

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

Программа для выделения синтаксиса алгоритмических языков при чтении лекций

Научный руководитель:
Доцент каф. ЭВМ к.т.н.
Парасич В.А.

Автор работы: студент
группы КЭ-405
Усков Ярослав Игоревич

Челябинск, 2024 г.

Актуальность темы

Подсветка синтаксиса является важным инструментом в обучении программированию. Она улучшает восприятие и понимание кода, облегчает поиск ошибок и повышает качество написания кода. Особенно актуальна эта тема в контексте чтения лекций и демонстрации примеров программирования. Кроме того, программа должна быть оптимизирована для работы на компьютерах разной мощности, что необходимо для использования в различных аудиториях университета.

Цели и задачи

Цель: Разработка программы для выделения синтаксиса алгоритмических языков программирования при чтении лекций.

Задачи:

1. Провести аналитический обзор аналогов и научной литературы.
2. Определить требования к программе.
3. Разработать архитектуру программы.
4. Реализовать программу.
5. Провести тестирование программы.

Аналитический обзор аналогов

```
xla_compilation_cache.cc x
190     const tensorflow::Value& value = variable_args[variable_id].value;
191     arg.type = value.dtype();
192     TF_RETURN_IF_ERROR(
193         TensorShapeToXLAShape(value.dtype(), value.shape(), &arg.shape));
194     arg.initialized = true;
195 } else {
196     // The values of uninitialized variables are not passed as inputs, since
197     // they are meaningless. However, it is legal to assign to a resource
198     // variable for the first time inside the XLA computation, so we do permit
199     // uninitialized variables.
200     arg.initialized = false;
201     arg.type = DT_INVALID;
202     arg.shape = xla::Shape();
203 }
204 ++input_num;
205 }
206
207 return Status::OK();
208 }
209
210 } // namespace
211
212 Status XlaCompilationCache::Compile(
213     const XlaCompiler::Options& options, const NameAttrList& function,
214     int num_constant_args, const std::vector<OptionalTensor>& variable_args,
215     OpKernelContext* ctx,
216     const XlaCompiler::CompilationResult** compilation_result,
217     xla::LocalExecutable** executable) {
218     VLOG(1) << "XlaCompilationCache::Compile " << DebugString();
219
220     if (VLOG_IS_ON(2)) {
221         VLOG(2) << "num_inputs=" << ctx->num_inputs()
222             << " num_constant_args=" << num_constant_args
223             << " num_variable_args=" << variable_args.size();
224         for (int i = 0; i < ctx->num_inputs(); i++) {
225             TensorShape shape = ctx->input(i).shape();
226             VLOG(2) << i << ": dtype=" << DataTypeString(ctx->input_dtype(i))
227                 << " present=" << ctx->has_input(i)
228                 << " shape=" << shape.DebugString();
229         }
230         for (const OptionalTensor& variable : variable_args) {
231             VLOG(2) << "variable present=" << variable.present
232                 << " type=" << DataTypeString(variable.value.dtype())
233                 << " shape=" << variable.value.shape().DebugString();
234         }
235     }
236 }
```

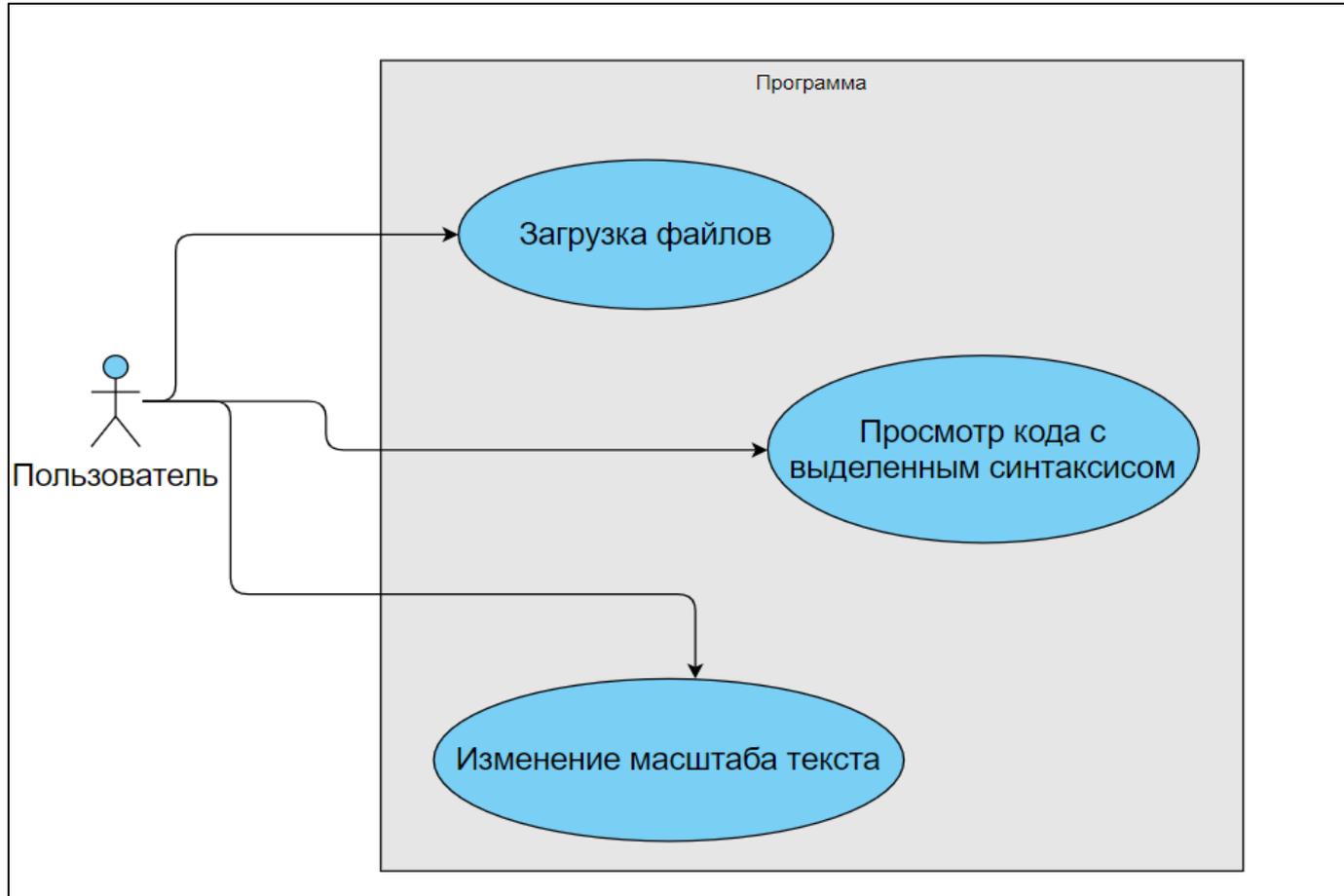
Line 212, Column 8

Spaces: 2 C++

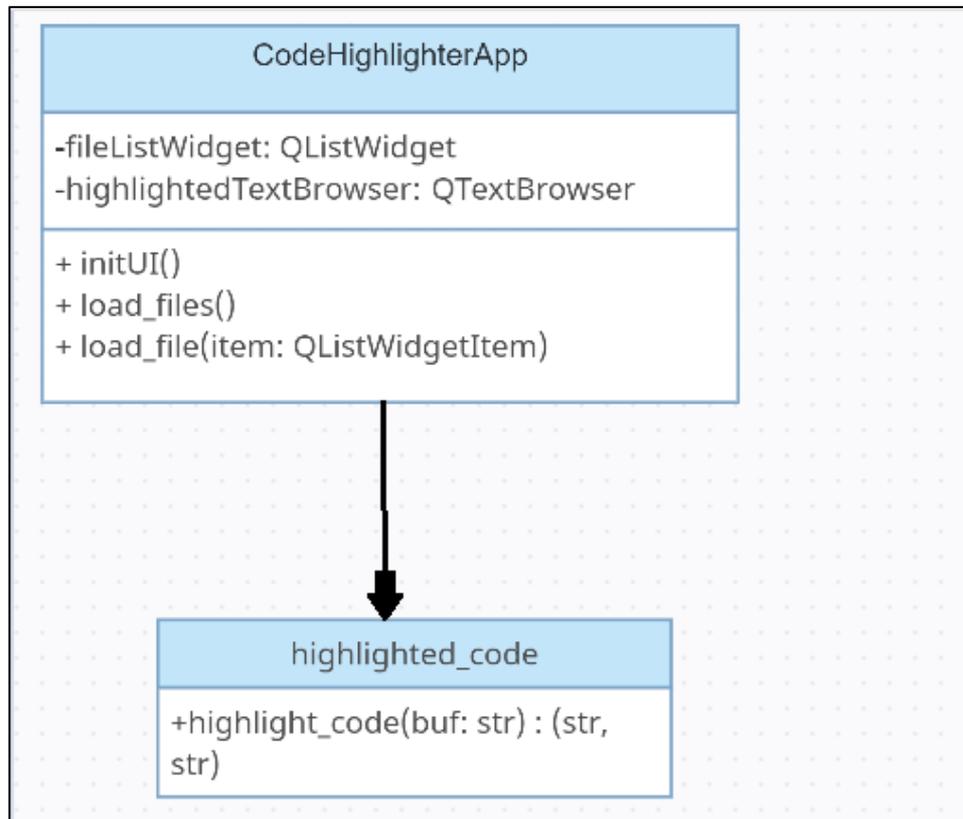
Аналитический обзор аналогов

Название приложения	Поддержка множества языков программирования	Поддержка больших файлов	Бесплатное использование	Высокая скорость работы	Удобство использования во время чтения лекций
Sublime Text	+	+	-	+	-
Visual Studio Code	+	-	+	-	-
Pycharm	+	+	Частично	-	-
Notepad++	+	+	+	+	-
Atom	+	-	+	-	-

Диаграмма вариантов использования



Архитектура программы



Архитектура программы

Архитектура программы включает:

- Главный класс приложения (`CodeHighlighterApp`), который отвечает за создание основного окна приложения.
- Функцию подсветки кода (`highlight_code`), использующую библиотеку `Pygments`.
- Инициализацию интерфейса (`initUI`), создающую и настраивающую элементы интерфейса пользователя.

Компоненты взаимодействуют следующим образом:

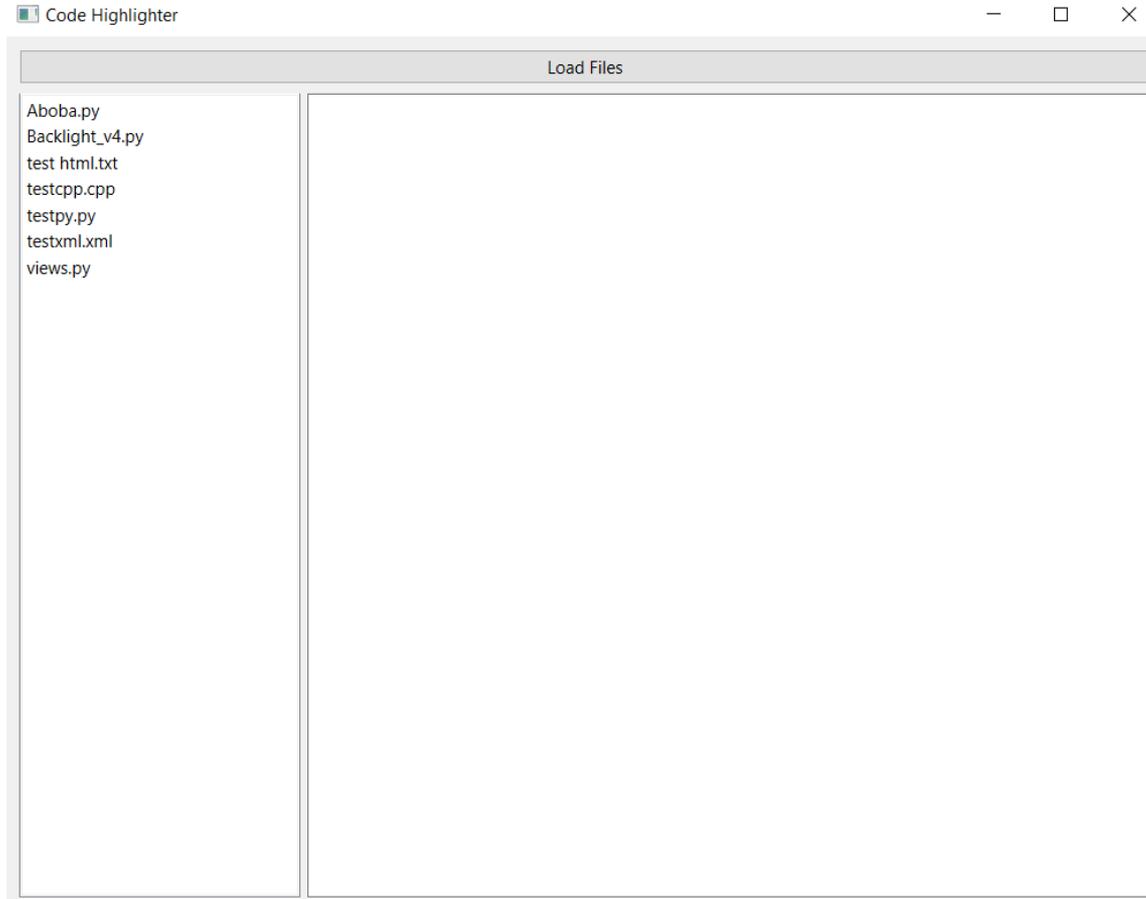
- Инициализация интерфейса (`initUI`) создает пользовательский интерфейс.
- Пользователь загружает файлы через метод `load_files`.
- Содержимое файлов загружается и передается функции `highlight_code` для применения выделения синтаксиса.
- Результат отображается в текстовом браузере (`QTextBrowser`).

Реализация программы

Программа была разработана с использованием Python и библиотек PyQt6 и Pygments. PyQt6 используется для создания графического интерфейса, а Pygments — для подсветки синтаксиса. Главное окно создается с использованием класса QMainWindow, а метод `initUI` создает и настраивает виджеты, такие как `QListWidget` и `QTextBrowser`.

Функция `highlight_code` определяет язык программирования с помощью Pygments, применяет подсветку синтаксиса и форматирует результат в виде HTML-кода. HTML-код с подсветкой и стили CSS отображаются в текстовом браузере.

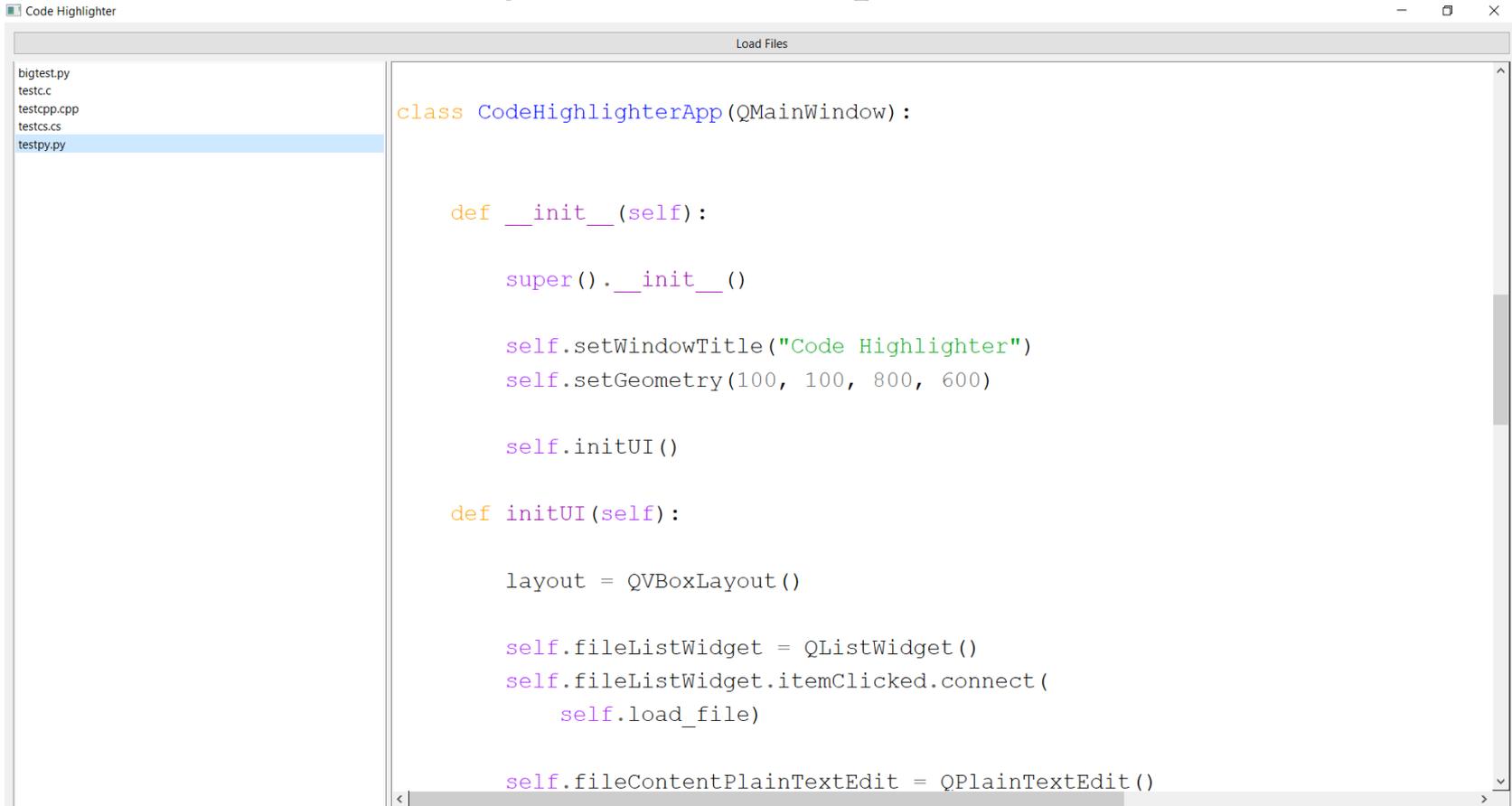
Реализация программы



Функциональное тестирование

№	Название теста	Ожидаемый результат	Тест пройден?
1	Загрузка файлов	Программа должна отображать в проводнике файлы с расширением. Программа выбирает необходимый файл и загружает его.	Да
2	Подсветка синтаксиса для Python	Программа применяет подсветку синтаксиса для файлов на языке Python, отображает текст с подсветкой синтаксиса.	Да
3	Подсветка синтаксиса для C++	Программа применяет подсветку синтаксиса для файлов на языке C++, отображает текст с подсветкой синтаксиса.	Да
4	Отображение больших файлов	Программа должна корректно отображать большие файлы без значительных задержек и ошибок.	Да
5	Масштабирование текста	Программа позволяет пользователю изменять размер текста, и изменения применяются ко всему отображаемому тексту.	Да

Результаты тестирования



The screenshot shows a window titled "Code Highlighter" with a "Load Files" header. On the left, a file list contains: bigtest.py, testc.c, testcpp.cpp, testcs.cs, and testpy.py. The "testpy.py" file is selected. The main editor area displays the following Python code:

```
class CodeHighlighterApp(QMainWindow):

    def __init__(self):

        super().__init__()

        self.setWindowTitle("Code Highlighter")
        self.setGeometry(100, 100, 800, 600)

        self.initUI()

    def initUI(self):

        layout = QVBoxLayout()

        self.fileListWidget = QListWidget()
        self.fileListWidget.itemClicked.connect(
            self.load_file)

        self.fileContentPlainTextEdit = QPlainTextEdit()
```

ЗАКЛЮЧЕНИЕ

В результате работы была разработана программа для выделения синтаксиса алгоритмических языков программирования при чтении лекций. Программа прошла все тестирования и готова к эксплуатации.

Для решения поставленной задачи были проведены следующие исследования:

1. Проведен аналитический обзор литературы и проанализированы существующие методы решения актуальной задачи. Также проведен обзор существующих аналогов.
2. В рамках работы разработана структура программы и выбраны необходимые компоненты.
3. Спроектирована архитектура программы, разработаны структурные и принципиальные схемы, на основе которых был создан прототип программы.
4. Осуществлена разработка алгоритмов для решения поставленных задач.
5. Реализована программная часть и проведено тестирование программы. В результате тестирования выяснилось, что все поставленные задачи были выполнены. Программа работает корректно и готова к использованию.

Спасибо за внимание!