**Увод**

Електронните таблици са изключително полезно средство за обработка на информация. Целта на този проект е да позволи за лесна и ефективна работа с електронни таблици от страна на потребителя.

Основните цели, които биват реализирани от проекта, са следните:

* Лесен за ползване и надежден интерфейс, който да позволява използването на реализираните функционалности.
* Възможност за отваряне на вече съществуващи таблични файлове и съответно запаметяване на таблици под формата на табличен файл.
* Проверки за коректността на въведените от потребителя данни и извеждане на подходящи съобщения в случай на нередност.
* Ефективно и лесно за навигация представяне на таблиците в паметта на компютъра.
* Възможност дадена клетка в таблицата да приема уникален тип и да има различни функционалности и вид съдържание в зависимост от типа си.
* Възможност за промяна на съдържанието на клетки в таблицата и на отмяна на вече направени направени промени.
* Възможност за отпечатване на съдържанието на таблицата чрез добре форматиран изход.
* Възможност за изчисление на стойност на клетки от тип “формула” като се позволява в дадена формула да има препратки към други клетки от таблицата.

Документацията е разделена на следните части:

1. Увод
2. Преглед на предметната област
3. Обща архитектура
4. Реализация
5. Заключение

**Преглед на предметната област**

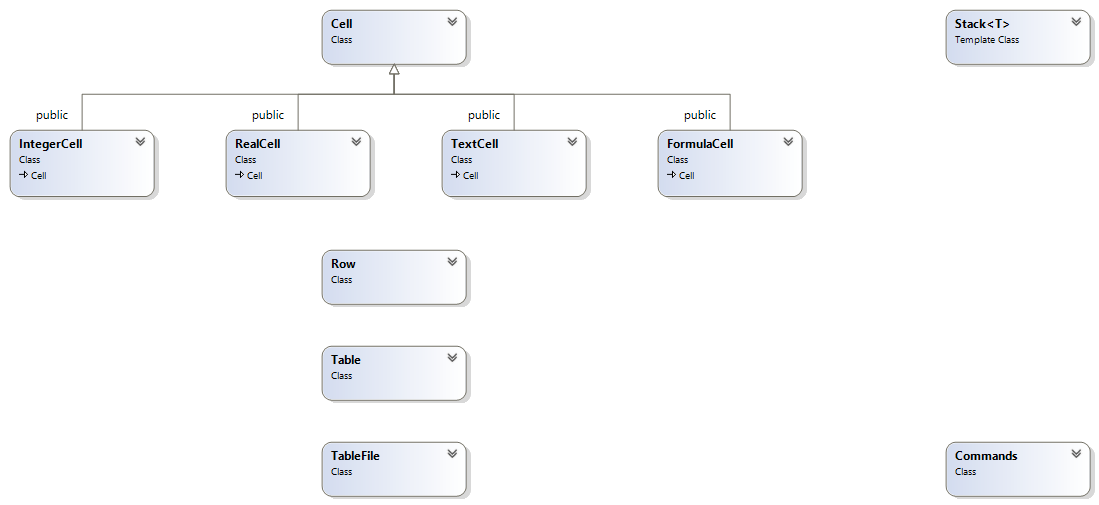
Следните концепции биват използвани в реализацията на проекта:

1. Методи
   * (Косвена) рекурсия - необходима при реализирането на някои от алгоритмите.
   * Динамично заделяне на памет - необходимо при реализацията на отделните класове и структури от данни, а също така и при работа с низове.
   * Динамично свързване - необходимо при реализацията на полиморфизъм, чрез който да се осъществи различната функционалност на клетките в зависимост от типа им.
2. Структури от данни
   * Стек - необходим при записване на направените в таблицата промени, а също така и при изпълнението на някои от алгоритмите.
3. Алгоритми
   * Алгоритъм за пресмятане на аритметични изрази - необходим при определянето на стойността на клетките, които са от тип “формула”.
   * Обхождане в дълбочина (DFS) - необходим при изчислението на стойността на клетки от тип “формула”, когато в аритметичния им израз има препратки към други клетки от същия тип.
   * Базови алгоритми за проверка на съдържанието на низ - необходими както при проверка за коректност на входните данни, така и за определяне на вида на дадена клетка в зависимост от подаденото съдържание.

**Обща архитектура**

Архитектурата се разделя на 3 части:

1. Помощна част
   * **class Stack<T>** - Шаблонен клас, който реализира структурата от данни “стек”.
2. Основна част
   * **class Cell** - Реализира клетка от “празен” тип и задава основните методи, които всеки друг тип клетка трябва да притежава.
   * **class IntegerCell** - Наследява **Cell** и реализира клетка от тип “цяло число”.
   * **class RealCell** - Наследява **Cell** и реализира клетка от тип “реално число”.
   * **class TextCell** - Наследява **Cell** и реализира клетка от тип “текст”.
   * **class FormulaCell** - Наследява **Cell** и реализира клетка от тип “формула”.
   * **class Row -** Реализира редовете на таблицата.
   * **class Table** - Реализира таблицата. Също така пази информация за направените промени в съдържанието на таблицата.
   * **class TableFile** - Съдържа адреса на отворения табличен файл и отговаря за прочитането и записването на съдържание на съответната таблицата под формата на файл.
3. Интерфейсна част
   * **class Commands** - Реализира достъпните за потребителя команди и осъществяващи проверки за коректността на входа им.



**Реализация**

1. Помощна част
   * **class Stack<T>** - Стекът е реализиран чрез свързано представяне. Има метод **bool push(T&&)**, който служи като оптимизация на стандартния “push”, тъй като използва “move” конструктора на обекта от тип T.
2. Основна част
   * **class Cell** - Разполага с методи, извличащи информация за клетката. Също така има статична член-функция char **checkType(const char\*)**, която по даден низ проверява дали му съответства някакъв тип клетка.
   * **class IntegerCell** - Добавя член-данна от тип **long long int**.
   * **class RealCell** - Добавя член-данна от тип **double**.
   * **class TextCell** - Добавя член-данна от тип **char**\*, реализираща низ.
   * **class FormulaCell** - Добавя член-данна от тип **char**\*, представляваща низ, който съдържа аритметичния израз на формулата.
   * **class Row -** Съдържа като член-данна динамичен масив от тип **Cell\***. Чрез динамично свързване елементите на масива се свързват към обекти, отговарящи на различните типове клетки.
   * **class Table** - Съдържа динамичен масив от тип **Row**, който реализира съдържанието на таблицата. Има вътрешна структура **PreviousContent**, която пази информация за направените промени и чрез стек от нея се пази цялата история на промените. Има и метод за принтиране на съдържанието на таблицата, при чието изпълнение се пресмятат стойностите на всички клетки от тип “формула”. За целта се използва алгоритъма за пресмятане на аритметични изрази, а при извикване на други клетки от тип “формула” се използва и версия на DFS за определяне на това кои клетки са изчислими. Използва се помощната структура **FormulaEvaluation** и косвена рекурсия.
   * **class TableFile** - Съдържа указател с тип **Table\***, който сочи към съдържанието на текущата таблица. Пази и адреса на файла, от който тя е била прочетена в низ от тип **char\***. Има методи, отговарящи за прочитането на съдържанието на файла и за запаметяване на съдържанието на текущата таблица в същия или друг файл.

3 Интерфейсна част

* + **class Commands** - Има методи, отговарящи за проверка на коректността на входа. Методът **void menu()** отговаря за разпознаване на командите, а останалите методи реализират функционалността на командите и използват само публичните методи на класовете от общата част.

**Заключение**

Крайните резултати от изпълнението на проекта са следните:

* Всички от поставените задачи бяха успешно реализирани. Това включва следните функционалности на програмата:
  + Успешно протичане на файлове в искания формат на данните и съответно запазване на таблици във файлове със същия формат.
  + Възможност отделните клетки да имат всеки от исканите типове в зависимост от тяхното съдържание. Програмата разпознава на кой тип клетка съотвества даден низ и кога низът е от невалиден тип, като открива по-честите грешки, които може да са били допуснати в този случай.
  + Извеждане на таблицата на екрана като изходът се форматира така, че всяка колона да е подходящо подравнена и да не заема много повече място от необходимото за извеждане на съдържанието й. Също така числата се подравняват вдясно, докато всякакви текстове се подравянват вляво.
  + Възможност за редактиране на клетки в таблицата. Програмата не извършва промени, ако потребителят е въвел невалидно съдържание за клетка, и съответно извежда подходящо съобщение. Също така има функционалност направените промени в таблицата могат да бъдат отменени.
  + Определяне на това дали дадена клетка от тип “формула” е изчислима и пресмятане на съдържанието й в случай, че е. Успешно взимане на стойността на други клетки, когато има препратки към тях в съответния аритметичен израз на формулата. Успешно реализиран приоритет на аритметичните операции като освен това са взети впредвид лявата асоциативност на ‘+’, ‘-’, ‘\*’ и ‘/’ и дясната асоциативност на ‘^’.
* Също така бяха реализирани и следните допълнителни функционалности:
  + Във аритметичните изрази е позволено да има скоби, които да указват допълнително реда на операциите. Това ползволява за по-голямо разнообразие на формулите, които могат да бъдат изчислени от програмата.
  + Във аритметичните изрази е позволено първият неинтервален символ след ‘=’ или ‘(’ да е някой от 5-та оператора. Тогава програмата приема, че пред този оператор стой числото 0. Това позволява операторите ‘+’ и ‘-’ да се използват като унарни, достатъчно е те и числото (или изразът) след тях да бъдат заградени в скоби. Тази функционалност дава възможност за директното използване на отрицателни числа в аритметичните изрази, без ‘-’ да създава двусмислие.

Окончателно, би могло да се заключи, че проектът е успешно реализиран.