Project Report: ANSI Terminal-Based Spreadsheet Program

Overview

The project aims to create a simplified ANSI terminal-based spreadsheet program, inspired by VisiCalc, with features like basic arithmetic calculations, cell referencing, formulas, and CSV file handling. The program uses an AnsiTerminal class for displaying content and capturing user input. The objective is to demonstrate knowledge of C++ fundamentals, Object-Oriented Programming (OOP), Standard Template Library (STL) usage, and file handling.

UML Class Diagram

A UML class diagram representing the relationships between key classes in the spreadsheet application is shown below (a diagram generated with PlantUML). This diagram includes classes such as Table, Cell, FormulaParser, FileManager, SpreadsheetApp, and AnsiTerminal.

Table: Manages a 2D array of Cell objects and supports formula parsing, cell referencing, and automatic recalculation.
Cell: Represents individual cells in the spreadsheet, which can store numbers, strings, or formulas.
FormulaParser: Parses and evaluates formulas, handling operations like SUM, AVER, STDDEV, MAX, and MIN.
FileManager: Handles file operations such as saving, loading, and creating new CSV files.
SpreadsheetApp: The main application class that integrates all functionalities and provides a user interface.
AnsiTerminal: A utility class that manages terminal display and user input.
Features Implemented
Basic Calculations and Formulas:

Users can enter numeric values and formulas (e.g., =A1 + B2 - C3) in cells.
Supported operations include addition, subtraction, multiplication, and division.
The program performs automatic recalculation of dependent cells whenever a cell value changes.
Automatic Recalculation:

Changes to cell values trigger automatic updates in any dependent cells. For example, if B2 = A1 + 5, changing A1 will automatically recalculate B2.
Column and Row Labels:

Rows are labeled numerically (1, 2, 3, ...) and columns alphabetically (A, B, C, ..., AA, AB, ...), similar to typical spreadsheet applications.
Cell Referencing:

Users can reference other cells within formulas using cell identifiers like A1 + B1.
The program supports absolute cell referencing.
The program includes functions like SUM, AVER, STDDEV, MAX, and MIN to calculate ranges of cells (e.g., SUM(A1..A10)).
Visual Interface:

The user interface displays the contents of the currently selected cell on the top line, allowing users to input commands and formulas.
File Operations (CSV Format):

Users can save and load spreadsheet data in CSV format. The FileManager class handles these operations, ensuring compatibility with external spreadsheet applications like Microsoft Excel.
Data Storage:

A 2D std::vector is used to store the values of the spreadsheet grid. Each cell contains either a number, string, or formula.
Features Not Implemented
Cyclic References: The program does not address cyclic references in formulas. This is considered an advanced feature and was outside the scope of the project.
AI-Based Assistance
Throughout the development of the project, I utilized AI-based tools to assist with certain aspects of the code:

FormulaParser:

I leveraged AI assistance to improve the formula parsing logic, specifically in evaluating cell references and handling different arithmetic operations. The AI provided insights into efficiently parsing formulas and computing their values.
Error Handling:

I received assistance with implementing proper error handling for formula parsing, especially for cases where invalid input or incorrect formulas are entered. The AI suggested appropriate ways to handle errors and provide feedback to users.
User Manual
Running the Program
Upon running the program, the user is prompted with the interface displaying the spreadsheet grid. The top line will show the contents of the currently selected cell.
Users can navigate between cells using arrow keys and input data directly into the selected cell.
To enter a formula in a cell, the user can type it in the format =A1 + B2, where A1 and B2 are references to other cells.
After entering data or formulas, the spreadsheet will automatically recalculate dependent cells.
Example Commands
Entering a number: Type 5 into a cell.
Entering a string: Type "Hello" into a cell.
Entering a formula: Type =A1 + B2 in a cell.
Summing a range: Type =SUM(A1..A10) to sum values from A1 to A10.
Conclusion
This project successfully implements a simplified ANSI terminal-based spreadsheet program that supports basic arithmetic calculations, formula parsing, and file operations in CSV format. By utilizing C++ features such as OOP, STL, and file handling, the program offers a functional and efficient solution for spreadsheet management. Additionally, AI

assistance was valuable in improving the formula parsing and error handling portions of the project.