Общо за проектите в диалогов режим

В тези проекти потребителят работи с програмата, като въвежда различни команди. Те могат да имат нула, един или повече параметри. Параметрите се разделят помежду си с интервали.

Една команда, която всички такива проекти трябва да поддържат е `exit`, за излизане от програмата:

```
> exit
Exiting the program...
```

Имената на командите трябва да са case-insensitive. Например `open`, `OPEN`, `Open` и `OpEn` са валидни изписвания на командата за отваряне на файл. Потребителят може да използва която предпочете форма.

Аргументи обаче може да са case-sensitive. Добър пример е командата `open`, която получава път до файл, който да се отвори. Този път може да бъде case-sensitive.

Понякога в самите аргументите може да има интервал. В такъв случай те се ограждат с кавички. Например ако искате да отворите файл с име `New File.txt`, това може да стане по следния начин:

```
> open "C:\Temp\New File.txt"
Successfully opened New File.txt
```

Освен ако не е казано друго, всяка от командите трябва да извежда съобщение, от което да е ясно дали е успяла и какво е било направено.

Понякога, дадена команда може да разчита на това преди нея да е била изпълнена друга. Например, преди да можете да обработите дадено изображение, трябва да сте го заредили в паметта. За такива команди трябва да правите проверка дали те могат да бъдат изпълнени. Ако това не е възможно, изведете съобщение за грешка и подскажете на потребителя какво да направи. Например:

```
> saveas "C:\Temp\New File.txt"
Error: no document is currently opened
Hint: open an existing file, or create a new document first.
```

Когато планирате архитектурата на решението си, планирайте как да отделите слоя обработващ командите от останалата логика на програмата.

Общо за проектите работещи с документи

Тези проекти зареждат информация от някакъв вид файлов формат, обработват я и след това дават възможност за обновяване на информацията на диска. Те трябва да включват функционалността описана в настоящия раздел.

За тези проекти, повечето команди могат да се изпълняват само ако в момента има отворен документ. Това става ако успешно бъдат изпълнени командите `open` или `new`. Респективно, ако изпълните `close`, това затваря текущия документ и повечето команди отново няма да работят.

Ако счетете за нужно, можете да добавите допълнителни аргументи към командите описани в този раздел. Например може да добавите аргумент на `saveas`, който да презаписва даден файл без да иска потвърждение.

Отваряне на файл (Open)

С командата `open` може да се зареди съществуващ файл. Тя получава като аргумент пътя на файла, който трябва да се отвори. Например:

```
> open C:\Temp\file.xml
Successfully opened file.xml
```

Ако файлът с указания път не съществува (или нямате достъп до него и т.н.), трябва да се изведе подходящо съобщение за грешка.

Когато отворите даден файл, неговото съдържание трябва да се зареди в паметта, след което файлът се затваря. Всички промени, които потребителят направи след това, се отразяват само в паметта. Ако потребителят иска да ги запише обратно на диска, той трябва да го направи изрично, чрез някоя от командите за запис.

Във всеки от проектите има посочен конкретен файлов формат, с който приложението ви трябва да работи. Това означава, че:

1. то трябва да може да чете произволен валиден файл от въпросния формат;
2. когато записва данните, то трябва да създава валидни файлове във въпросния формат.

Ако при прочитането на информацията от файла установите някакъв проблем, програмата трябва да изведе подходящо съобщение за грешка. Например, възможно е файл съдържащ изображение да е бил повреден и да е била загубена информацията за част от пикселите в него.

Ако при извикването на `open` в програмата вече има зареден друг документ, тя първо трябва да го затвори и чак след това да отвори файла.

Документите имат свойство “текущо файл”, което сочи към файла, с който те са свързани. Ако отворите документ с командата `open`, той автоматично се свързва с файла, от който е бил прочетен.

Затваряне на документ (Close)

Това може да стане ръчно с командата `close`. Ако изпълните командите `open`/`new`, когато в програмата има зареден документ, те автоматично го затварят и чак след това се отваря посоченият файл/създава празен документ.

Когато затваряте даден документ, проверете дали промените в него са били записани на диска. Ако това не е така, попитайте потребителя дали иска да ги запише, да ги отхвърли или да се откаже от затварянето.

При затваряне на документ трябва да се почистят всички данни свързани с него. Уверете се, че не допускате memory leaks.

```
> close
Successfully closed file.xml
> save
Cannot save – no document loaded
```

Създаване на празен документ (New)

Създава нов, празен документ. Той няма текущ файл. Ако искате да го запишете на диска, трябва да използвате `saveas`.

Записване в същия файл (Save)

Записва направените промени обратно в текущия файл на документа. Ако документът няма текущ файл, командата дава грешка.

```
> save
Successfully saved file.xml
```

Записване под ново име (SaveAs)

Позволява на потребителя да укаже файл, в който да се запишат данните за документа. След успешно изпълнение на тази команда, текущият файл на документа се променя. Текущ става файлът, чийто път е бил подаден на `saveas`.

```
> saveas "C:\Temp\another file.xml"
Successfully saved another file.xml
```

1: Електронни таблици

Представяне на данните

Данните на една таблица ще записваме в текстов файл (в CSV формат) по следния начин:

1. Всеки ред във файла представя отделен ред в таблицата.
2. Всеки ред във файла съдържа данни разделени със запетаи. Тези данни се интерпретират като стойностите в клетките на реда.
3. Всеки ред в таблицата може да съдържа различен брой клетки. Затова и всеки ред във файла може да съдържа различен брой елементи разделени със запетаи.
4. Празен ред във файла представя празен ред в таблицата. (т.е. ред, в който всички клетки са празни).
5. Между две запетаи във файла може да няма никакви данни. По този начин се представя празна клетка.
6. Между данните и запетаите може да има произволен брой празни символи (whitespace).

За една таблица може да има различни представяния. Например таблицата:

10	20	30	40
10		1000	
	10		

може да се представи по следните начини (възможни са и други представяния):

```
10, 20, 30, 40

10,,1000,
,,,
,10
```

```
10, 20, 30  , 40

10,   , 1000,
  ,   ,     ,
  , 10
```

Типове данни в таблицата

Всяка клетка в таблицата има тип, като в една таблица може да има едновременно клетки от различни типове. Вашето приложение трябва да може да поддържа следните типове:

Цяло число – поредица от цифри, без никакви други символи между тях. В началото на числото може да има знак '+' или '-'. Например:

```
123
-123
+123
```

Дробно число – поредица от цифри, следвана от символ за точка и след нея друга поредица от цифри. В началото на числото може да има знак '+' или '-'. Например:

```
123.456
-123.456
+123.456
```

Символен низ (стринг) – поредица от произволни символи оградени в кавички. Подобно на низовете в C++, ако искате да включите символа за кавичка в даден низ, трябва да го представите като \", а ако искате да включите наклонена черта, трябва да я представите като \\. Например:

```
"Hello world!"
"C:\\temp\\"
"\"This is a quotation\""
```

ВАЖНО: Забележете, че кавичките около символния низ играят роля само при прочитането на низа (или когато го четем от файла или при въвеждане с командата `edit`). Те не са част от самия низ.

Формула – формулата винаги започва със символ за равенство. В нея могат да участват следните операции: събиране (+), изваждане (-), умножение (*), деление (/) и степенуване (^). Във формулата могат да участват или числа или препратки към клетки в таблицата. Ако във формулата участва препратка към клетка, на това място в изчислението трябва да се използва стойността съхранена в дадената клетка. Повече информация за формулите е дадена по-долу.

Нужна функционалност

Колоните, редовете и клетките в таблицата ще обозначаваме по стандартния за този тип приложения начин:

- Колоните в таблицата се обозначават с буквите A - Z. За улеснение приемаме, че в нея не може да има повече от 26 колони.
- Редовете се обозначават с цели числа. Първият ред е с номер 1. Можем да имаме неограничен брой редове.
- Клетките се обозначават като пресечна точка на колона и ред. Например A11, B1, Z12345 и т.н. Изписването е case-insensitive. Например A1 и a1 обозначават една и съща клетка.

Ако при зареждането на данните, приложението ви открие проблем, то трябва да изведе подходящо съобщение за грешка и да прекрати операцията. Съобщението трябва да подскаже на потребителя какво не е наред във входните данни. Например:

- Ако съдържанието на дадена клетка е от неизвестен тип, трябва да се изведе на кой ред и коя колона е клетката и какво точно е некоректното съдържание. Например нека предположим, че на ред 2, колона 3, потребителят е въвел 123.123.123. Приложението ви може да изведе например следното съобщение: *“Error: row 2, col 3, 123.123.123 is unknown data type”*.

След като вашето приложение отвори даден файл, то трябва да може да извършва посочените по-долу операции:

Извеждане на таблицата на екрана (Print)

При извеждане, данните в колоните трябва да се подравнят. Между отделните колони трябва да се поставят символи за отвесна черта (|). Най-отгоре и най-вляво на таблицата да се посочат индексите на съответните редове и колони. По-долу е даден пример за входен файл и възможно негово извеждане:

Входен файл	Работа на командата

`10, "abc", 123.56`
`"\"Quoted\""`
`1, 2, 3, 400`
`,,A3+B3+C3`
	`> print`
	`  | A  | B  | C  | D  |`
	`1 |  10 | abc | 123.56 |  |`
	`2 |"Quoted" |  |  |  |`
	`3 |  1 |  2 |  3 | 400 |`
	`4 |  |  |  6 |  |`

Редактиране на клетки (Edit)

Командата позволява на потребителя да променя стойностите на отделните клетки. Аргументите са референция към клетка и нова стойност, която искаме да впишем в нея. Например:

```
> edit A2 123456
Successfully set A2 to 123456
```

Потребителят може да въведе произволен тип данни, който се поддържа от вашата програма (например цяло число, дробно число, низ, формула и т.н.). Забележете, че по този начин може да се промени типът на дадена клетка, например от число, тя може да стане формула или текст.

Ако потребителят въведе неправилни данни, приложението ви не трябва да променя нищо в таблицата, а само да изведе на екрана съобщение, че са въведени неправилни данни. В този случай приложението ви НЕ трябва да прекратява своето изпълнение, а просто да не променя клетката. Например:

```
> edit A2 12.34.56
Error: incorrect value 12.34.56
```

Потребителят може да въведе произволен номер на клетка. Ако тя излиза извън размерите на текущо заредената таблица, тя трябва автоматично да се разшири. По този начин се създават голямо количество нови клетки. Освен ако за тях не се задава конкретна стойност, те трябва да бъдат празни. Например, да предположим, че текущо заредената таблица съдържа 2 реда и 5 колони. Потребителят може да въведе команда:

```
> edit Z2000 "Hello world!"
Successfully set Z2000 to Hello world
```

След нея, таблицата ще бъде с размери 2000 реда и 26 колони. Клетката Z2000 ще бъде със стойност символния низ `Hello world`, а всички новодобавени клетки ще бъдат празни.

Формули

В дадена формула могат да участват единствено:

1. Литерали: цели или дробни числа.
2. Препратки към произволни типове клетки.

Формулите са символни низове, които започват със символа равно (`=`). В тях като минимум реализирайте операциите събиране, изваждане, умножение, деление и степенуване.

При сметките важат следните правила:

1. Ако в дадена формула участват само числа, то сметката се извършва по традиционните правила на аритметиката. Като специален случай можем да отделим делението на две цели числа. В такъв случай не бива да губите остатъка и резултатът трябва да бъде дробно число (например 1 делено на 2 дава резултат 0,5).

2. Ако в дадена формула участва низ, той трябва да се конвертира до число. Това става по следния начин: Ако низът съдържа само цифри или поредица от цифри, символ точка и друга поредица от цифри, той се конвертира до съответното число. Всички други низове се конвертират до нула. Например:

Низ	Конвертирана стойност
“123”	123
“123.456.789”	0
“123.456”	123.456
“Hello world”	0
“123abc”	0

3. Ако в дадена формула участва празна клетка, тя се конвертира до нула. Това важи и за клетки, чиито координати надхвърлят размерите на таблицата.

4. Ако в дадена формула има грешка (например деление на нула), приложението ви не трябва да прекъсва своето изпълнение. Вместо това, когато то извежда таблицата на екрана, в съответната клетка се извежда текст `#ERROR`.

По-долу е дадена примерна таблица. В нея клетките в жълт цвят са от тип число. Клетките в зелено са от тип символен низ:

	A	**B**	**C**
1	10	Hello world!	123.56
2	123		

По-долу са дадени формули, които се оценяват в примерната таблица по-горе. За всяка формула е дадена и нейната оценка:

Формула в клетката	**Реално извършена сметка**	**Стойност на клетката**	**Коментар**

= 10 + 10	10 + 10	20	
= A1 + C1	10 + 123.56	133.56	
= A1 * B1	10 * 0	0	Низът „Hello world!“ се конвертира до нула
= A1 * A2	10 * 123	1230	Низът „123“ се конвертира до 123.
= A1 * B2	10 * 0	0	Клетката B2 е празна
= A1 * Z1000	10 * 0	0	В таблицата няма ред 1000, нито колона Z. Считаме, че клетката Z1000 е празна.
= 10 / 0	10 / 0	ERROR	
= 10 / B2	10 / 0	ERROR	
= A1 / B2	10 / 0	ERROR	