Тема №5 – Приложение за работа с електронни таблици (spreadsheets)

Калоян Георгиев, 8mi0400123

1. **Увод**
   1.1. Описание и идея на проекта:

Проектът представялва конзолно приложение, което работи с електронни таблици. То може да приема текстови файлове, чиито данни са отделени чрез запетаи и нови редове. Използвайки прилжението, потребителя получава достъп до информацията в таблиците.

   1.2. Цел и задачи на разработката

Приложението може да отваря такива файлове, като за целта потребителят посочва директорията, в която се намира съответния файл. След посочването на файла, приложението разпределя информацията от него в клетки във виртуална таблица. След зареждането на данните в таблицата, те могат да бъдат от различни типове. Тази таблица би могала да бъде отворена от приложението, за да може потребителя какво тя съдържа като информация. Потребителя може да редактира информацията във всяка една клетка от таблицата по отделно. Освен чилса и символни низове, в клектите може да се съдържат и изрази (като например събиране, изваждане,..., сравнение) В един такъв израз, освен числа, могат да се съдържат и препратки към други клетки в таблицата. След като потребителя е прилючил в редактирането на таблицата, той може да я запази. Запазването може а става в същия файл, от който е взета първоначало текущата таблица, или в нов файл, като потребителя трябва да посочи име на файла и директорията му, кудето да бъде запазен. Ако желае да отвори нов файл с нова таблица, той може да затвори текущия и да отвори нов. След като приключи с работата си потребителя може да затвори приложението.

   1.3. Структура на документацията

В документацията на проекта, ще са посочени концепции и алгоритми, които са използвани. Ще бъдат описана архитектурата на проекта. Ще бъдат обяснени, използваните впрограмата, методи и фукнции. Ще бъдат посочени тестови данни и съответните им резултати при изпълнението на програмата.

2. **Преглед на предметната област**
   2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

При зареждането на определен файл, той трябва да бъде прочетен и данните в него да бъдат рапределени във виртуална таблица. За целта при четенето на определен файл, последователно започва да се чете всики символ (един по един), като при достигането на разделителния символ – запетая, символния низ, който се е събрал от до сега прочетените символи се запава в таблицата. За целта, първо се проварява какъв тип данна съдържа той (число, цимволен низ, израз). След това той бива запазен за определеното му място в траблицата. Тъй като четенето на данните става последователно, данните във виртуалната таблица се записват един по един като се започва от първия индекс на първия ред до последния индекс на последния ред (като определянето на реда и индекса става, чрез броене на до сега изминалите запетаи и нови редова).

При принтирането на заредената витуална таблица отделните клетки трябва да са подравнени по колони. За целта при самото принтитране на данните, се минава веднъж през всяка отделна клатка в

таблицата по колони, като за всяка колона се запазва дължината на най-дългата стойност в нея. Така преди притирането на стойността на всяка отделна клетка, се принтират толкова празни места, че да последния символ от съответната стоойност да се намира на същата позиция като последния символ на най-дългата стойност.

Но за да може една клетка да бъде принтирана, то нейнта стойност тярбва да бъде изчислена. При числата и символните низове е очевидн, но стойността на един израз трябва да бъде изчислена не при запазването и, а в хода на всяко едно принтиране. Причината за това е, че, ако стойността в клетката, която е посочена като препратка в някой израз, е променена, то и стойността на текущата клетка ще се промени.

Променянето на стойност в дадена клетка наподобява на първоначалното запазване на стойности (при зареждането на файл). Първоначално се провярява дали посочета клетка от потребителя е валидна. Ако е, посочената стойност (която приема формата на символен низ) се проверява дали е число, стринг или израз (със същите методи, които биват използвани при записването файловете), след което се запазва на посочената клетка в опередделената форма – число, низ или израз.

Запазването на витруалната таблица във файл се случва елемент по елемет, като след след всеки елемент се записва и едни разделителен символ – запетая, освен когато е последен от реда си, тпгава вместо запетая се слага символ за применаване на нов ред. Изключение прави последния елемент на последния ред – след него не се записва нищо.

Запазването в нов файл в същото като запазване в текущия, с изключение на това, че потребителя тряба да въведе името на новия файл и директорията, където желае да се намира.

2.2. Дефиниране на проблеми и тяхното разрешаване

Когато потребителят иска да се изпълни действие от приложението той трябва да въведете съответната команда. При въвеждането на невалидна команда никакво действие не се изпълнява с изключение на извеждането на съобщение за грешка. След което се дава нова възможност за въвеждане на команда от потребителя.

Когато потребителят желае да отвори файл, чието име или директория е въведено грешно, на екрана се извежда съобщение, че посоченият файл не би могъл да се отвори. След което, ако потребителя желаеm може, да се опита да отвори нов файл.

При принтирането на файла, стойностите на клектит, които съдържат изрази, трябва да се изчислят. Биха могли да се получат два проблема: Първият е добре познатото ни на всички – делене на 0. Когато израза съдържа делене на 0 или делене на препратка към клетка, чиято стойност е 0, на мястото на съответната клетка като стойност се извежда текста „ERROR“; Вторият проблем е като се получи безкрайна рекурсия чрез препратки към други клетки (Например – имаме клетка А, която съдържа препратка към клетка Б, но в клетката Б се съдържа препратка към клетката А), така рекурсията няма дъно и не би могло да се изчисли конкретна стойнст. Тогава на мястото на съответната клетка като стойност се извежда текста „NULL“.

Когато потребителят иска да промени стойност на клетка във вече заредена таблица, той може да въведе адрес на клетка, която е извън границите на таблицата. Тогава да екрана се изчежда

съобщение за невалиден адрес на клетка и, ако потребителят желае, може да въведе нов адрес на клетка и да промени стойността и.

При запазването би могло файлът, в който ще се записва таблицата, да не се отвори. Тогава операцията се прекратява и се извежда съобщение на екрана уведомяващо потребителя.

3. Проектиране

3.1. Обща архитектура Програмата

съдържа 6 класа:

1. Table – който отговаря за таблицата. Той съдържа брой редове, брой колони и хетерогенна матрица от *Element*-и.

2. my_string – помощен клас, който опростява работата в другите касове, замествайки char*-масивите

3. Element – абстрактен клас, представящ клетките в матрицата на *Table*. Той има три класове-наследници: *MyNumber, MyString* и *Formula*

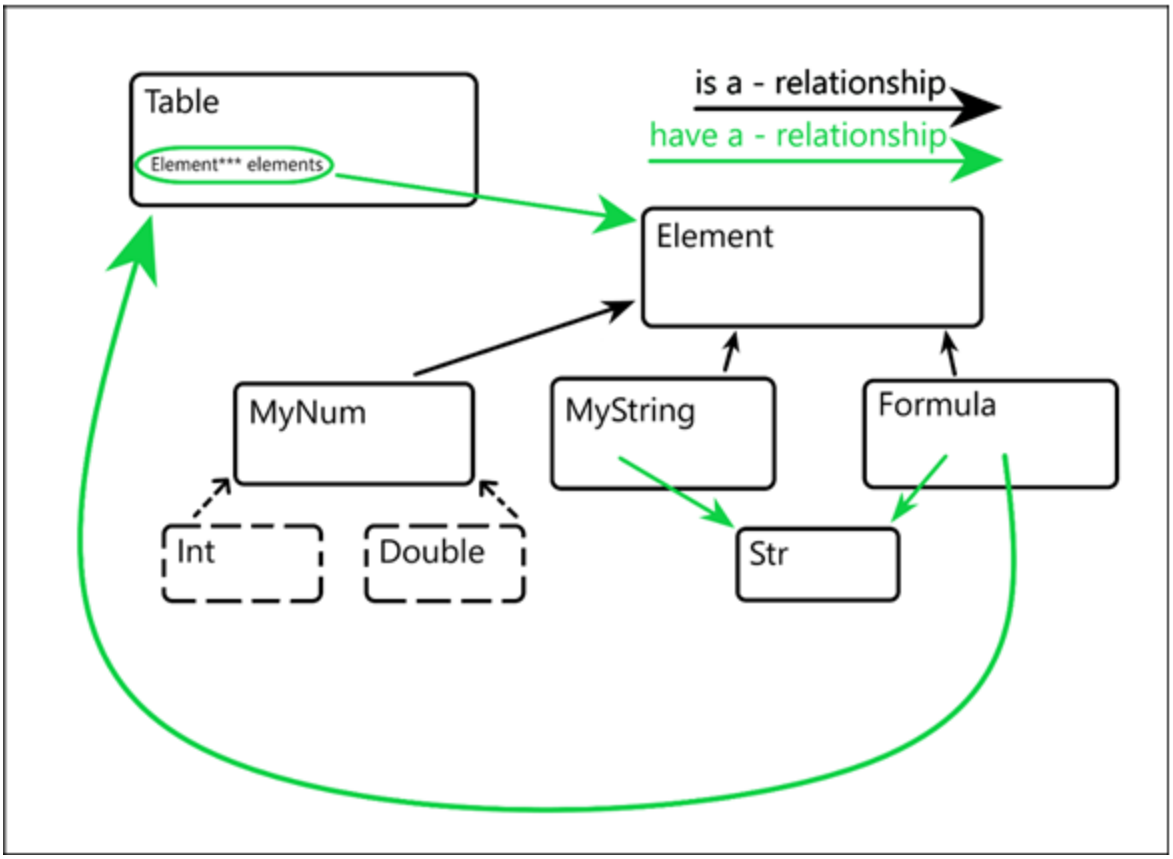
4. Num – темплейтен клас (наследник на *Element*). В хода на програмата той създава два класа – един от тип *int* и един от тип *double*. Този клас предтавя клетките в таблицата, които да числа.

5. Str – клас, наследяващ *Element*. Този клас представя клетките в таблицата, които са символни низове.

6. Formula – клас, наследяващ *Element*. Този клас предатвя изрази (формули). Всяка формула започва със „=“ след което има два литерала (които биха могли да бъдат или числа, или препратки към дтруги клетки) и между тях символ за аритметична операция или символ за сравнение.

7. Interface – служи за работа с потребителя. Съдържа класове-команди, организирани в полиморфна йерархия.

3.2. Диаграма



4. Реализация, тестване

4.1. Реализация на класове

Вкласа Table има метод (*addElement*), който добавя елементи на подадена позиция, като за да разбере какъв елемет да използва методите *isInt, isDouble, isFormula*.

В класа има има метод *edit*, който извиква метода *addElement. И*ма и клас *print,* който отпечатва виртуалната таблица на конзолата, както описано по-горе.

В класа *Element* има два *pure virtual* метода, предефиниран операторът << и метод *print,* който извиква оператор <<. Има такъв отделен метод, защото преди една клетка да се отпечата нейната стойност трябва да бъде изчислена (както бе описано по-горе, относно изразите). И, разбира се, един виртуален деструктор.

В *Num* и в *Str* виртуалните методи биват предефинирани.

```
my_string Formula::getValueAsStr() const
{
    try
    {
        return doubleToStr(getNum());
    }
    catch (...)
    {
        return "NULL";
    }
}
```

В класа *Formula* биват предефенирани виртуалните методи, като за принтирането на конзолата на съответаната клетка (която е от тип *Formula*) се извиква *getValueAsStr,* която извиква *GetNum* и връща резултата от *double* в тип *Str,* използвайки метода doubleToStr.

Относно *GetNum* – взема всеки един от двата литерала и проверява дали са препратки към други клетки. Ако са – извиква *GetNum* на посочената клетка, ако ли не – извзлича стоийността на числото.

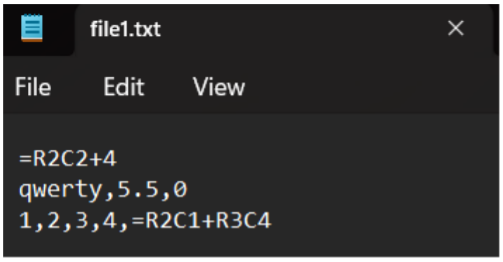
```
if (strcmp(symbol.str(), "/") == 0)
{
    this->met = false;
    if (rLiteral != 0)
    {
        return (double)lLiteral / (double)rLiteral;
    }
    else
    {
        throw std::invalid_argument("Division by zero");
    }
}
```

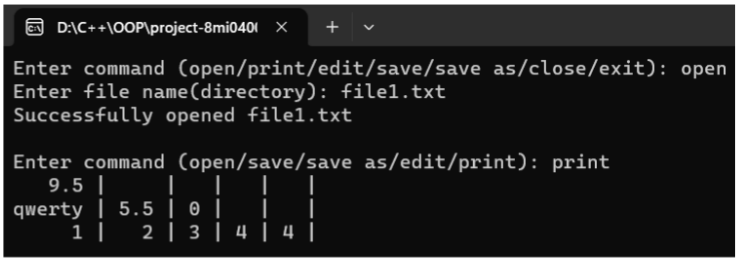
След което проверява какъв е знака в израза и прави нужната операция. Като специфичното е при деленето.

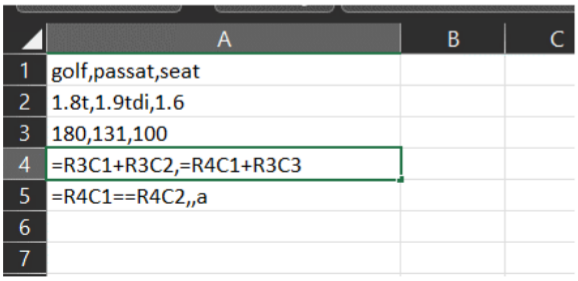
4.2. Управление на паметта

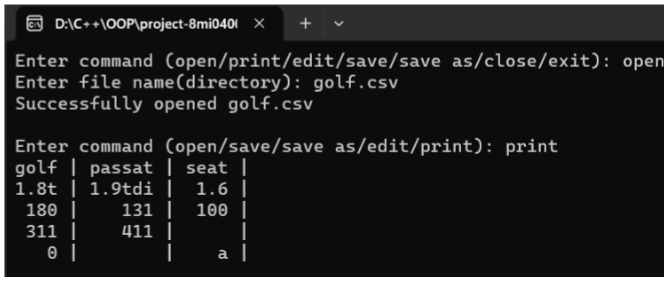
В класовете *Str* и *Formula* единственото заделяне на памет е чрез *my_string,* който сам се грижи за паметта си. Но в класа *Table* имаме динамична – хетерогонна матрица, за която трябва да се погрижим. Затова имаме деструктор):

4.3. Тестови сценарии









5. **Заключение**

5.1. Обобщение на изпълнението на началните цели

Реализацията на програмата започна от създаването на класовете *Element, Num, Str u Formula,* като биват тествани многократно. След което бива създаден класа *Table* и вече, с достигането до метода *print,* дойде време да се разлизира и метода *GetNum* на класа *Formula.*

След което дойде време и за *main*-а и функциите, които се използват в него. И нарая тестването с тестовите данни, посочени по-горе.

5.2. Насоки за бъдещо развитие и усъвършенстване

В бъдеще би могло да се бодавят нови типове данни. Това не би било проблем, защото ще трябва да се добави нов клас за тях и съответните му проверки и методи, които да извличат информацията за него от символен низ.

Използвана литература

https://en.cppreference.com/w/

https://stackoverflow.com

https://github.com

Лични записки