



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

# **ПРОЕКТ ЗА КУРСА ПО ОБЕКТНО ОРИЕНТИРАНО ПРОГРАМИРАНЕ**

Тема: Приложение за работа с електронни таблици

Изготвен от:  
Андрей Калчев

СОФИЯ

2022

# **ПЪРВА ГЛАВА**

## **Увод**

### **1.1. Описание и идея на проекта**

Идеята на проекта "Приложение за работа с електронни таблици" е да се напише програма, която поддържа функции за обработка на данни, записани в текстови файлове. Тя трябва да може да ги прочита, валидир, принтира, редактира и запазва.

### **1.2. Цел и задачи на разработката**

Целта на разработката е да се напише програма, която да прочита данни от текстов файл и да ги представя във вид на електронна таблица, която трябва да има основните функционалности на електронните таблици.

Първата задача е да се създаде базов клас "Клетка", заедно с класове наследници за всеки отделен тип данни, в който да се записват данните от файла, заедно с допълнителни параметри, необходими за по нататъшно редактиране на клетките.

Втората задача е да се напише функция, която да определя типа на клетката спрямо прочетените данни.

Третата задача е създаване на клас Таблица. Класът трябва да има методи за прочитане, валидиране и запазване на множество данни във вид на клетки, за записване на таблицата в текстов файл, за изваждане на данните на екрана, за редактиране на клетките, за сортиране и за създаване на извадка.

### **1.3. Структура на документацията (3–4 изр.)**

Документация на проекта се състои от: Заглавна страница, Уводна страница, Глава с описание на плана за изпълнение на задачата и основни сложности, Глава описваща архитектурата на проекта, Глава за основната реализация, Заключение и Използвана литература.

## **ВТОРА ГЛАВА**

### **Преглед на предметната област**

#### **2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани**

В проекта са използвани основните концепции на обектно ориентираното програмиране: енкапсулация, абстракция, преизползване и полиморфизъм.

Включени са дефинициите на вектор и стринг и за сортиране се използва методът на мехурчето.

#### **2.2. Дефиниране на проблеми и сложност на поставената задача**

Първият проблем е преобразуването на данните от текстовия файл в конкретния тип клетка, което се усложнява от проверката за изпусната запетая.

Вторият проблем са празните клетки, които не са отбелязани категорични като такива със запетай.

Третият проблем е в дефинирането на релацията на еквивалентност. Примерно клетката "400" трябва да е по-голяма от 100 и по-малко от "C\test", но 100 е по-голямо от "C\test". Така написано условието в множеството от клетки няма релация на еквивалентност.

#### **2.3. Подходи, методи за решаване на поставените проблеми**

Първият проблем се решава от проверка за минимална дължина на данните. Клетките от тип валута и стринг имат дължина, под която те не могат да бъдат валидни, а проверката за цяло и дробно число остава тривиална.

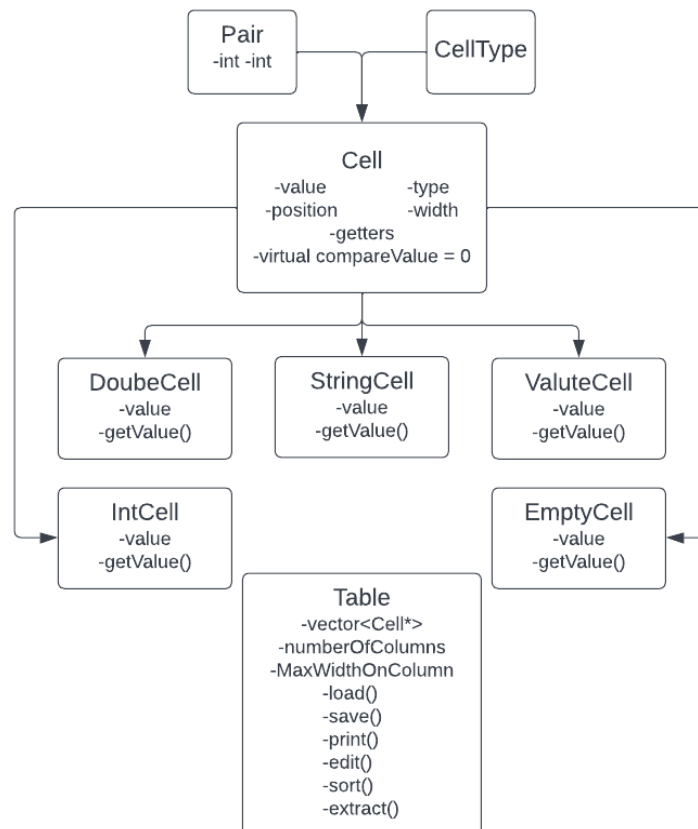
Вторият проблем се решава с допълнителна характеристика, която се пази в класа Таблица - брой колони.

Третият проблем се решава от избраният метод за сортиране. Използва се това сравнение, което се прави първо.

## ТРЕТА ГЛАВА

### Проектиране

#### 3.1. Обща архитектура – ООП дизайн



Фиг. 3.1 Диаграма на класовете

На фигура 3.1 е показана общата архитектура на проекта. Класът Клетка е абстрактен базов клас. Класовете, които го наследяват са клетки от тип цели числа, дробни числа, стринг, валута и празна клетка, всяка от която съдържа стойност от съответния тип данни. Двойка е структура написана, за да се съхраняват колоните и редовете на клетките на едно място и се използва в базовия клас Клетка. Тип на клетката е изброим тип, който съдържа всички възможни типове данни на клетките и стойност за невалидна клетка и се използва в класа Клетка. Таблица е клас, който съдържа вектор от тип указател към клетка, за съхранение на множество клетки и метори да тяхното обработване.

## 3.2. Важни функции извън класовете

```
CellType typeOfString(const std::string& value);  
int slashesCount(const std::string& value, int i);  
  
Cell* createCell(unsigned row, unsigned column, std::string value);  
std::string removeWhiteSpaces(const std::string& value);
```

Фиг. 3.2 Функции за създаване на обект от клас Клетка

На Фиг. 3.2 са показани дефинициите на функциите, които създават обект спрямо прочетената стойност от текстов файл. Първата определя на типа на клетката спрямо стойността на стринг. Третата създава на обект от клас наследник и връща указател към него. Останалите две са помощни функции.

```
bool compareTwoCellsBigger(const Cell* one, const Cell* two);  
bool compareTwoCellsSmaller(const Cell* one, const Cell* two);  
bool compareTwoCellsBiggerOrEqual(const Cell* one, const Cell* two);  
bool compareTwoCellsSmallerOrEqual(const Cell* one, const Cell* two);  
bool compareTwoCellsEqual(const Cell* one, const Cell* two);  
bool compareTwoCellsDifferent(const Cell* one, const Cell* two);  
int getEquasion(std::string& condition);
```

Фиг. 3.3 Помощни функции за клас Таблица

На Фиг. 3.3 са показани дефинициите на шестте оператора за сравнение на обекти от класове наследници на Клетка. Те се използват в методите за сортиране и създаване на извадка на класа Таблица. Показана е и дефиницията на функцията за определяне на знака за сравнение за третия тип създаване на извадка.

## ЧЕТВЪРТА ГЛАВА

### Реализация, тестване

#### 4.1. Реализация на класове

Абстрактен базов клас Клетка - Съдържа Съдържа абстрактен метод, който връща дробно число, което се използва за сравняване на клетки от различен тип.

Наследниците на класа Клетка са: Целочислена Клетка, Клетка за дробни числа, Клетка за стринг, Клетка за валута, Празна Клетка. Всеки от

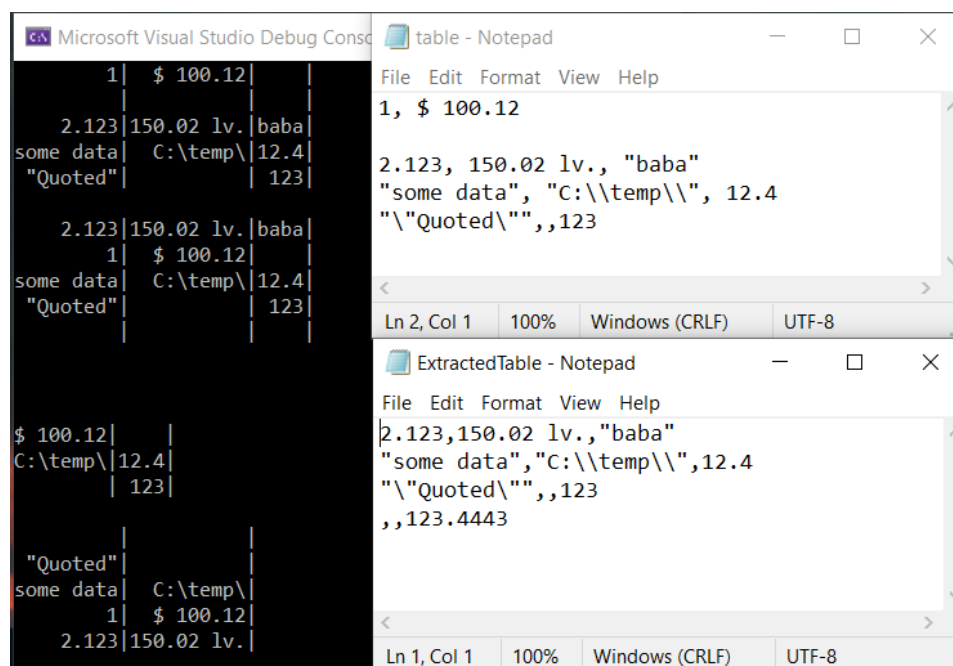
наследниците има параметър за стойност, който е от съответния тип за клетката.

Класът Таблица е основният клас за програмата. Той съдържа масив от клетки и методи за зареждане на файл, запазване във файл, извеждане на стойностите, промяна на стойност, сортиране и създаване на извадка.

## 4.2. Управление на паметта и алгоритми. Оптимизации

Основното управление на паметта се извършва от класовете вектор и стринг. Едната явно заделена динамична памет е във функцията за създаване на клетка по подадени стойност и позиция. Функцията се извиква в частен метод на класа Таблица, който се използва само в конструктора на класа. За освобождаване на тази памет при грешка или при изтриване на обект от класа се грижи друг частен метод. Втората заделена динамична памет е при методите за създаване на извадка. Тя се освобождава, която извадката вече не е нужна. Последната е в третия метод за извадка. Създава се нова клетка за сравнение, която се освобождава когато вече не е нужна или при възникване на изключение.

## 4.3. Демонстрация



```
1 | $ 100.12 | |
2.123 | 150.02 lv. | baba |
some data | C:\temp\ | 12.4 |
"Quoted" | | 123 |

2.123 | 150.02 lv. | baba |
1 | $ 100.12 | |
some data | C:\temp\ | 12.4 |
"Quoted" | | 123 |

$ 100.12 | |
C:\temp\ | 12.4 |
| 123 |

"Quoted" | |
some data | C:\temp\ |
1 | $ 100.12 |
2.123 | 150.02 lv. |
```

```
File Edit Format View Help
1, $ 100.12

2.123, 150.02 lv., "baba"
"some data", "C:\\temp\\", 12.4
"\"Quoted\\\"",123

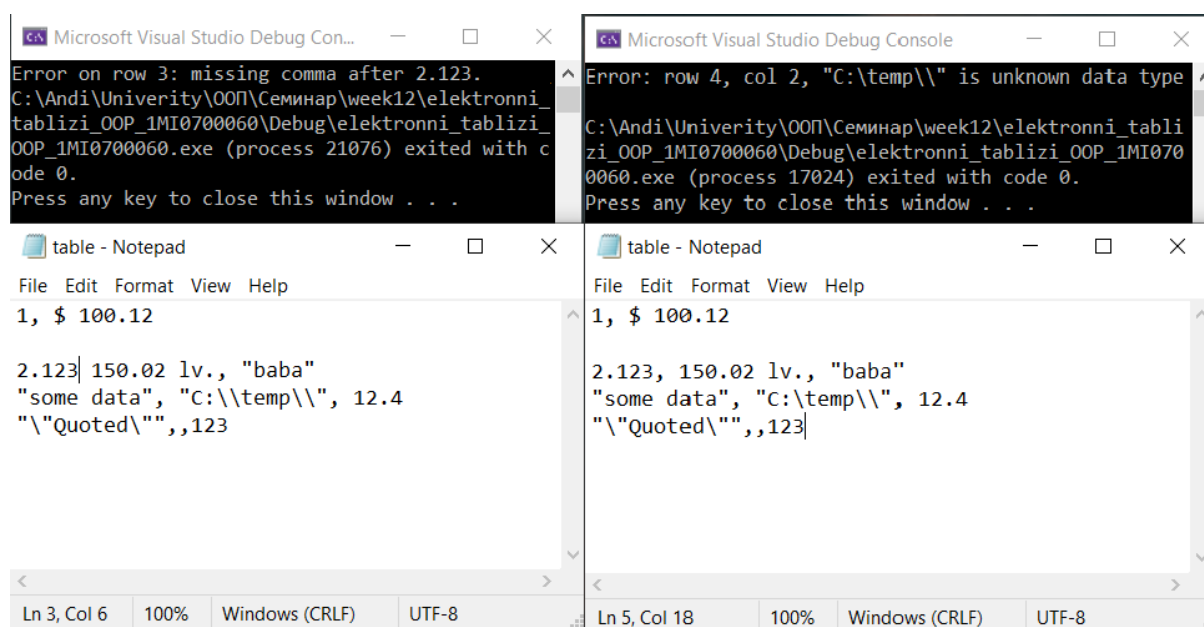
Ln 2, Col 1 100% Windows (CRLF) UTF-8

ExtractedTable - Notepad
File Edit Format View Help
2.123,150.02 lv., "baba"
"some data", "C:\\temp\\", 12.4
"\"Quoted\\\"",123
,,123.4443

Ln 1, Col 1 100% Windows (CRLF) UTF-8
```

Фиг. 4.1 Тестове на програмата

На Фиг. 4.1 са демонстрирани всички функционалности на програмата. Таблицата в горния текстов редактор е оригиналната таблица, която се зарежда в програмата. Първата таблица в конзолата е принтираната прочетена таблица. Втората - сортирана по първи стълб в низходящ ред таблица. Третата - извадка от втори ред втори стълб до четвърти ред трети стълб. Четвъртата - извадка на първа и втора колона, сортирана във възходящ ред по първи стълб. Таблицата във втория текстов редактор е записаната извадка на втората таблица по критерия  $\#3 \geq 0$ , след като клетката на пети ред трета колона е променена на 123.4443.



Фиг. 4.2 Съобщения за грешка при зареждане на файл

На Фиг. 4.2 са илюстрирани двете грешки при зареждане на невалиден текстов файл. На теста отляво е демонстрирано намирането на пропуснати запетаи в оригиналния файл. На теста отдясно е показано различаването на валидни типове данни. И в двата случая програмата се прекратява по нататъшната си работа.

## ПЕТА ГЛАВА

### Заклучение

#### 5.1. Обобщение на изпълнението на началните цели

Всички поставени задачи са изпълнени.

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

Идеи за бъдещо усъвършенстване на проекта са цялата работа с проверяването и конвертирането на текстови данни да се извършва от специално направен за това клас, трите метода за извадка на таблица да се обединят в един и създаване на потребителско меню, което се навигира със стрелки.

### ИЗПОЛЗВАНА ЛИТЕРАТУРА

- <https://cplusplus.com/reference/vector/vector/> - документация за vector
- <https://cplusplus.com/reference/string/string/?kw=string> - документация за стринг
- <https://www.geeksforgeeks.org/passing-a-function-as-a-parameter-in-cpp/> - подаване на функция като параметър на друга функция
- <https://cplusplus.com/reference/string/string/?kw=string> - exit() функция
- <https://mathbits.com/MathBits/CompSci/Introduction/clear.htm> - изчистване на конзолата
- <https://learn.fmi.uni-sofia.bg/mod/resource/view.php?id=241316> - работа с текстови файлове