***Софийски университет “Св.Климент Охридски”  
Факултет по математика и информатика***

A picture containing sketch, font, symbol, graphics

Description automatically generated

Обектно-ориентирано програмиране   
летен семестър 2022/2023

ДОКУМЕНТАЦИЯ НА ПРОЕКТ

***ПРИЛОЖЕНИЕ ЗА РАБОТА***

***С ЕЛЕКТРОННИ ТАБЛИЦИ***

Изготвил: Фросина Мулачка

ФН: 1MI8000006

Група: 1 Курс: 2

Специалност: Информационни Системи

**Юни**

**2023**

**Глава 1. Увод**

***1.1.Описание и идея на проекта***

Идеята на проекта е да се създаде конзолно приложение, което работи с електронни таблици. Приложението трябва да може да зарежда, записва и затваря една таблица. Таблицата се представя в текстов файл, като всеки ред във файла представя отделен ред в таблицата, а данните в реда се разделят със запетаи.

Програмата трябва да поддържа функционалности като извеждане на таблицата на екрана, редактиране на стойностите в клетките и обработка на грешки при неправилни данни.

***1.2.Цел и задача на разбработката***

Целта на разработката на приложението за работа с електронни таблици е да създаде функционално конзолно приложение, което да позволява работа с данни в табличен формат. Приложението трябва да може да отваря, записва и затваря електронни таблици, както и да извършва операции като извеждане на съдържанието на таблицата на екрана и редактиране на стойностите на клетките.

Задачите на разработката включват:

1. **Зареждане на таблица:** Приложението трябва да може да зарежда електронни таблици от текстови файлове и да ги преобразува във вътрешна структура, която да позволява манипулация с данните.
2. **Запис на таблица:** Потребителят трябва да може да запазва промените в таблицата обратно в текстовия файл (или в нов файл), като приложението трябва да гарантира правилното форматиране на данните във файла.
3. **Затваряне на таблица:** Приложението трябва да освобождава ресурсите, свързани с отворената таблица, и да я затваря.
4. **Извеждане на таблицата на екрана:** Приложението трябва да има възможност да извежда съдържанието на таблицата на стандартния изход, като клетките трябва да бъдат подравнени и разделени с отвесни черти.
5. **Редактиране на клетки:** Потребителят трябва да може да редактира стойностите на отделните клетки в таблицата, като приложението трябва да валидира въведените данни и да ги запазва в правилния формат.
6. **Обработка на формули:** Приложението трябва да поддържа формули в клетките, които могат да съдържат аритметични операции, литерали (числа) и препратки към други клетки. Трябва да се извършва правилното оценяване на формулите и да се обработват грешките, като вместо грешни резултати се извежда "ERROR".
7. **Обработка на грешки:** Приложението трябва да бъде устойчиво към възможни грешки във входните данни или форматирането на таблицата. Трябва да се предоставят съобщения за грешки и да се осигури коректна обработка на изключенията.

Целта и задачите на разработката са да осигурят функционално и надеждно приложение, което да позволява лесна и ефективна работа с електронни таблици.

**Глава 2. Преглед на предметната област**

***2.1.Основни дефиниции, концепции и алгоритми***

Програмата се състои от две основни части – класът **Cell** и класът **Spreadsheet**.

1. **Cell**:
   * Служи за съхранение на данни в една клетка от електронната таблица.
   * Форматирайки го като **pure virtual** клас, постигаме по-добра абстракция и спазваме основните ООП принципи. Класът Cell е наследяван от класовете: **Integer, Fractional, String, Formula** които са валидния тип данни които може да съдържа една клетка, както по-горе е споменато.
   * Предоставя функционалности за получаване на стойностите на клетката, промяна на стойностите и извличане на информация за клетката.
2. **Spreadsheet:**

* Класът в който е реализирана главната функционалност на проекта.
* Представя цялата електронна таблица и управлява данните в нея.
* Предоставя функционалности за парсване на входния файл, извличане на стойностите на клетките, промяна на клетки, отпечатване на таблицата и запис на таблицата във файл.

***2.2. Дефиниране на проблеми и сложност на поставената задача/Подходи на решаването на проблемите***

1. **Парсване на файл:**
   * *Проблем:* Прочитането на входния файл и извличането на данните за електронната таблица може да бъде предизвикателно, особено ако файлът съдържа големи количества данни.
   * *Сложност:* Сложността на парсването на файловете зависи от размера на файла и сложността на формата на данните.
   * *Подходи за решаване на проблема:*Дефиниране на формат на данните (даден в условието) и извличане на информацията съгласно него (създаване на собствен парсър);  
       
     Обработка на възможни грешки и изключения при парсването на файловете;
2. **Оценяване на формули:**
   * *Проблем:* Формулите в електронната таблица могат да съдържат препратки към други клетки, което изисква правилно оценяване на тях.
   * *Сложност:* Оценяването на формулите изисква обхождане на клетките и изпълнение на математическите операции, което може да бъде времеемко и сложно.
   * *Подходи за решаване на проблема:*  
     Имплементиране на алгоритми за обхождане на клетките и пресмятане на формулите, като се отчитат зависимостите между клетките.
3. **Редактиране на клетки:**
   * *Проблем:* Промяната на стойностите в клетките и обновяването на свързаните клетки и формули може да предизвика проблеми като неконсистентност или грешки в данните.
   * *Сложност:* Сложността на редактирането на клетки зависи от броя на свързаните клетки и зависимостите между тях. При промяна на стойностите може да е необходимо да се обновят и преоценят всички зависими клетки.
   * *Подходи за решаване на проблема:*  
       
     Използване на алгоритми за обновяване на зависимите клетки и изпълнение на формулите след редакция на клетка;  
       
     Проверка за валидност на промените и обработка на грешки, които могат да възникнат при редактирането на клетки.

**Глава 3. Проектиране**

***3.1.Обща архитектура – ООП дизайн***

При проектирането на приложението за електронна таблица се използва обектно-ориентиран подход за постигане на модуларност, разширяемост и поддръжка на кода.

1. Клас **Cell** (абстрактен клас): Този клас представлява абстрактна клетка в електронната таблица и е базов клас за различните типове данни, които могат да бъдат съхранени в клетката. Има следните член-данни:
   * **convertedValue** - преобразувана числена стойност на клетката
   * **stringValue** - символен низ, представляващ стойността на клетката като текст
   * **type** - тип на клетката (CellType)
   * **row** - номер на реда, в който се намира клетката
   * **col** - номер на колоната, в която се намира клетката

Има следните методи:

* + Конструктори за създаване на обект от класа
  + Селектори (get-методи) за достъп до член-данните на класа
  + Методи за промяна на стойността на клетката
  + Виртуални методи **print()** и **saveInFile()**, които се наследяват от производните класове и представят начин за отпечатване на клетката и запис във файл
  + Виртуален метод **clone()**, който създава копие на клетката

1. Класовете-наследници на Cell, т.е. **Integer,Fractional,String,Formula** имплементират (чисто) виртуалните методи на базовият клас.
2. Клас **Row**: Този клас представлява ред от електронната таблица и съдържа набор от клетки. Има следните член-данни:
   * **cells** - вектор от указатели към клетките в реда
   * **rowNumber** - номер на реда

Има следните методи:

* + Конструктори за създаване на обект от класа
  + Методи за добавяне на клетка, достъп до клетките и др.

1. Клас **Spreadsheet**: Този клас представлява електронна таблица и основната функционалност на приложението. Има следните член-данни:
   * **rows** - вектор от указатели към редовете на таблицата
   * **numberOfRows** - брой редове в таблицата
   * **numberOfColumns** - брой колони в таблицата
   * **fileName** - име на текущо отворения файл Има следните методи:
   * Конструктори за създаване на обект от класа
   * Методи за отваряне на файл, парсиране на файла и създаване на таблицата
   * Методи за отпечатване на таблицата, редактиране на клетка, запис на таблицата във файл и други помощни методи.
2. Клас **Helpers** е статичен клас който предоставя набор от помощни методи, които се използват в различни части на програмата и помагат за обработката на данни и изпълнението на различни операции.
3. Клас **Interaction**: Този клас предоставя възможност за интеракция с потребителя чрез команди. Включва следните команди:
   * **open** - отваряне на файл
   * **close** - затваряне на текущо отворения файл
   * **print** - отпечатване на таблицата
   * **edit** - редактиране на клетка
   * **save** - запис на таблицата в текущо отворения файл
   * **saveas** - запис на таблицата в нов файл
   * **help** - извеждане на списък с команди и помощна информация
   * **exit** - изход от програмата Има следните методи:
   * Конструктори за създаване на обект от класа
   * Методи за обработка на въведените команди и изпълнение на съответните действия

**Глава 4. Реализация, тестване**

***4.1. Реализация на класовете***

С така разделените класове, се улеснява работата при разбработката на проекта и кодът е доста четим.

A picture containing text, screenshot, software, multimedia software

Description automatically generated

A picture containing text, screenshot, software, multimedia software

Description automatically generated

A picture containing text, screenshot, software, multimedia software

Description automatically generated

A screenshot of a computer program

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated with medium confidence

A screen shot of a computer

Description automatically generated

***4.2. Управление на паметта и алгоритми. Оптимизации.***

***Управление на паметта:***

* За класовете, които създават динамични обекти (например класовете Cell, Row и други), е важно да се осигури правилно управление на паметта, за да се избегнат утечки на памет и дефектите на програмата.
* При създаването на динамични обекти трябва да се използва операторът new, а при освобождаването на заделената памет - операторът delete.
* В класовете, които използват динамична памет, трябва да се реализират голямата четворка за правилно копиране и освобождаване на паметта.

***Оптимизации на алгоритмите:***

* Избягване на излишни операции и изчисления, така че при извикване на функции параметрите ги подаваме като константни референции, когато е възможно, за да избегнеме копирането на данни.
* Избягването на излишни проверки и изчисления, оптимално управление на паметта при добавяне, премахване или редактиране на клетки и ефективно изчисление на формули и функции.

***4.3. Планиране, описание и създаване на тестови сценария***

Имаме следната електронна таблица, записана във текстов файл с име: spreadsheet

A screenshot of a computer

Description automatically generated with medium confidence

A computer screen shot of a black screen

Description automatically generated with low confidence

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Глава 5. Заключение**

***5.1. Обобщение на изпълнението на началните цели.***

Мисля, че проектът успя да постигне своите начални цели, които включваха създаване на приложение за обработка на таблични данни. Беше разработена архитектура, базирана на принципите на обектно-ориентираното програмиране, която включва класове за клетки, редове и таблица. Основните функционалности, като отваряне, редактиране и запазване на файлове, бяха успешно реализирани.

***5.2.Насоки за бъдещо развитие и усъвършенстване***

В бъдеще, може да се разработят допълнителни функционалности и да се направят усъвършенствания в приложението. Някои от насоките за бъдещо

* Подобряване на интерфейса на потребителското взаимодействие
* Подобряване на ефективността
* Добавяне на допълнителни функции за обработка на данни

С бъдещото развитие и усъвършенстване на приложението, то може да стане по-мощен и удобен инструмент за обработка и анализ на таблични данни.

**Използвана литература:**

1. <https://en.cppreference.com/w/>
2. <https://www.geeksforgeeks.org/>
3. Maтериали от лекции и упражнения