Софийски университет “Св. Климент Охридски”

Факултет по математика и информатика

Курсов проект по ООП

летен семестър 2019/2020

**Проект №2**

**Работа със SVG файлове**

**От**

**Димитър Сотиров**

**Специалност: Информационни системи 1-ви курс**

**ФН:71983**

**Глава 1. Увод**

* 1. Описание и идея на проекта

Тази програма е един вид SVG (scalable vector graphics) image converter. Идеята е клиентът да се запознае с удобствата от ползването на формата SVG и така популярността му да нарасне. Разработката прави боравенето с този формат значително по-лесно и превръща работата в удоволствие.

* 1. Цел и задачи на разработката

Задачата на проекта е широката общественост да се запознае с удобствата на формата, главните му свойства, представени по един по-лесен начин и да се убедят, че той превъзхожда широко ползваните формати jpg и png.

За цел, разработката има именно достигането до колкото се може повече хора и популяризирането на формата.

* 1. Структура на документацията

Глава 1. Увод

Глава 2. Преглед на предметната област

Глава 3. Проектиране

Глава 4. Реализация, тестване

Глава 5. Заключение

**Глава 2. Преглед на предметната област**

2.1. Основни дефиниции и концепции, които ще бъдат използвани

2.1.1 Основни дефиниции:

„converter“ – конвертор

„чисто виртуална функция“ – функция, чиято имплементация е задължителна в дъщерните класове

„абстрактен клас“ – клас, който съдържа виртуални функции. Обекти от този тип не могат да се създават, само указатели.

„stack“ – статичната памет на компютъра.

“heap” – динамичната памет на компютъра.

„memory leak“ – загуба на памет. Основна причина за загубата на памет е невнимателното ползване на „heap“-a.

„голяма четворка в обектно-ориентираното програмиране“ – конструктор по подразбиане, копи конструктор, оператор = , деструктор

„капсулация“ – access modifier-a на член данните. Показва нивото на достъпване на полетата.

2.1.2 Концепции, които ще бъдат използвани

За постигане на оптимално решение са ползвани основните принципи на обектно ориентираното програмиране – капсулация, наследяване, абстракция и полиморфизъм.

2.2. Дефиниране на проблеми и сложност на поставената задача

2.2.1 Гореспоменатият “memory leak” е основен проблем при разработката на голям проект, съобразяващ се с управлението на памет.

2.2.2 Друг основен проблем е правилното капсулиране на член данните.

2.2.3 Невнимание и създаване на обект от абстрактния клас

2.2.4 Видове наследяване

2.2.5 Разпознаване на типове в хетерогенен контейнер (Vector<Figure\*> figures)

2.2.6 При изпълнението на задачата много трябва да се внимава с множеството „подводни камъни“ и потенциални проблеми, които могат да се появят при невнимание.

2.3. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите

2.3.1 За предотвратяването на “memory leak” заделяме точно толкова динамична памет, колкото ни е необходима. При всяка промяна изтриваме заделената преди това памет и заделяме нова, съобразено с това, за което заделяме и колко място заема то.

2.3.2 Слагайки коректни access modifier-и на различните полета в класа, спазваме един от основните принципи на обектно-ориентираното програмиране, а именно капсулацията.

2.3.3 При внимателна работа с проекта не допускаме създаването на обект от абстрактния клас, тъй като така нарушаваме трети принцип на ООП – абстракцията.

2.3.4 Различните видове наследяване са нещо, което може лесно да се обърка, особено при междувременна смяна на access modifier-ите на полетата в класа (protected -> private, например). Много внимателно проверяваме видовете наследявания, за да не загубим достъп до поле, което после се достъпва от някаква функция.

2.3.5 Внимателно работим с контейнера, защото е в сила принципа: Всеки кръг е фигура, но не всяка фигура е кръг. Именно заради това, векторът е от тип Figure\*.

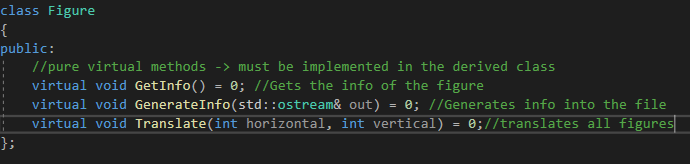
**Глава 3. Проектиране**

3.1. Обща архитектура

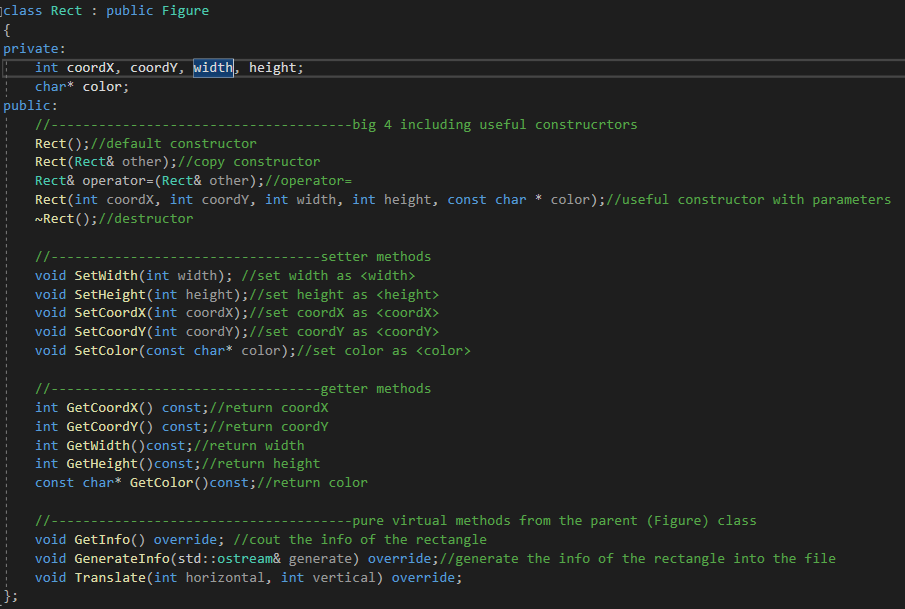
Системата съдържа два основни класа: class Figure и class MainCommands. Класът class Figure е абстрактен клас, който се явява родител на класовете class Circle, class Line, class Rect, т.е тези дъщерни три класа наследяват от class Figure. Връзката с файловете се получава благодарение на class MainCommands и неговите методи за обработка на файлове.

При старт програмата ще иска от клиента да подаде команда и в зависимост от тази команда ще бъде извикана коректната функция, отговаряща за правилната работа на програмата. Ако програмата се ползва за пръв път, препоръчително е да се напише първо командата „help“, която ще покаже на клиента кои са поддържаните от програмата команди.

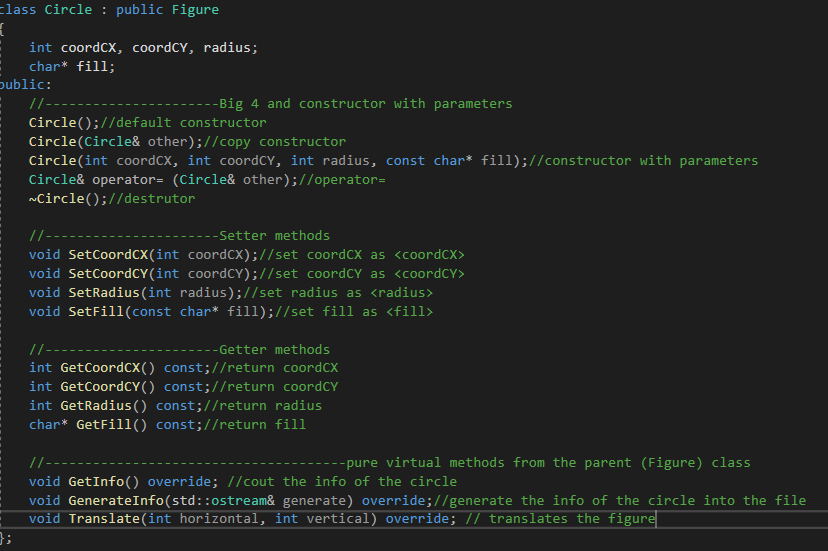
3.2. Диаграми и визуализация на програмата

 Фиг. 1.1

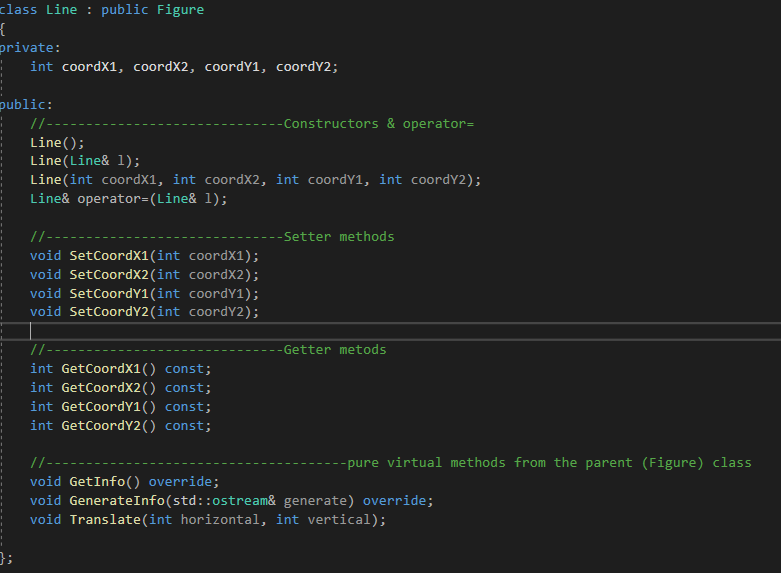
На фиг. 1.1 виждаме абстрактния parent клас Figure, съдържащ единствено чисто виртуални функции.

 Фиг. 1.2

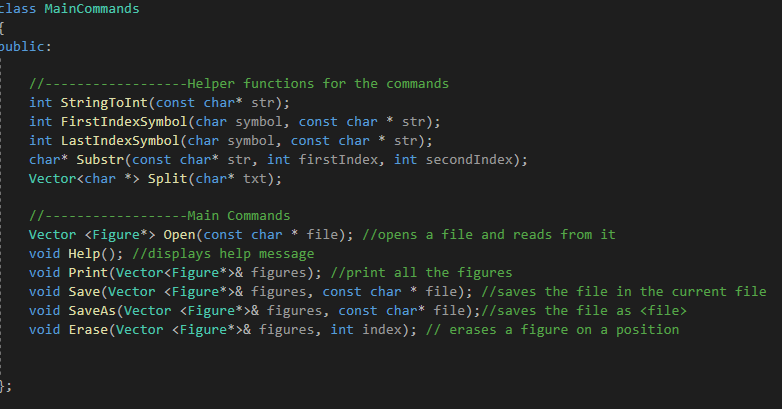
На фиг. 1.2 може да се видят методите на дъщерният class Rect.

 Фиг. 1.3

На фиг. 1.3 може да се видят методите на дъщерният class Circle.

 Фиг. 1.4

На фиг. 1.4 може да се видят методите на дъщерният class Line.

 Фиг. 1.5

На фиг. 1.5 е изобразен class MainCommands , съдържащ основните функции на програмата

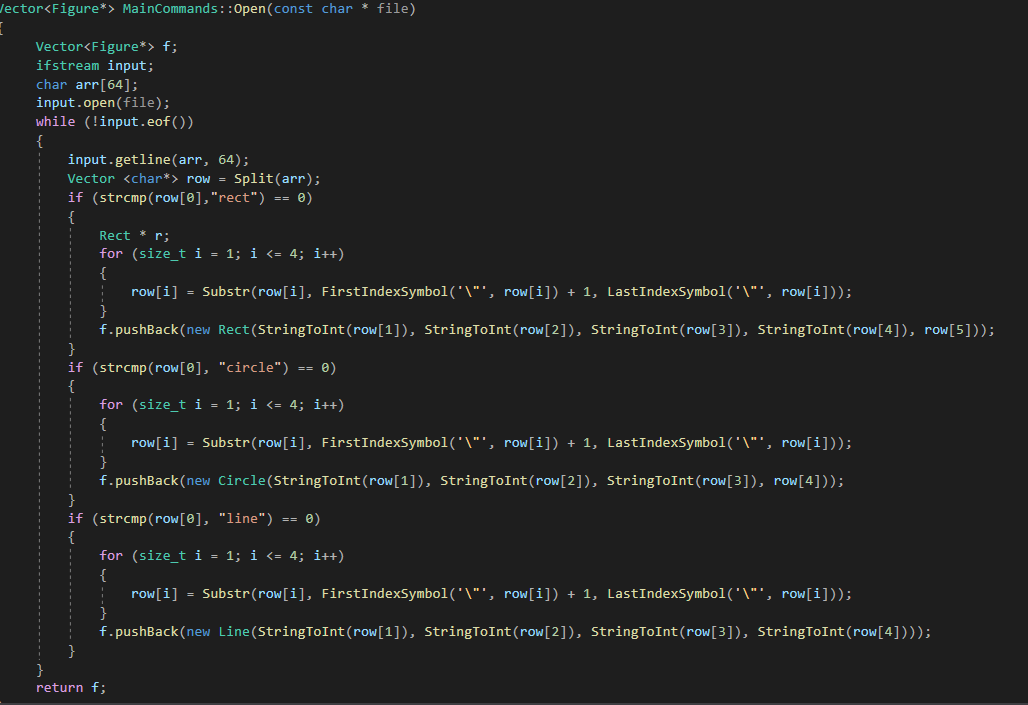
**Глава 4. Реализация, тестване**

4.1. Реализация на класове

class MainCommands

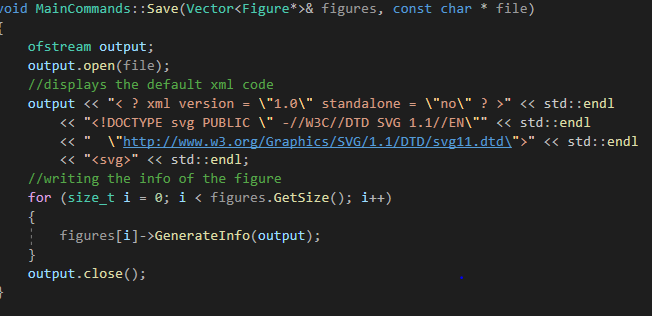
Основната трудност в този клас беше методът Open, който чете xml код или друга информация от файл и я превръща в разбираема за човека информация.

Реализацията на методът Open:

 Фиг. 2.1

На фиг. 2.1 виждаме алгоритъма за четене на xml код от файл и разпознаването на различните фигури в кода и добавянето им към хетерогенния контейнер f. В зависимост от прочетеното, добавяме към контейнера фигура от прочетения тип с прочетените параметри

Следващият основен метод, който ще погледнем, е void MainCommands::Save(Vector<Figure\*>& figures, const char \* file). Той отговаря за запазването на промените, направени по време на ползването на програмата, в отворения файл, генерирайки xml код според това, какви са фигурите във figures.

 Фиг. 2.2

На фиг. 2.2 виждаме реализацията на основен компонент от системата,а именно процесът на запазване във файл. След като сме направили желаните промени, ги записваме във файла, който сме отворили.

4.2. Управление на паметта

Поради естеството на задачата, не се налага широкото ползване на динамична памет.

4.3. Тестови сценарии

Препоръчителен алгоритъм на ползване на програмата:

Ако клиентът за пръв път пуска програмата, е хубаво да напише първо командата „help“, за да види кои команди се поддържат от системата.

След това може клиентът да създава нови фигури, да ги транслира , както и да ги изтрива.

**Глава 5. Заключение**

Като заключение, смятам, че разработката е една доста добра платформа за полуляризация на формата и вярвам, че целта за достигане на масовост ще бъде постигната много бързо след пускането на програмата за ползване.

**Използвана литература:**

1. <https://www.geeksforgeeks.org/>
2. <https://www.cplusplus.com/>
3. <https://stackoverflow.com/>
4. <https://en.cppreference.com/w/>
5. <https://www.w3schools.com/>