string \*column) – додає колонку справа в кінець файлу
* delete\_field(int number) – видаляє колонку за номером

Інші методи:

* change\_data() – приватний метод. Перезаписує дані с масиву/вектору в файл.
* print\_data() – виводить в консоль значення. При роботі з масивами використовується вкладені цикли. В реалізації з вектором – використовується ітератор.
* change\_square(string name, int row, int col) – змінює значення поля.
* read\_square(int row, int col) – виводить значення поля (будь-якого).
* read\_current\_row() – читає поточний рядок
* find\_size() – знаходить розмір файлу (кількість рядків та колонок)
* delete\_arr(string \*\*arr, int n) та make\_arr() – наявні лише в реалізації із динамічним масивом. Відповідно видаляють значення з масиву та створюють новий динамічний двовимірний масив.

Програми на С та С++ роблять те ж саме. На С++ також відсутня реалізація зміни роздільника для файлів .csv. Наведені вище методи є наявними в реалізації з динамічними масивами. У версії з векторами аргументи функції, де подавалися масиви, відповідно стануть векторами.

Тестування бібліотеки та класу зручно робити за допомогою файлу tests.txt. В ньому розписані в зрозумілій формі як можна ввести нові значення для тестів. І вводити значення тестів можна саме в цей файл. В файлі CSV\_cross\_test.cpp одночасно з одного файла беруться тестові виклики функцій для бібліотеки на С та класу з реалізацією вектора на С++. Знову ж таки – реалізація з вектором робить те ж саме, що і реалізація за допомогою динамічних масивів.