ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧЕРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий
Высшая школа программной инженерии

Отчет по курсовой работе по дисциплине «Конструирование программного обеспечения»

Выполнила студентка гр. 3530904/80101 Прохорова А. И.

Руководитель Иванов А. С.

Санкт-Петербург
2021

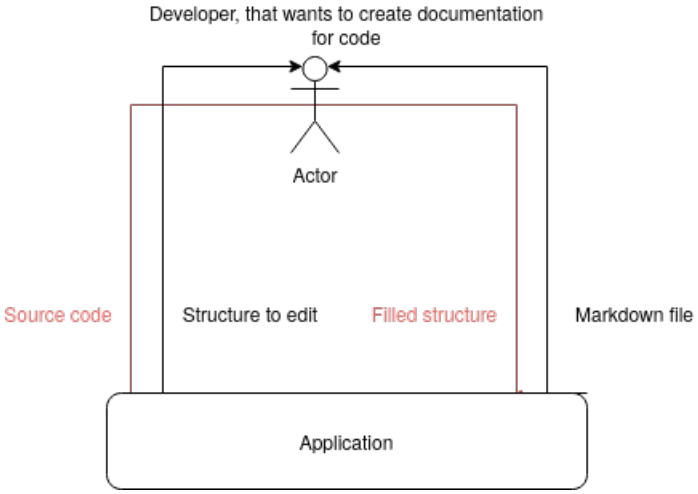
1 Задача

Создание приложения с графическим интерфейсом, упрощающее создание документации к исходному коду: в результате парсинга должны выделяться логически значимые единицы (классы, функции), создаваться древовидная структура, которую можно интерактивно изменить, заполнить и сконвертировать в markdown файл.

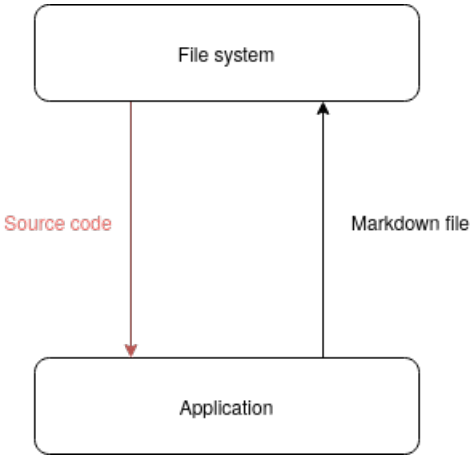
2 Требования

- Как пользователь, я не хочу вручную выписывать названия классов и функций из кода, а получать их в готовом виде, чтобы тратить меньше времени на документацию.
- Как пользователь, я хочу иметь возможность редактировать те названия, которые получились в результате автоматического парсинга.
- Как пользователь, я хочу к текущему списку классов и функций добавлять свои, если какие-то были пропущены в процессе парсинга и удалять те, которые мне кажутся неважными, для каждого элемента добавлять описание, чтобы получить наиболее удовлетворяющую меня документацию.
- Как пользователь, из древовидной структуры классов и функций с описаниями я хочу получить читаемый markdown файл.

3 System Context diagram



4 Container diagram



5 Структура проекта

Программа состоит из двух частей - графическое приложение и статическая библиотека, реализующая парсинг исходного кода. Для статической библиотеки parser написаны unit и интеграционные тесты.

6 Сборка и запуск

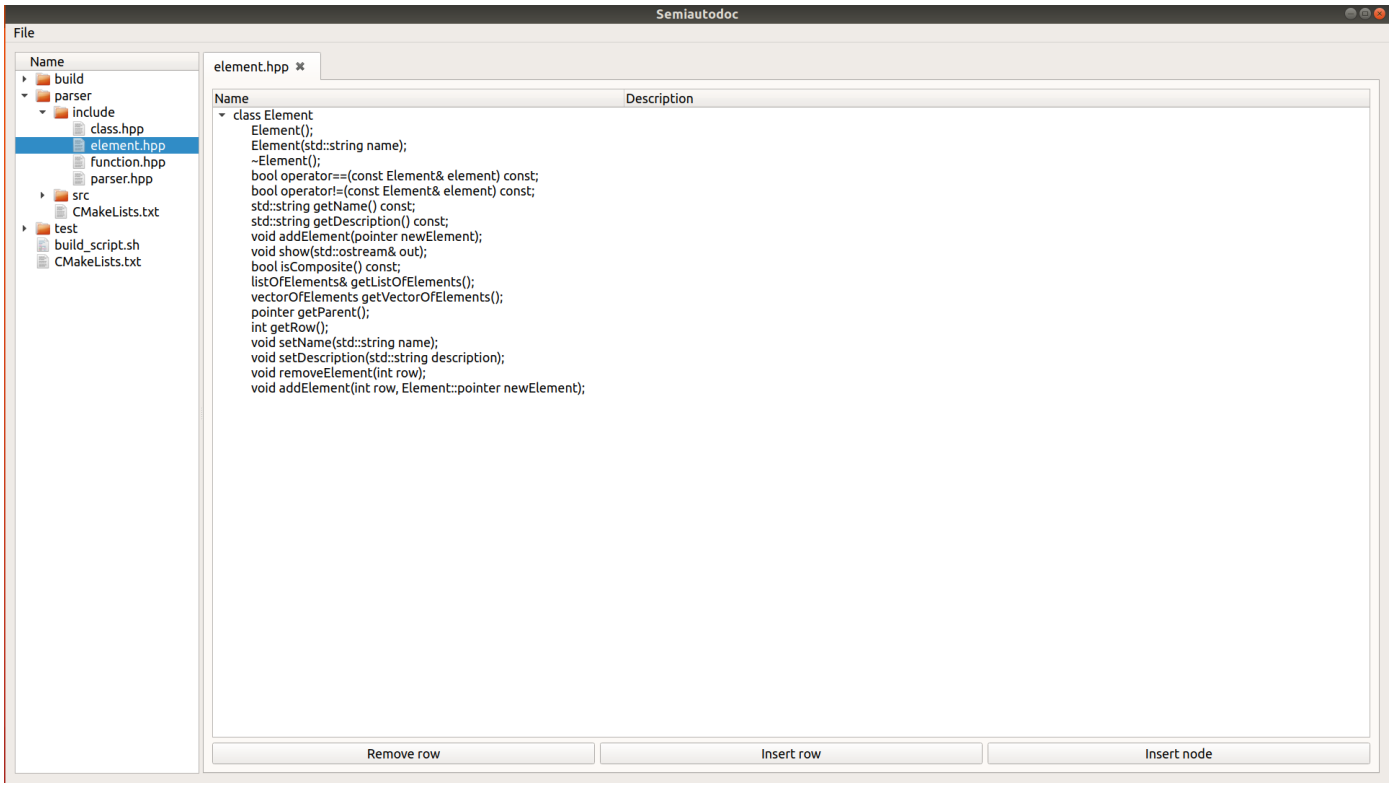
cd semiautodoc
qmake semiautodoc.pro && make
./build/release/semiautodoc

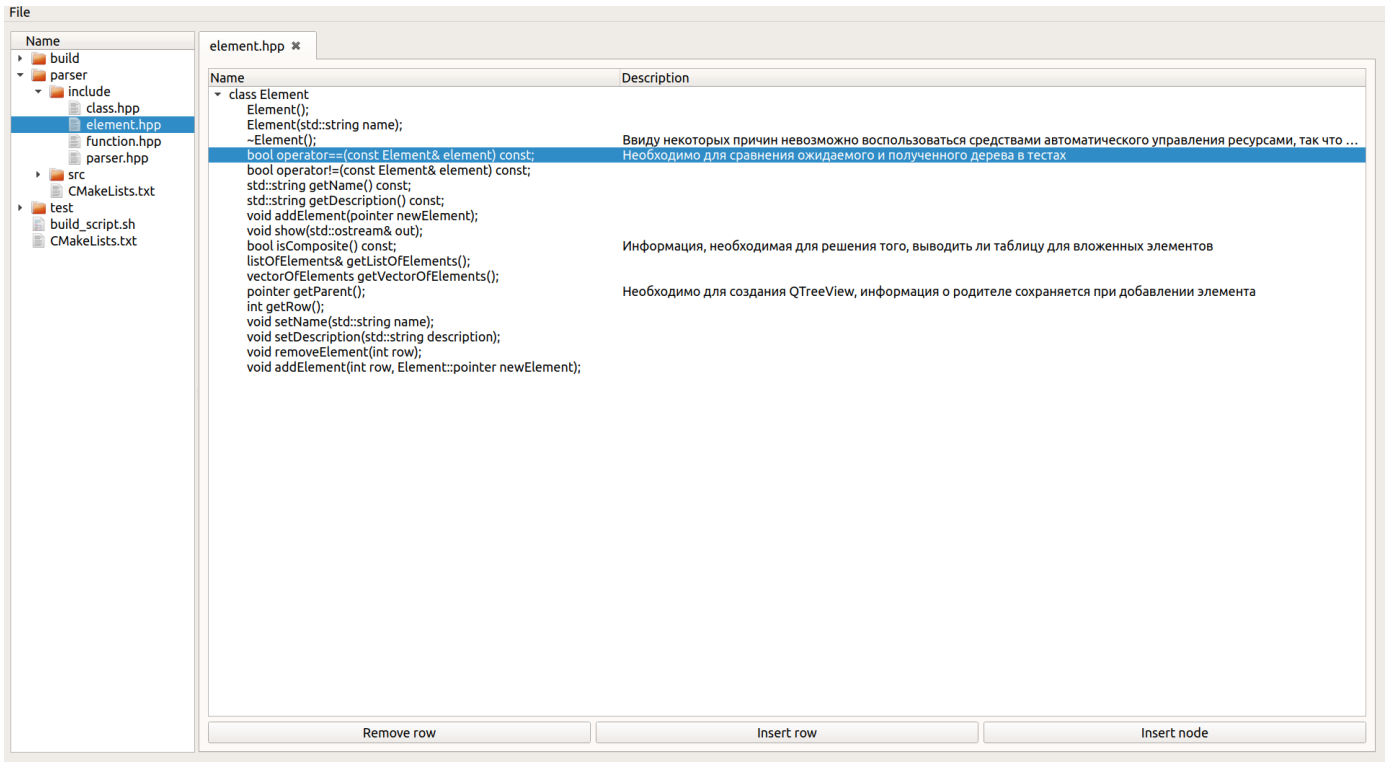
6.1 Запуск тестов для parser

cd semiautodoc/parser/build
make gtest
make
make test

Unit тесты находятся в папке parser/test, интеграционные тесты - в файле parser/test/test-parser.cpp.

6.2 Пример работы программы





22 lines (21 sloc) | 1.36 KB Raw Blame

class Element

Name	Description
Element();	
Element(std::string name);	
~Element();	Ввиду некоторых причин невозможно воспользоваться средствами автоматического управления ресурсами, так что освобождать память приходится вручную
bool operator==(const Element& element) const;	Необходимо для сравнения ожидаемого и полученного дерева в тестах
bool operator!=(const Element& element) const;	
std::string getName() const;	
std::string getDescription() const;	
void addElement(pointer newElement);	
void show(std::ostream& out);	
bool isComposite() const;	Информация, необходимая для решения того, выводить ли таблицу для вложенных элементов
listOfElements& getListOfElements();	
vectorOfElements getVectorOfElements();	
pointer getParent();	Необходимо для создания QTreeView, информация о родителе сохраняется при добавлении элемента
int getRow();	
void setName(std::string name);	
void setDescription(std::string description);	
void removeElement(int row);	
void addElement(int row, Element::pointer newElement);	

6.3 Вывод

В результате работы над курсовым проектом, было создано графическое приложение для создания документации.