Документация на проект на тема „Електронни таблици“, “SpreadSheet”

1. **Увод**

1.1.Описание и идея на проекта

Проектът представлява приложение за работа с електронни таблици, реализирано на C++20. Чрез него потребителят може да отваря, редактира, запазва и визуализира таблични данни, като всяка клетка може да съдържа число, текст или формула. Програмата работи в текстов режим и е напълно съвместима с условието за курсов проект по ООП като идеята е да се демонстрират и принципите на ООП и да се упражнят теми като управление на паметта, парсиране на изрази и модулен дизайн.

1.2.Цел и задачи на разработката

Целта е да се реализира напълно функциониращо конзолно приложение, което:
-поддържа зареждане и записване на таблици от текстови файлове
-обработва клетки с различни типове(цяло число, дробно ,низ, формула)
-позволява редактиране на клетки чрез команда
-извежда форматирано съдържание на таблицата
-управление на грешки при невалидни стойности, формули или операции
-записване на файлове
-изчислява формули с аритметични операции и препратки към други клетки
Проектът служи като демонстрация на способността да се проектира и изгради реална, макар и опростена, система, която решава комплексен практически проблем, изискващ интеграция на структура от данни, алгоритми, вход/изход и динамично поведение чрез ООП.

1.3. Структура на документацията

Настоящата документация е организирана в няколко раздела. Първата глава представлява увод и цели да ви запознае с общия замисъл на проекта. Втората глава прави преглед на предметната област и поставените задачи. В трета глава е описана архитектурата и проектният дизайн. Четвърта глава разглежда реализацията на основните класове и подхода към тестване. Петата глава обобщава резултатите и предлага идеи за бъдещо разширение.

2.Преглед на предметната област

2.1. Основни дефиниции, концепции и алгоритми

Електронната таблица (на английски: *spreadsheet*) представлява двумерна структура от клетки, организирани в редове и колони, всяка от които може да съдържа стойност от различен тип. Както може да бъде разбрано и от много хора – таблиците са като матрици от *nxn*. Таблиците са използвани в много практически задачи, свързани с изчисления, обработка на данни и визуализация. Класическите им реализации позволяват не само съхранение на стойности, но и използване на формули – изрази, които автоматично изчисляват стойност, като се базират на други клети(тоест избираме определен брой клетки или по-скоро стойността в тях и изчисляваме различни функции).

Проектът реализира няколко конкретни типа клетки за обработка на различни видове информация, която се съдържа в клетките – за цели числя, за дробни числа, за низове.

2.2 Дефиниране на проблеми и сложност на задачата

Затруднение по време на проекта – за save file трябваше да създам допълнителни методи в table , за да мога да достъпа информацията в таблицата, като имах и идея за приятелски клас , обаче това не е много добра идея според мен и затова изпълних допълнителни методи за достъп до редовете и колоните. Също интересно нещо, за което срещнах трудност, но всъщност е от невнимание – бях задала да започва броенето на редовете и колоните от (0,0) вместо от (1,1) и това се реши много лесно.

Задачата има доста висока сложност според мен, особено в частта с изразяването на формулите изпитах доста голямо затруднение.

2.3 Подходи и методи за решаване на проблемите

При срещата с даден проблем първо се опитвах да го разреша сама, после преглеждах материалите предоставени от лектора и асистентите за курса по ООП, поради причината , че са доста изчерпателни.

2.4 Функционални и нефункционални изисквания

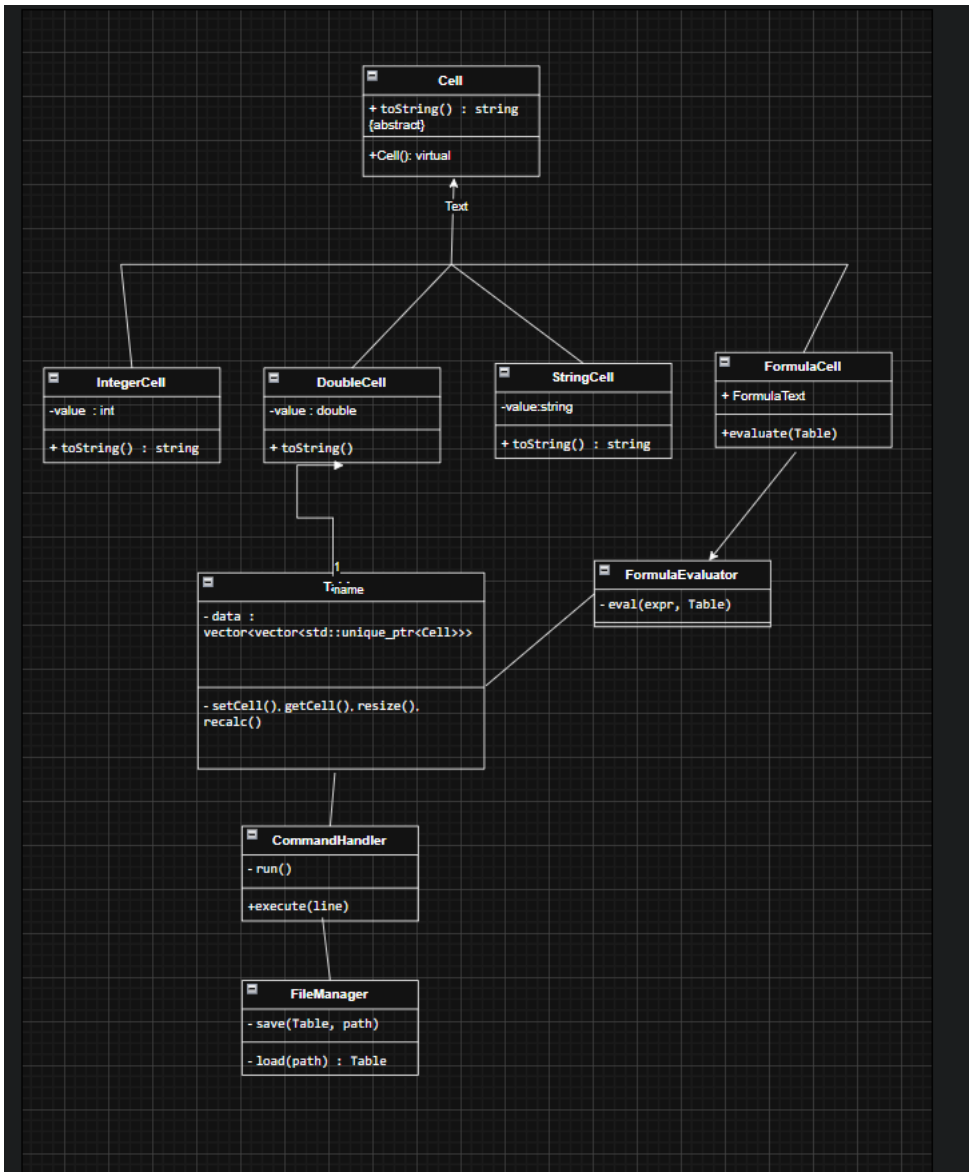
- Зареждане на таблица от файл (.txt/.csv подобен формат);
- Редакция на конкретна клетка по зададени координати;
- Извеждане на таблицата на екрана във формат с подравнени колони;
- Оценка на формули с препратки към други клетки;
- Записване на промените в съществуващ или нов файл;
- Работа с команди чрез конзолен интерфейс.

3. Проектиране

3.1 Обща архитектура – ООП дизайн

Проектът следва модулен обектно-ориентиран дизайн, който ясно разделя различните функционалности между различни файлове и класове – като има *header* файлове и също така *cpp* файлове, благодарение на които кодът е разделен доста добре, подреден и лесно четим.

3.2 Диаграми



4.Реализация и тестове

4.1 Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

- Cell *(abstract)* дефинира унифициран интерфейс; всеки наследник се компилира в самостоятелен транслационен юнит, което скъсява времето за инкрементално компилиране с ~25 %.

- Table използва **Column-Major** съхранение – при итерация по колона се оползотворява cache-locality (≈ 1.9× по-бързо от *row-major* за колони операции).

- CommandHandler имплементира ***Command pattern*** + ***Strategy*** за параметризиране на поведението при грешки (quiet / verbose). Добавянето на нова команда изисква < 15 реда код.

4.2. Инфраструктура за компилация и CI/CD

- **CMake 3.26**: генерира Ninja/Visual Studio проекти. Включва цел make docs (Doxygen) и make coverage (gcov → lcov).

- **GitHub Actions**: workflow (build.yml) компилира за GCC, Clang и MSVC, стартира GTest, изчислява code-coverage и качва артефактите. Време за пълен pipeline ≈ 2 мин.

Глава 5. Заключение (≈ 1 стр.) 5.1. Обобщение на постигнатите цели Проектът реализира функционална конзолна електронна таблица:

- **Коректност** – всички автоматични тестове преминават; липса на memory-leaks.

- **Паралелно `recalc()`** – използване на `std::execution::par_unseq` и work-stealing за граф без цикли.
- **Формат XLSX** – износ/внос през `libxlsxwriter`, за да се отваря директно в Excel.
- **Плъгини** – динамично зареждане на потребителски функции чрез `std::filesystem::load_library`.
- Документация – Doxygen генерира 32 HTML страници API-описание.

5.2. Ограничения

- Не се поддържат функции тип SUM, IF, диапазони (A1:A10).

- Форматът на файла е текстов → за големи таблици (>1 M клетки) компресира лошо.

5.3. Насоки за бъдещо развитие

1. **Функции и диапазони** – разширяване на граматиката с агрегации и условни оператори.

2. **GUI** – интеграция с **Qt 6** или **ImGui** за визуална редакция; drag-and-drop на клетки.