Работа с командния ред

Следните условия за работа с команден ред **важат за всички теми за проекти**. Моля, запознайте се внимателно с тях.

Вашата програма трябва да позволява на потребителя да отваря файлове (`open`), да извършва върху тях някакви операции, след което да записва промените обратно в същия файл (`save`) или в друг, който потребителят посочи (`saveas`). Трябва да има и опция за затваряне на файла без записване на промените (`close`).

За да предостави на потребителя възможност да укаже каква операция да се изпълни, програмата трябва да работи в **интерактивен команден режим**: при стартиране, тя подканва потребителя да въведе едноредови команди и след това в зависимост от въведените команди да ги изпълнява. Командите могат да имат нула, един или повече параметри, които се изреждат един след друг, разделени с един или няколко интервала. Някои от командите може да изискват допълнително въвеждане на информация при изпълнението си.

При отваряне на даден файл, неговото съдържание трябва да се зареди в паметта, след което файлът се затваря. Всички промени, които потребителят направи след това трябва да се пазят в паметта, но не трябва да се записват обратно, освен ако потребителят изрично не укаже това. При изход, програмата трябва да подсеща потребителя дали желае да запише последните промени във файла. Подсещането да се извежда само ако след последното записване е изпълнена операция, която променя данните в паметта.

Бонус: Ако желаете, можете да реализирате възможност, при която текущото състояние на паметта се записва във временен файл за възстановяване (recovery file) като защита при евентуално прекратяване на програмата при грешка. При отваряне на файл, програмата да проверява за наличие на файл за възстановяване и да предлага на потребителя да избере дали данните от него да бъдат възстановени или пренебрегнати.

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

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

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

Дадените по-долу команди трябва да се поддържат от всеки от проектите. Под всяка от тях е даден пример за нейната работа:

Open

Зарежда съдържанието на даден файл. Ако такъв не съществува, се създава нов с празно съдържание.

След като файлът бъде отворен и се прочете, той се затваря и програмата вече не трябва да работи с него, освен ако потребителят не поиска да запише обратно направените промени (както е указано в командите **Save** и **Save As** по-долу), в който случай файлът трябва да се отвори отново за запис.

Ако при зареждането на данните, програмата открие грешка, тя трябва да изведе подходящо съобщение за грешка и да прекрати своето изпълнение.

Пример:

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

Successfully opened file.xml
```

Close

Затваря текущо отворения файл. Командата се изпълнява успешно само ако има текущ отворен файл. Затварянето изчиства текущо заредената информация и след това програмата не може да изпълнява други команди, освен отваряне на файл (Open).

Пример:

```
> close

Successfully closed file.xml
```

Save

Записва направените промени обратно в същия файл, от който са били прочетени данните. Командата се изпълнява успешно само ако има текущ отворен файл.

Пример:

```
> save

Successfully saved file.xml
```

Save As

Записва направените промени във файл, като позволява на потребителя да укаже неговия път. Командата се изпълнява успешно само ако има текущ отворен файл.

Пример:

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

Successfully saved another file.xml
```

Help

Извежда кратка информация за поддържаните от програмата команди.

Пример:

```
> help

The following commands are supported:

open <file>           opens <file>

close                 closes currently opened file

save                  saves the currently open file

saveas <file>         saves the currently open file in <file>

help                  prints this information

exit                  exists the program
```

Exit

Излиза от програмата. Ако има незаписани промени, предлага на потребителя да избере затваряне с пренебрегване на промените (Close) или записване (Save или Save As).

```
> exit

You have an open file with unsaved changes, please select close or save
first.

> close

Successfully closed file.xml

> exit

Exiting program...
```

Тема 8: SVG файлове

В рамките на този проект трябва да се разработи програма, която работи със файлове, представени чрез подмножество на формата Scalable Vector Graphics (SVG). Програмата трябва да може да зарежда фигури от файла, да извършва върху тях дадени операции, след което да може да записва промените обратно на диска.

За улеснение, в рамките на проекта ще работим само с основните фигури (basic shapes) в SVG. Приложението ви трябва да поддържа поне три от тях. Например, можете да изберете да се поддържат линия, кръг и правоъгълник. За повече информация за това кои са базовите фигури, вижте https://www.w3.org/TR/SVG/shapes.html.

Също така, за улеснение считаме, че координатната система, в която работим е тази по подразбиране: положителната полуос X сочи надясно, а положителната полуос Y сочи надолу.

Дизайнът на програмата трябва да е такъв, че да позволява при нужда лесно да може да се добави поддръжка на нови фигури.

Когато съдържанието на един SVG файл се зареди, трябва да се прочетат само фигурите, които програмата поддържа, всички останали SVG елементи могат да се игнорират.

След зареждане на фигурите, потребителят трябва да може да изпълнява дадените в следващия раздел команди, които добавят, изтриват или променят фигурите.

Записаният от програмата файл трябва да е валиден файл във формата SVG, който да може коректно да се отваря от други програми (напр. Браузър).

Операции

След като приложението отвори даден файл, то трябва да може да извършва посочените по-долу операции, в допълнение на общите операции (`open`, `close`, `save`, `saveas`, `help` и `exit`):

`print`	Извежда на екрана всички фигури.
`create`	Създава нова фигура.
`erase <n>`	Изтрива фигура с пореден номер `<n>`.
`translate [<n>]`	Транслира фигурата с пореден номер `<n>` или всички фигури, ако `<n>` не е указано.
`within <option> ...`	Извежда на екрана всички фигури, които изцяло се съдържат в даден регион. Потребителят може да укаже чрез `<option>` какъв да бъде регионът – кръг (`circle`) или правоъгълник (`rectangle`)

Пример:

Примерен SVG файл (figures.svg)

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg>
  <rect x="5" y="5" width="10" height="10" fill="green" />
  <circle cx="5" cy="5" r="10" fill="blue" />
  <rect x="100" y="60" width="10" height="10" fill="red" />
</svg>
```

Пример за работа на програмата

```
> open figures.svg
Successfully opened figures.svg

> print
1. rectangle 5 5 10 10 green
2. circle 5 5 10 blue
3. rectangle 100  60 10 10 red

> create rectangle -1000 -1000 10 20 yellow
Successfully created rectangle (4)

> print
1. rectangle 1 1 10 20 green
2. circle 5 5 10 blue
3. rectangle 100 60 10 10 red
4. rectangle 1000 1000 10 20 yellow

> within rectangle 0 0 30 30
1. rectangle 5 5 10 10 green
2. circle 5 5 10 blue

> within circle 0 0 5
No figures are located within circle 0 0 5

> erase 2
Erased a circle (2)

> erase 100
There is no figure number 100!

> print
1. rectangle 5 5 10 10 green
```

```
2. rectangle 100 60 10 10 red
3. rectangle 1000 1000 10 20 yellow

> translate vertical=10 horizontal=100
Translated all figures

> print
1. rectangle 105 15 10 10 green
2. rectangle 200 70 10 10 red
3. rectangle 1100 1010 10 20 yellow

> save
Successfully saved the changes to figures.svg

> exit

Exit
```