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

Теми за проекти

курс Обектно-ориентирано програмиране
за специалности Информатика, Компютърни науки и Софтуерно инженерство
Летен семестър 2016/2017 г.

Обща информация за проектите

Проектите се оценяват по редица от критерии, част от които са описани по-долу. Тъй като курсът се фокусира върху обектно-ориентираното програмиране и неговата реализация в езика C++, най-важното изискване за проектите е те да са изградени съгласно добрите принципи на ООП. Решението, в което кодът е процедурен, има лоша ООП архитектура и т.н. се оценява с нула точки. Други важни критерии за оценка на проектите са:

- Дали решението работи коректно, съгласно спецификацията. Решение, което не работи и/или не се компилира носи минимален брой (или нула) точки.
- Дали решението отговаря на заданието на проекта.
- Каква част от необходимата функционалност е била реализирана.
- Дали решението е изградено съгласно добрите практики на обектно-ориентирания стил. Тъй като курсът се фокусира върху ООП, решения, които не са обектно-ориентирани се оценяват с нула или минимален брой точки.
- Оформление на решението. Проверява се дали кодът е добре оформен, дали е спазена конвенция за именуване на променливите, дали е добре коментиран и т.н.
- Дали решението е било добре тествано. Проверява се какви тестове са били проведени върху приложението, за да се провери дали то работи коректно. Очаква се по време на защитата да можете да посочите как сте тествали приложението, за да проверите дали то работи коректно и как се държи в различни ситуации.

По време на защитата се очаква да можете да отговорите на различни въпроси, като например: (1) каква архитектура сте избрали, (2) защо сте избрали именно нея, (3) дали сте обмислили други варианти и ако да — кои, (4) как точно работят различните части от вашия код и какво се случва на по-ниско ниво и др.

Оценката на всеки от проектите се формира от онази негова част, която е била самостоятелно разработена от вас. Допустимо е да използвате код написан от някой друг (напр. готова библиотека или помощ от ваш приятел/колега), но (1) той не носи точки към проекта и (2) това трябва да бъде ясно обявено както при предаването, така и при защитата на проекта, като ясно обозначите коя част от проекта сте разработили самостоятелно. Това означава, че:

1. Използваният наготово код трябва да се маркира ясно, като поставите коментари на подходящи места в кода си.
2. По време на защитата трябва да посочите кои части сте разработили самостоятелно и кои са взети от други източници.

Както е написано по-горе, когато в проекта си използвате чужд код, сам по себе си той не ви носи точки. Допълнителни точки могат да се дадат или отнемат, според (1) способността ви за внедряване на кода във вашето решение (напр. в случаите, когато се използва външна библиотека) и за това (2) дали добре разбирате какво прави той.

Начин на работа

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 (save as). Трябва да има и опция за затваряне на файла, без записване на промените (close). За целта, когато програмата ви се стартира, тя трябва да позволява на потребителя да въвежда команди и след това да ги изпълнява.

Когато отворите 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.

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.

Както казахме по-горе, потребителят трябва да може да въвежда команди, чрез които да посочва какво трябва да се направи. Командите могат да имат нула, един или повече параметри, които се изреждат един след друг, разделени с интервали.

Освен ако не е казано друго, всяка от командите 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.

Всички останали команди могат да се изпълняват само ако има успешно зареден файл.

След като файлът бъде отворен и се прочете, той се затваря и приложението ви вече не трябва да работи с него, освен ако потребителят не поиска да запише обратно направените промени (вижте командата save по-долу), в който случай файлът трябва да се отвори наново. За целта трябва да изберете подходящо представяне на информацията от файла.

Ако при зареждането на данните, приложението ви открие грешка, то трябва да изведе подходящо съобщение за грешка и да прекрати своето изпълнение.

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

Close

Затваря текущо отворения документ. Затварянето изчиства текущо заредената информация и след това програмата не може да изпълнява други команди, освен отваряне на файл (Open).

```
> close
Successfully closed file.xml
```

Save

Записва направените промени обратно в същия файл, от който са били прочетени данните.

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

Save As

Записва направените промени във файл, като позволява на потребителя да укаже неговия път.

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

Exit

Излиза от програмата

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

Проект: Работа със SVG файлове

В рамките на този проект трябва да се разработи приложение, което работи със файлове във Scalable Vector Graphics (SVG) формат. Приложението 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 приложението трябва да е такъв, че да позволява при нужда лесно да можете да добавите поддръжка на нови фигури.

Когато зареждате съдържанието на един SVG файл, трябва да прочетете само фигурите, които приложението ви поддържа и можете да игнорирате всички останали SVG елементи.

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

Когато записвате фигурите във файл, трябва да генерирате валиден SVG файл

Операции

Print	Извежда на екрана всички фигури.
Create	Създава нова фигура.
Erase	Изтрива фигура
Translate	Транслира една или всички фигури. Ако потребителят не посочи конкретна фигура, тогава се транслират всички фигури; ако се посочи конкретна – променя се само тя.
Within	Извежда на екрана всички фигури, които изцяло се съдържат в даден регион. Потребителят може да укаже какъв да бъде регионът – кръг или правоъгълник

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
```