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»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____ 2024 г.

Программа для выделения синтаксиса алгоритмических языков при чтении лекций

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУРГУ-090301.2024.405 ПЗ ВКР

Руководитель работы,
к.т.н., доцент каф. ЭВМ
_____В.А. Парасич
«___»_____2024 г.

Автор работы,
студент группы КЭ-405
_____Я.И. Усков
«___»_____2024 г.

Нормоконтролёр,
ст. преподаватель кафедры ЭВМ
_____ С.В. Сяськов
«___»_____2024 г.

Челябинск-2024

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»

УТВЕРЖДАЮ
Заведующий кафедрой ЭВМ
_____ Д.В. Топольский
«___»_____2024 г.

ЗАДАНИЕ

на выпускную квалификационную работу бакалавра

студенту группы КЭ-405
Ускову Ярославу Игоревичу
обучающемуся по направлению
09.03.01 «Информатика и вычислительная техника»

1. **Тема работы: «**Программа для выделения синтаксиса алгоритмических языков при чтении лекций» утверждена приказом по университету от «22» апреля 2024 г. №764-13/12

2. **Срок сдачи студентом законченной работы:** 1 июня 2024 г.

3. **Исходные данные к работе:**
   1) используемые средства разработки: Python, HTML.
4. **Перечень подлежащих разработке вопросов:**
   1) аналитический обзор научно-технической и методической литературы по тематике работы, поиск аналогичных решений;
   2) определение требований к разрабатываемой программе;
   3) реализация программы;

4) тестирование программы.

5. **Дата выдачи задания**: 1 декабря 2023 г.

Руководитель работы _____/*В.А. Парасич*/

Студент _____/*Я.И. Усков*/

КАЛЕНДАРНЫЙ ПЛАН

Этап	Срок сдачи	Подпись руководителя
Введение и обзор литературы	10.03.2024	
Определение требований, предъявляемых к проекту. Проектирование технической и программной архитектур системы	21.03.2024	
Реализация программы	04.04.2024	
Составление программы тестовых испытаний, с последующим их проведением	5.05.2024	
Компоновка текста работы и сдача на нормоконтроль	23.05.2024	
Подготовка презентации и доклада	30.05.2024	

Руководитель работы _____/*В.А. Парасич*/

Студент _____/*Я.И. Усков*/

АННОТАЦИЯ

Усков Я.И. Программа для выделения синтаксиса алгоритмических языков при чтении лекций – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ВШ ЭКН; 2024, 37 с., библиогр. список – 15 наим.

В данной выпускной квалификационной работе проведен анализ современной научно-технической и методической литературы по подсветке синтаксиса алгоритмических языков программирования. Рассмотрены существующие решения, выявлены их преимущества и недостатки.

В ходе практической работы определены критерии для программ подсветки синтаксиса, собраны данные по техническим компонентам. На основе анализа разработана архитектура программы, создано программное обеспечение, реализующее подсветку синтаксиса для множества языков программирования.

Проведено итоговое тестирование, подтвердившее соответствие программы требуемым характеристикам, включая корректность подсветки синтаксиса, стабильность работы и удобство использования.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В современном мире информационные технологии играют ключевую роль в различных областях науки, техники и повседневной жизни. Одним из важнейших аспектов подготовки специалистов в области информационных технологий является обучение программированию. Преподавание алгоритмических языков программирования требует использования различных инструментов и методов, которые позволяют эффективно передавать знания и навыки студентам.

Одним из таких инструментов является подсветка синтаксиса — функция, позволяющая выделять различные элементы программного кода различными цветами и стилями. Подсветка синтаксиса облегчает восприятие и понимание кода, помогает быстрее находить ошибки и улучшает общее качество написанного кода. Она особенно полезна при чтении лекций и демонстрации примеров программирования, так как позволяет преподавателю наглядно показывать синтаксические структуры и ключевые элементы языка программирования.

Целью данной выпускной квалификационной работы является разработка программы для выделения синтаксиса алгоритмических языков программирования при чтении лекций. Программа должна обеспечивать поддержку множества языков программирования, иметь интуитивно понятный интерфейс и быть удобной в использовании для преподавателей и студентов.

В ходе работы были рассмотрены существующие решения для подсветки синтаксиса, проведен их анализ и выделены основные преимущества и недостатки. На основе этого анализа были сформулированы функциональные требования к разрабатываемой программе. В результате разработки было создано приложение, которое позволит преподавателям эффективно использовать его при проведении лекций и демонстрации примеров кода.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметная область данной выпускной квалификационной работы включает в себя изучение и анализ технологий и методов, используемых для подсветки синтаксиса в текстовых редакторах и средах разработки. Подсветка синтаксиса играет ключевую роль в обучении программированию, так как позволяет улучшить читаемость и понимание кода, облегчает поиск ошибок и способствует более эффективному усвоению учебного материала.

Подсветка синтаксиса представляет собой важную функцию в текстовых редакторах и интегрированных средах разработки (IDE), которая обеспечивает визуальное выделение различных элементов кода. Это значительно упрощает восприятие программного кода, делает его более читабельным и структурированным. Основными элементами кода, подлежащими подсветке являются:

− ключевые слова языков программирования;

− идентификаторы (имена переменных, функций, классов и т.д.);

− литералы (числа, строки);

− операторы.

Технологии и методы подсветки синтаксиса

Для реализации подсветки синтаксиса используются различные технологии и методы, среди которых можно выделить следующие:

− регулярные выражения (использование регулярных выражений для распознавания и выделения ключевых элементов кода);

− парсеры и анализаторы (создание парсеров, которые анализируют структуру кода и выделяют различные элементы);

− готовые библиотеки и фреймворки (использование существующих библиотек и фреймворков для подсветки синтаксиса).

1.1. Обзор аналогов

На данный момент существует множество приложений, которые позволяют решать задачи, связанные с подсветкой синтаксиса и использованием текстовых редакторов для программирования. Однако, ни одно из них не решает все задачи одновременно, что делает актуальной разработку нового приложения. Рассмотрим наиболее подходящие к заданным требованиям аналоги:

− Sublime Text;

− Visual Studio Code;

− PyCharm;

− Notepad++;

− Atom.

Sublime Text — это популярный текстовый редактор, известный своей высокой скоростью работы и удобством использования. Он поддерживает подсветку синтаксиса для множества языков программирования и обладает широкими возможностями для настройки и расширения функциональности с помощью плагинов.

Преимущества:

− быстрая и легковесная программа;

− поддержка множества плагинов, включая подсветку синтаксиса для большинства языков программирования.

Недостатки:

− ограниченные возможности настройки без установки дополнительных плагинов;

− платная версия для полноценного использования.

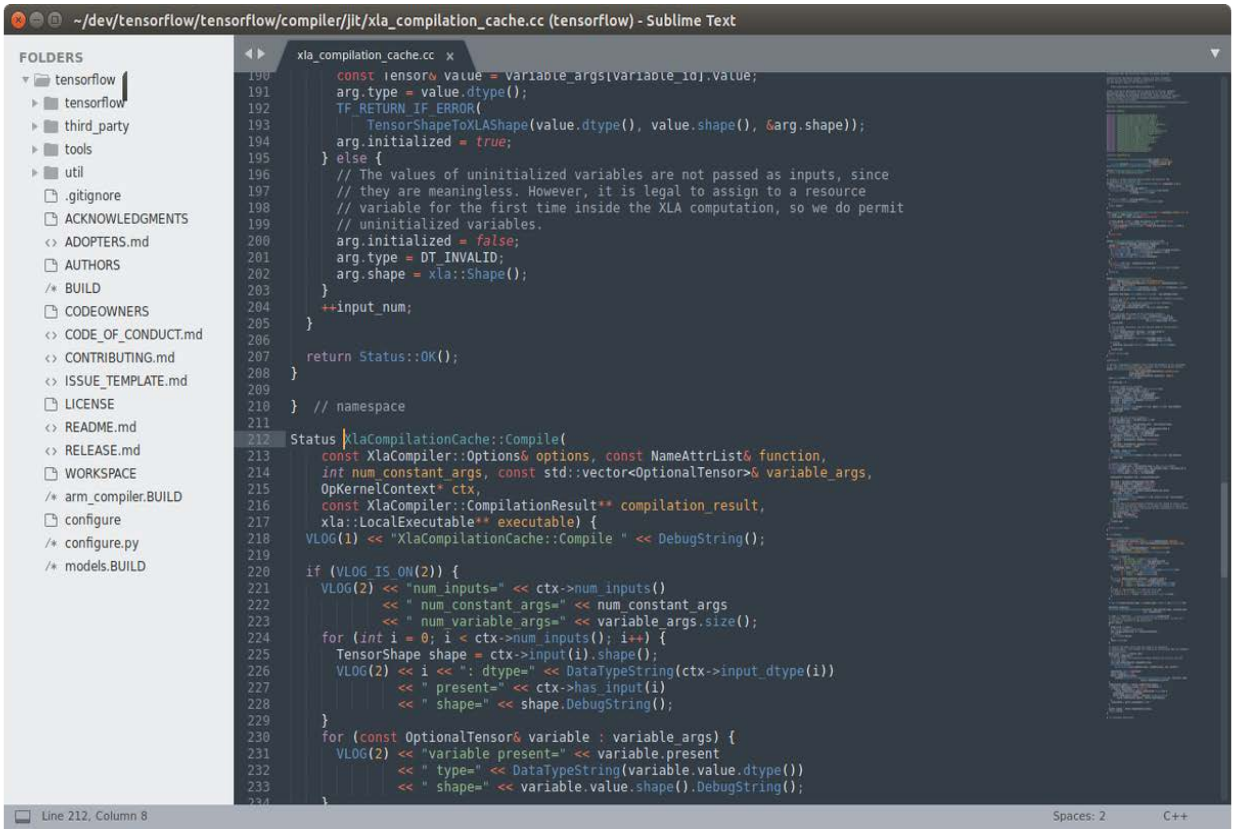

Рисунок 1 – Sublime Text

Visual Studio Code — бесплатный редактор кода от Microsoft, который поддерживает множество языков программирования и обладает широкими возможностями для расширения функциональности через плагины.

Преимущества:

– удобный интерфейс;

– широкая поддержка языков программирования;

– мощная система подсветки синтаксиса

Недостатки:

– высокие системные требования;

– занимает больше ресурсов по сравнению с другими текстовыми редакторами.

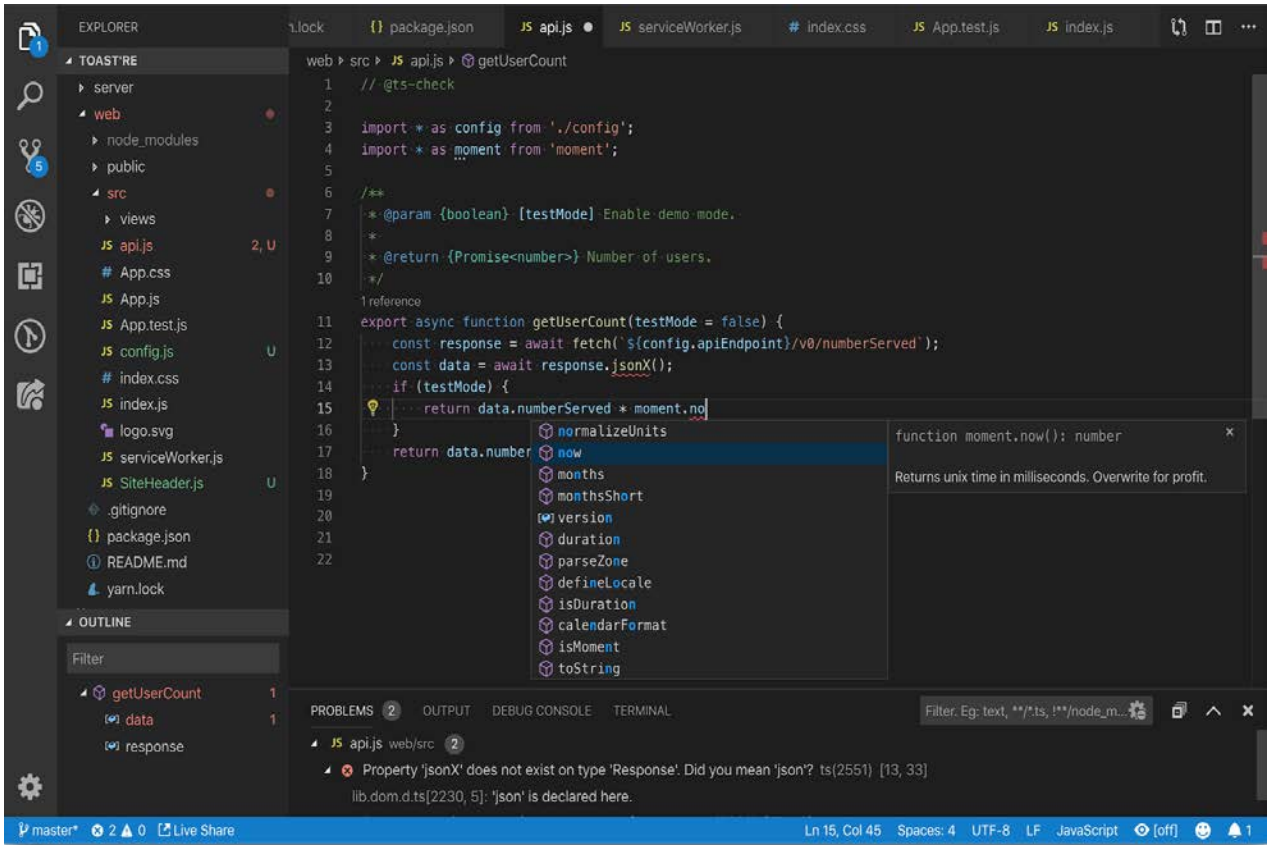

Рисунок 2 – Visual Studio Code

PyCharm — мощная среда разработки от JetBrains, предназначенная для разработки на языке Python. Она поддерживает подсветку синтаксиса, автодополнение кода, отладку и другие функции, необходимые для эффективной разработки.

Преимущества:

- поддержка подсветки синтаксиса и множества других функций для повышения продуктивности.

Недостатки:

- платная версия с расширенными возможностями;

- высокая потребность в системных ресурсах.

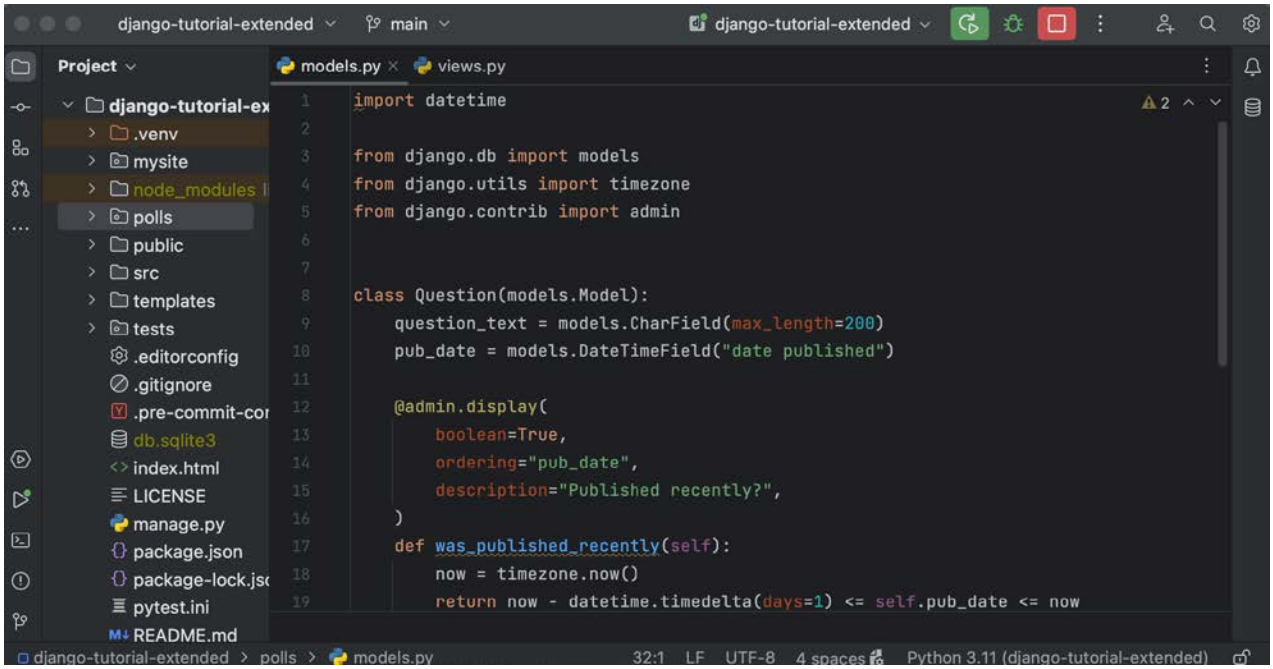

Рисунок 3 – PyCharm

Notepad++ — это легкий и быстрый текстовый редактор для Windows, который поддерживает подсветку синтаксиса для множества языков программирования. Он широко используется программистами благодаря своей простоте и эффективности.

Преимущества:

- быстрота работы;
- возможность настройки подсветки синтаксиса;
- поддержка множества языков программирования;
- бесплатное использование.

Недостатки:

- ограниченные возможности по сравнению с более мощными редакторами;
- только для Windows.

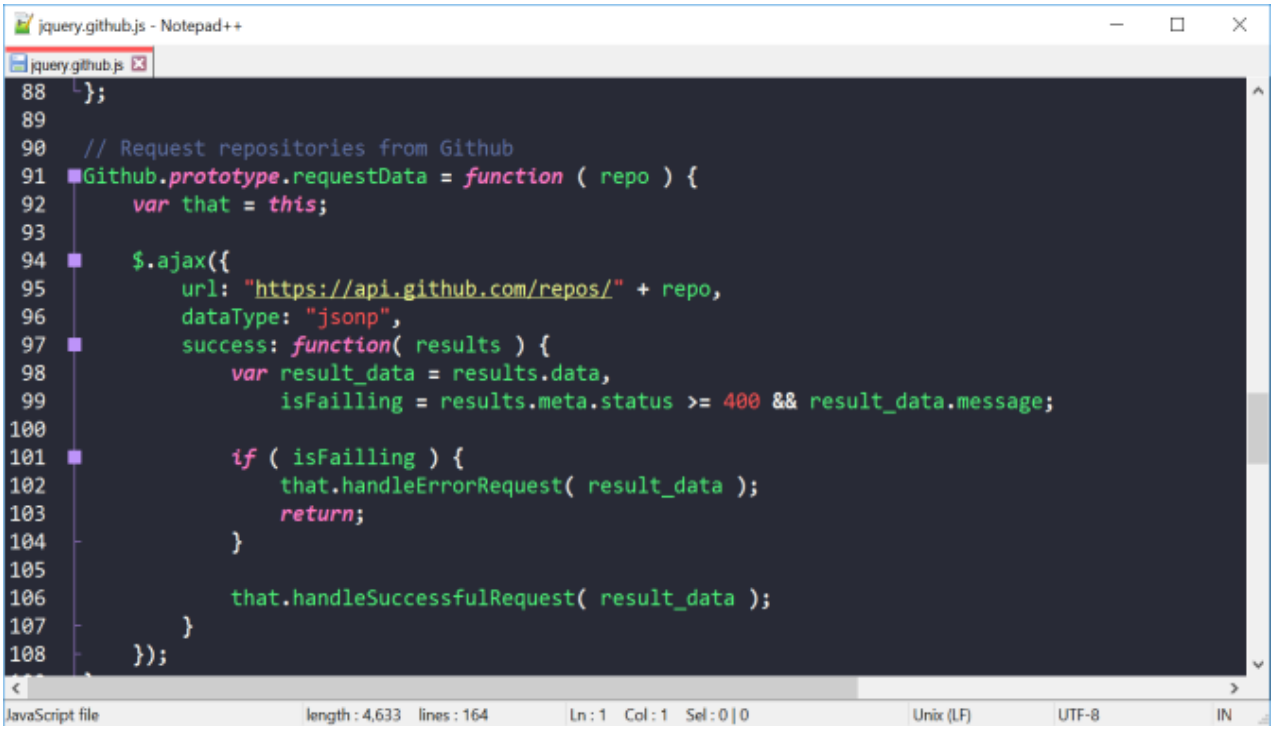

Рисунок 4 – Notepad++

Atom — бесплатный и открытый текстовый редактор, разработанный GitHub. Он поддерживает множество языков программирования и обладает возможностью расширения функциональности через плагины.

Преимущества:

- бесплатное использование;
- поддержка множества языков программирования;
- возможность использования плагинов;
- возможность настройки и расширения функциональности.

Недостатки:

- проблемы с производительностью при работе с большими файлами;
- использование значительных системных ресурсов.

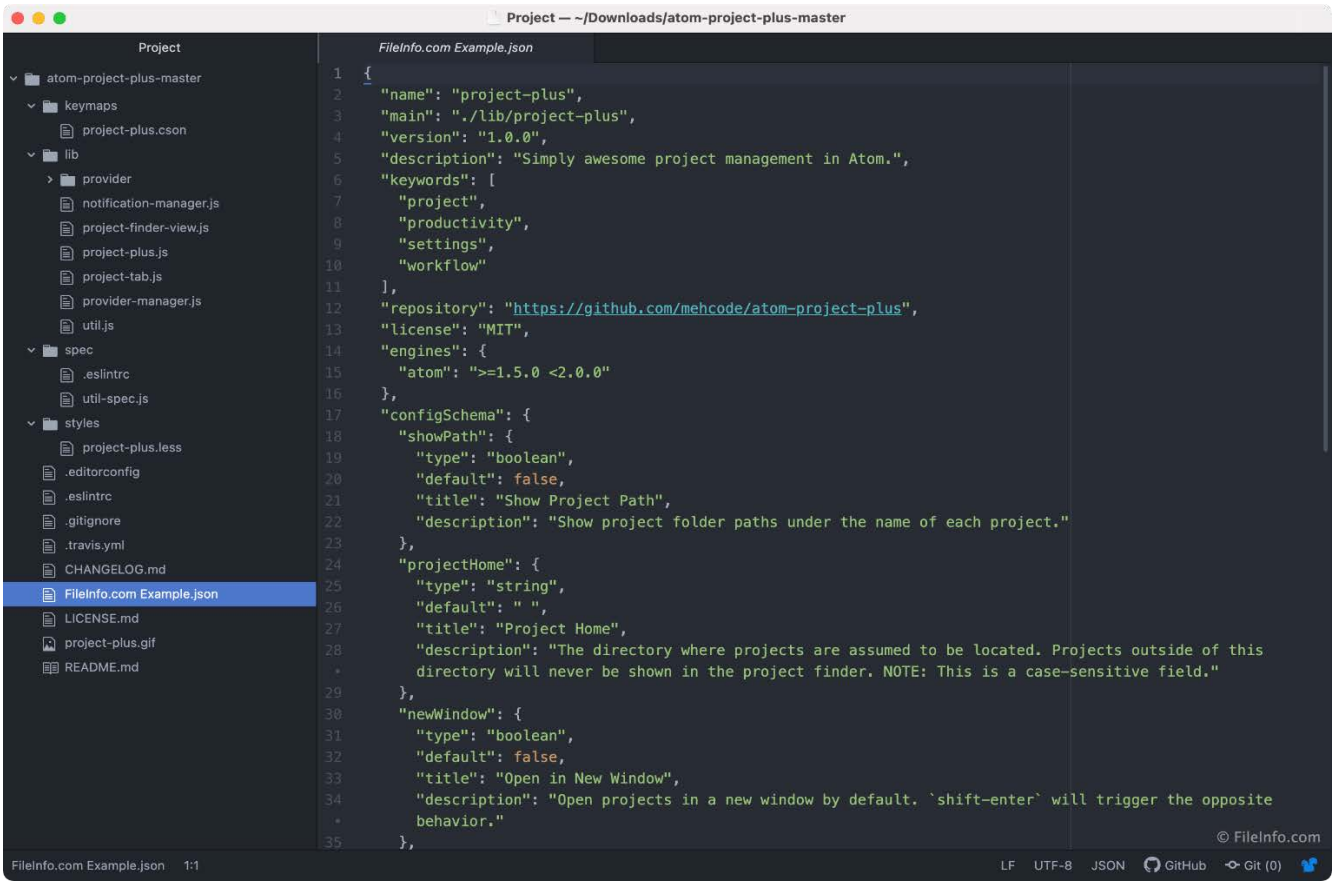

Рисунок 5 – Atom

Анализ существующих решений показывает, что каждое из них имеет свои сильные и слабые стороны. Тем не менее, ни одно из этих решений не удовлетворяет всем требованиям, предъявляемым к программе, предназначенной для подсветки синтаксиса алгоритмических языков программирования в образовательной среде. Это подчеркивает необходимость разработки нового приложения, которое будет учитывать специфику использования в процессе преподавания и обучения.

1.2. Сравнительный анализ аналогов

В таблице 1 представлен сравнительный анализ аналогов разрабатываемого приложения по некоторым критериям.

Таблица 1 – Сравнительный анализ аналогов разрабатываемого приложения.

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	+	-	+	-	-

Исходя из данных, приведенных в таблице, можно прийти к выводу, что существующие текстовые редакторы и среды разработки обладают широкими возможностями и поддерживают множество языков программирования. Однако, они не удовлетворяют специфическим требованиям образовательных задач, таким как удобство использования при чтении лекций и демонстрации примеров кода.

1.3. Актуальность проблемы

В современных условиях развития информационных технологий и программирования повышается потребность в качественном обучении студентов алгоритмическим языкам программирования. Одним из ключевых аспектов такого обучения является возможность эффективного визуального восприятия синтаксических конструкций различных языков программирования. Это делает актуальной разработку

программных инструментов, способных улучшить восприятие и понимание учебного материала.

Целью данной выпускной квалификационной работы является разработка программы для выделения синтаксиса алгоритмических языков программирования во время чтения лекций. Подобная программа будет способствовать повышению качества преподавания и усвоения материала студентами за счет визуализации ключевых элементов кода.

Актуальность разработки данной программы обусловлена следующими факторами:

1. Повышение качества обучения: подсветка синтаксиса способствует лучшему пониманию структуры и логики программного кода, облегчает процесс обучения, особенно для начинающих программистов.
2. Ускорение процесса обучения: визуальное выделение элементов кода помогает быстрее находить ошибки и улучшает восприятие материала, что особенно важно при чтении лекций и демонстрации примеров программирования.
3. Интеграция с образовательными процессами: программа может быть интегрирована в существующие учебные планы и использоваться как на лекциях, так и в процессе самостоятельной работы студентов.
4. Поддержка множества языков программирования: в условиях быстрого развития информационных технологий важно обеспечить поддержку различных языков программирования, что делает программу универсальным инструментом для преподавателей и студентов.

Разрабатываемая программа будет отличаться от существующих аналогов следующими особенностями:

1. Интуитивно понятный интерфейс: программа будет разработана с учетом потребностей преподавателей и студентов, обеспечивая простой и удобный интерфейс.

2. Высокая производительность: программа будет оптимизирована для работы на устройствах с различными техническими характеристиками, обеспечивая стабильную и быструю работу.

Таким образом, разработка программы для выделения синтаксиса алгоритмических языков программирования является актуальной задачей, направленной на повышение эффективности образовательного процесса в области информационных технологий и программирования.

Вывод по первой главе

В первой главе были рассмотрены основные аспекты подсветки синтаксиса, её значение и принципы работы. Был проведен анализ существующих решений, что позволило выявить их сильные и слабые стороны. На основе этого анализа была обоснована необходимость разработки нового программного обеспечения, способного удовлетворить специфические требования образовательного процесса. Таким образом, проведенное исследование и обзор литературы подтвердили актуальность и значимость разработки программы для подсветки синтаксиса, что является важным шагом на пути к улучшению процесса обучения программированию.

2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОЕКТИРУЕМОЙ ПРОГРАММЕ

2.1. Функциональные требования

Разрабатываемая программа для подсветки синтаксиса должна удовлетворять следующим функциональным требованиям:

1. Поддержка множества языков программирования:

программа должна поддерживать подсветку синтаксиса для различных языков программирования, таких как Python [1], C++ [2], SQL [3], и другие.

2. Автоматическое определение языка:

программа должна автоматически определять язык программирования на основе содержимого кода и применять соответствующую подсветку синтаксиса.

3. Интуитивно понятный интерфейс:

интерфейс программы должен быть простым и удобным для использования как преподавателями, так и студентами. Все основные функции должны быть доступны через графический интерфейс.

4. Высокая производительность:

программа должна обеспечивать быструю и стабильную работу даже при обработке больших файлов с исходным кодом.

5. Поддержка масштабирования текста:

программа должна позволять изменять размер текста.

2.2. Нефункциональные требования

К нефункциональным требованиям разрабатываемой программы для подсветки синтаксиса можно отнести следующие:

1. Надежность: программа должна быть надежной и минимизировать вероятность сбоев и ошибок во время работы.

2. Производительность: программа должна обеспечивать высокую производительность, поддерживая быстрое выполнение всех операций, связанных с подсветкой синтаксиса, даже при работе с большими файлами.

3. Удобство в использовании: программа должна быть интуитивно понятной и удобной в использовании, чтобы пользователи могли легко освоить ее функции без необходимости в долгом обучении.

Диаграмма вариантов использования

Для проектирования системы использовался стандартизированный язык моделирования UML (англ. "Unified Modeling Language"). В соответствии с требованиями была создана диаграмма вариантов использования, которая демонстрирует функциональные возможности разрабатываемой программной системы, доступные пользователям (рисунок 6). В нашем случае диаграмма отображает взаимодействие актера «Пользователь» с системой.,

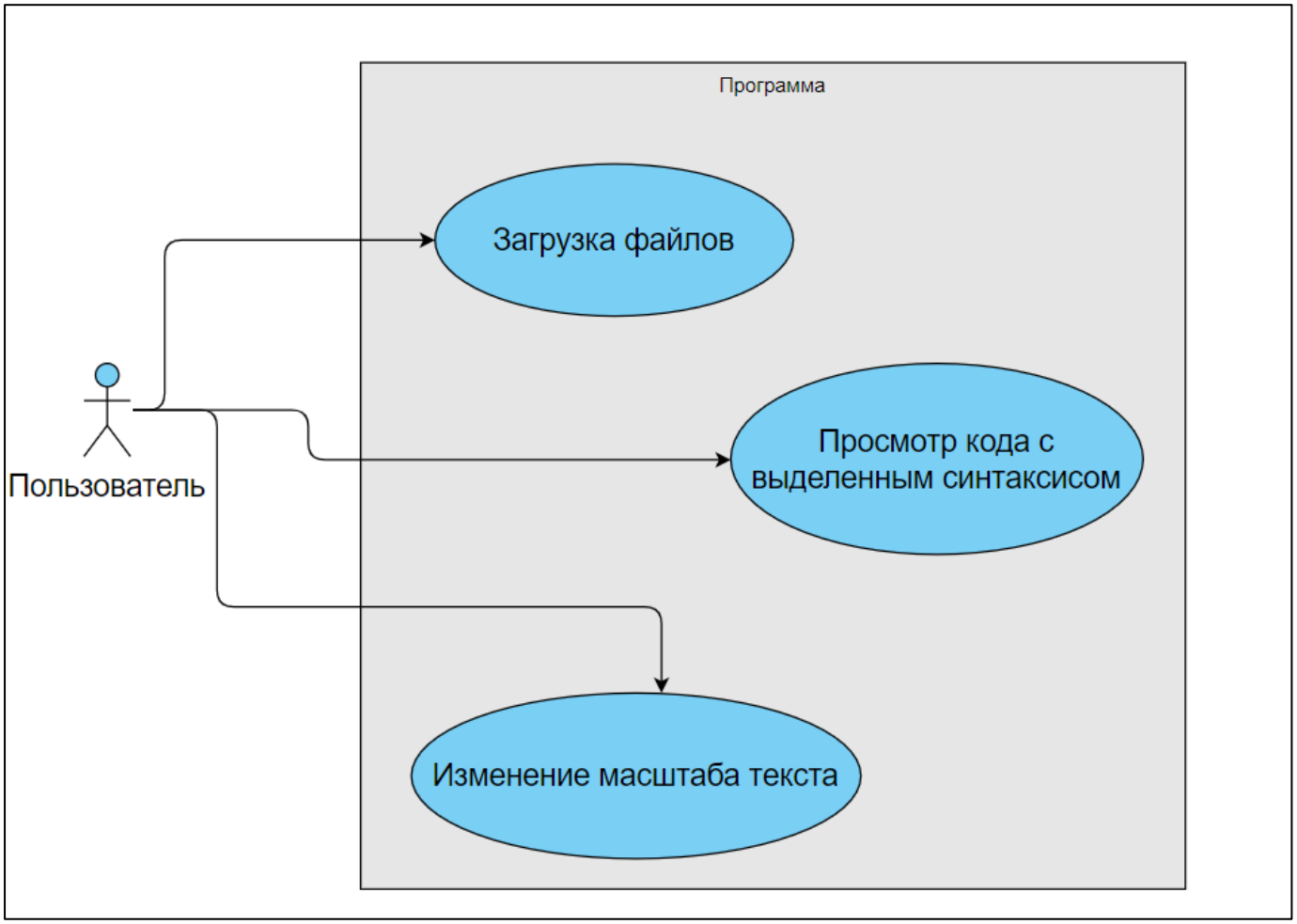

Рисунок 6 – Диаграмма вариантов использования

С системой взаимодействует только один актер – «Пользователь», использующий программу. Ему доступны перечисленные ниже варианты использования.

1. Вариант использования «Загрузка файлов». Пользователь может загрузить файлы для последующей обработки и отображения.
2. Вариант использования «Просмотр кода с выделенным синтаксисом». Пользователь может просмотреть код из загруженного файла с выделенным синтаксисом
3. Вариант использования «Изменение масштаба текста». Пользователь может изменить масштаб просматриваемого файла с выделенным синтаксисом.

Спецификация вариантов использования приведена в таблицах 2-4.

Таб 2 – Спецификация вариантов использования «Загрузка файлов»

Прецедент: Загрузка файлов
ID: 1
Краткое описание: Пользователь загружает файлы в программу для последующей обработки и отображения
Главные актеры: Пользователь
Второстепенные актеры: отсутствуют
Предусловия: программа должна быть запущена
Основной поток: 1. Пользователь нажимает кнопку "Load Files". 2. Программа открывает диалоговое окно для выбора файлов. 3. Пользователь выбирает один или несколько файлов и подтверждает выбор. 4. Программа добавляет выбранные файлы в список для отображения.
Постусловия: выбранные файлы отображены в списке файлов программы.
Исключения: если файл не может быть загружен, программа показывает сообщение об ошибке.

Таблица 3 – Спецификация вариантов использования «Просмотр кода с выделенным синтаксисом»

Прецедент: Просмотр кода с выделенным синтаксисом
ID: 2
Краткое описание: Пользователь просматривает код с подсветкой синтаксиса после загрузки файлов.
Главные актеры: Пользователь
Второстепенные актеры: отсутствуют
Предусловия: файл должен быть загружен и отображен в списке файлов.
Основной поток: 1. Пользователь выбирает файл из списка. 2. Программа загружает содержимое файла. 3. Программа применяет подсветку синтаксиса к содержимому файла. 4. Программа отображает код с подсветкой синтаксиса в текстовом браузере.
Постусловия: Пользователь видит содержимое файла с примененной подсветкой синтаксиса.
Исключения: если файл не может быть открыт или подсветка синтаксиса не может быть применена, программа показывает сообщение об ошибке.

Таблица 4 – Спецификация вариантов использования «Изменение масштаба текста»

Прецедент: Изменение масштаба текста
ID: 3
Краткое описание: Пользователь изменяет размер текста для лучшего восприятия кода.
Главные актеры: Пользователь
Второстепенные актеры: отсутствуют
Предусловия: файл должен быть загружен и отображен в текстовом браузере.
1. Основной поток: Пользователь использует элемент управления для изменения размера текста. 2. Программа изменяет размер шрифта в текстовом браузере. 3. Текст с подсветкой синтаксиса отображается с новым размером шрифта.
Постусловия: Пользователь видит текст с новым размером шрифта.
Исключения: Отсутствуют

Вывод по второй главе

Во второй главе были определены функциональные и нефункциональные требования к системе. На основе этих требований была построена диаграмма вариантов использования, определены основные актеры, взаимодействующие с системой, а также приведены краткое описание и спецификация вариантов использования.

3. РЕАЛИЗАЦИЯ

В данной главе представлена реализация программы для выделения синтаксиса алгоритмических языков во время чтения лекций.

3.1 Программные средства реализации

Для разработки программы подсветки синтаксиса кода были выбраны следующие программные средства и технологии:

1. Язык программирования Python

Python был выбран в качестве основного языка программирования для разработки данной программы. Основными причинами выбора Python являются:

1) Простота и читаемость кода: Python известен своей лаконичностью и читаемостью, что делает его идеальным для разработки приложений, требующих быстрой итерации и разработки.

2) Богатый набор библиотек: Python имеет широкий набор стандартных библиотек и сторонних пакетов, которые значительно упрощают разработку сложных приложений.

3) Сообщество и поддержка: Python обладает большим и активным сообществом разработчиков, что обеспечивает наличие множества ресурсов и примеров для обучения и решения возникающих проблем.

2. Библиотека PyQt6[4]

PyQt6 — это набор привязок Python к фреймворку Qt6, который используется для создания графического интерфейса пользователя (GUI). Основные преимущества использования PyQt6:

1) Мощный и гибкий GUI-фреймворк: Qt6 предоставляет широкий набор виджетов и инструментов для создания современных и функциональных пользовательских интерфейсов.

2) Поддержка множества виджетов: PyQt6 позволяет создавать сложные интерфейсы с использованием стандартных и настраиваемых виджетов.

3) Интеграция с Python: PyQt6 позволяет легко интегрировать элементы GUI с Python-кодом, обеспечивая простоту разработки и расширяемость приложений.

3. Библиотека Pygments[5]

Pygments — это библиотека для подсветки синтаксиса, поддерживающая множество языков программирования. Преимущества использования Pygments:

1) Широкая поддержка языков: Pygments поддерживает подсветку синтаксиса для более чем 300 языков и форматов файлов.

2) Гибкость и настраиваемость: библиотека позволяет легко настраивать стили подсветки и интегрировать их в различные приложения.

3) Легкость использования: Pygments предоставляет простые в использовании API (англ. "Application Programming Interface") для определения языка программирования и применения подсветки синтаксиса.

Использование перечисленных программных средств и технологий позволило эффективно реализовать программу для подсветки синтаксиса кода, обеспечив её функциональность и удобство использования.

3.2. Архитектура программы

Архитектура программы для подсветки синтаксиса включает несколько основных компонентов, которые обеспечивают её функциональность и удобство использования. Рассмотрим основные модули и их назначение.

Главный класс приложения (CodeHighlighterApp):

Этот класс наследует QMainWindow и отвечает за создание основного окна приложения, его настройку и управление основными элементами интерфейса.

Функция подсветки кода (highlight_code):

Функция, использующая библиотеку Pygments для определения языка программирования и подсветки синтаксических элементов. Она также отвечает за формирование HTML-кода[6] с подсветкой и стиль CSS[7] для отображения в текстовом браузере.

Инициализация интерфейса (initUI):

Метод, который создает и настраивает элементы интерфейса пользователя, такие как списки файлов, области для отображения кода и кнопки загрузки файлов.

Загрузка файлов (load_files):

Метод, вызываемый при нажатии кнопки "Load Files". Он открывает диалоговое окно для выбора файлов и добавляет выбранные файлы в список для последующего отображения.

Загрузка содержимого файла (load_file):

Метод, который загружает содержимое выбранного файла, применяет подсветку синтаксиса и отображает результат в текстовом браузере.

На рисунке 7 представлена диаграмма классов для данной программы

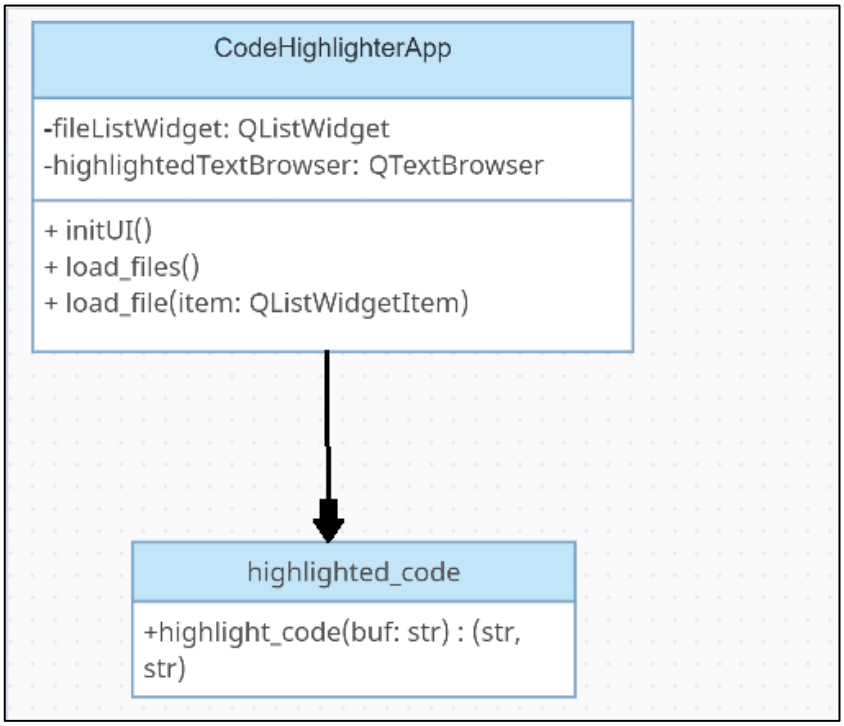

Рисунок 7 – Диаграмма классов

Взаимодействие компонентов

1. Инициализация интерфейса (initUI)

Компонент CodeHighlighterApp инициализирует пользовательский интерфейс, создавая и настраивая виджеты, такие как QListWidget для отображения списка файлов и QTextBrowser для отображения кода с подсветкой синтаксиса.

2. Загрузка файлов (load_files)

Когда пользователь нажимает кнопку "Load Files", компонент CodeHighlighterApp вызывает метод load_files, который открывает диалоговое окно для выбора файлов.
Выбранные файлы добавляются в QListWidget для последующего отображения.

3. Загрузка содержимого файла (load_file)

Когда пользователь выбирает файл из списка, компонент CodeHighlighterApp вызывает метод load_file, который загружает содержимое выбранного файла.
Загруженный контент передается функции highlight_code для применения выделения синтаксиса.

4. Выделение синтаксиса (highlight_code).

Функция highlight_code использует библиотеку Pygments для определения языка программирования загруженного файла и применения соответствующих стилей подсветки синтаксиса.
Возвращаемый HTML-код с подсветкой и стилями CSS передается обратно в CodeHighlighterApp для отображения в QTextBrowser.

5. Отображение кода (QTextBrowser)

Компонент QTextBrowser в CodeHighlighterApp отображает код с подсветкой синтаксиса, полученный от функции highlight_code.

3.3. Реализация программы

Главное окно приложения создается с использованием класса `QMainWindow`. Ниже приведен фрагмент кода для инициализации главного окна и настройки его основных элементов.

Листинг 1 – Реализация класса QmainWindow

```
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()  # Создаем вертикальный макет для размещения виджетов
        self.fileListWidget = QListWidget()  # Создаем виджет списка файлов
        self.fileListWidget.itemClicked.connect(
            self.load_file)  # Подключаем обработчик для загрузки файла при клике на
элемент списка
        self.highlightedTextBrowser = QTextBrowser()  # Создаем виджет для отображения
подсвеченного HTML-кода
        self.highlightedTextBrowser.setFont(QFont("Courier", 10))    # Устанавливаем
моноширинный шрифт
        fileContentSplitter = QSplitter(
            Qt.Orientation.Horizontal)    # Создаем разделитель для горизонтального
разделения экрана
        fileContentSplitter.addWidget(self.highlightedTextBrowser)  # Добавляем виджет
с подсвеченным HTML-кодом
        mainSplitter = QSplitter(Qt.Orientation.Horizontal)  # Создаем разделитель для
вертикального разделения экрана
        mainSplitter.addWidget(self.fileListWidget)  # Добавляем виджет списка файлов
        mainSplitter.addWidget(fileContentSplitter)
        mainSplitter.setSizes([200, 600])  # Устанавливаем размеры областей разделителя
        buttonLayout = QHBoxLayout()
        loadButton = QPushButton("Load Files")  # Создаем кнопку для загрузки файлов
        loadButton.clicked.connect(self.load_files)    # Подключаем обработчик для
загрузки файлов при нажатии на кнопку
        layout.addLayout(buttonLayout)  # Добавляем макет кнопок в основной макет
        layout.addWidget(mainSplitter) #Добавляем основной разделитель в основной макет
        container = QWidget()  # Создаем контейнер для основного макета
        container.setLayout(layout)  # Устанавливаем основной макет в контейнер
        self.setCentralWidget(container)  # Устанавливаем центральный виджет главного
окна
```

Программа создаёт главное окно приложения для подсветки синтаксиса кода. В методе __init__ устанавливается заголовок окна и его размеры, затем вызывается метод initUI для настройки интерфейса. В initUI создаётся вертикальный компоновщик для основного интерфейса, добавляется виджет QListWidget для

отображения списка загруженных файлов, и связывается событие клика по элементу списка с методом load_file. Также создаётся виджет QTextBrowser для отображения текста с подсветкой синтаксиса, которому устанавливается шрифт. Горизонтальные разделители QSplitter используются для разделения списка файлов и области текста. Кнопка QPushButton для загрузки файлов связывается с методом load_files. Все элементы интерфейса добавляются в основной компоновщик, и контейнер QWidget устанавливается как центральный виджет главного окна.

Загрузка файлов

Для загрузки файлов и отображения их содержимого в текстовом поле используется диалоговое окно выбора файлов и функция для чтения файлов. Ниже приведен фрагмент кода для реализации этой функциональности.

Листинг 2 – Реализация функций загрузки и чтения файлов

```
def load_files(self):
    files, _ = QFileDialog.getOpenFileNames(self, "Open Files", "", "All Files (*)")
    if files:
        for file_path in files:
            file_name = QFileInfo(file_path).fileName()
            self.fileListWidget.addItem(file_name)
def load_file(self, item):
    file_path = item.text()  # Получаем путь к файлу из выбранного элемента списка
        with open(file_path, 'r', encoding='utf-8') as f:  # Открываем файл для чтения
            content = f.read()  # Считываем содержимое файла
```

Функция load_files открывает диалоговое окно для выбора файлов, позволяет пользователю выбрать файлы и добавляет их имена в виджет списка файлов. Функция load_file загружает содержимое выбранного файла из списка, открывает файл для чтения, читает его содержимое и сохраняет его в переменную для дальнейшей обработки и отображения в интерфейсе программы.

Подсветка синтаксиса

Для подсветки синтаксиса кода используется библиотека Pygments. Ниже приведен фрагмент кода, реализующий функцию подсветки синтаксиса.

Листинг 3 – Реализация функции выделения синтаксиса

```
def highlight_code(buf):
    try:
        lexer = guess_lexer(buf)  # Определяем язык программирования для кода
        formatter = HtmlFormatter(style="xcode", noclasses=True)  #    форматтер для
HTML с подсветкой синтаксиса
        highlighted_code = highlight(buf, lexer, formatter)  # Подсвечиваем синтаксис
кода
    except Exception:
        return f'<pre>{buf}</pre>', ""  # В случае ошибки возвращаем код без подсветки
и пустые стили
    highlighted_code = re.sub(r'^<div[^>]*>|</div>$', '', highlighted_code)
    highlighted_code = highlighted_code.replace('\n', '<br>')  # Добавление тегов <br>
для переносов
    return highlighted_code, formatter.get_style_defs()  # Возвращаем HTML с подсветкой
и стили
```

Функция highlight_code принимает текст кода и определяет язык программирования с помощью Pygments. Затем она применяет подсветку синтаксиса и форматирует результат в виде HTML-кода. Если происходит ошибка, функция возвращает исходный текст без изменений. После этого она удаляет ненужные теги и заменяет символы новой строки на
. В итоге функция возвращает HTML-код с выделением синтаксиса и стили CSS, которые могут быть использованы для отображения кода в текстовом браузере.

Интерфейс

Графический интерфейс пользователя приложения разработан с использованием библиотеки PyQt6 и включает несколько основных компонентов:

1) виджет для списка файлов реализуется с помощью QListWidget;
2) виджет для отображения текста с подсветкой синтаксиса реализуется с помощью QTextBrowser;
3) кнопка для загрузки файлов реализуется с помощью QPushButton.

Интерфейс приложения разработан таким образом, чтобы обеспечить простоту и удобство использования. Пользователь может легко загружать файлы, просматривать их содержимое и видеть подсвеченный синтаксис кода.

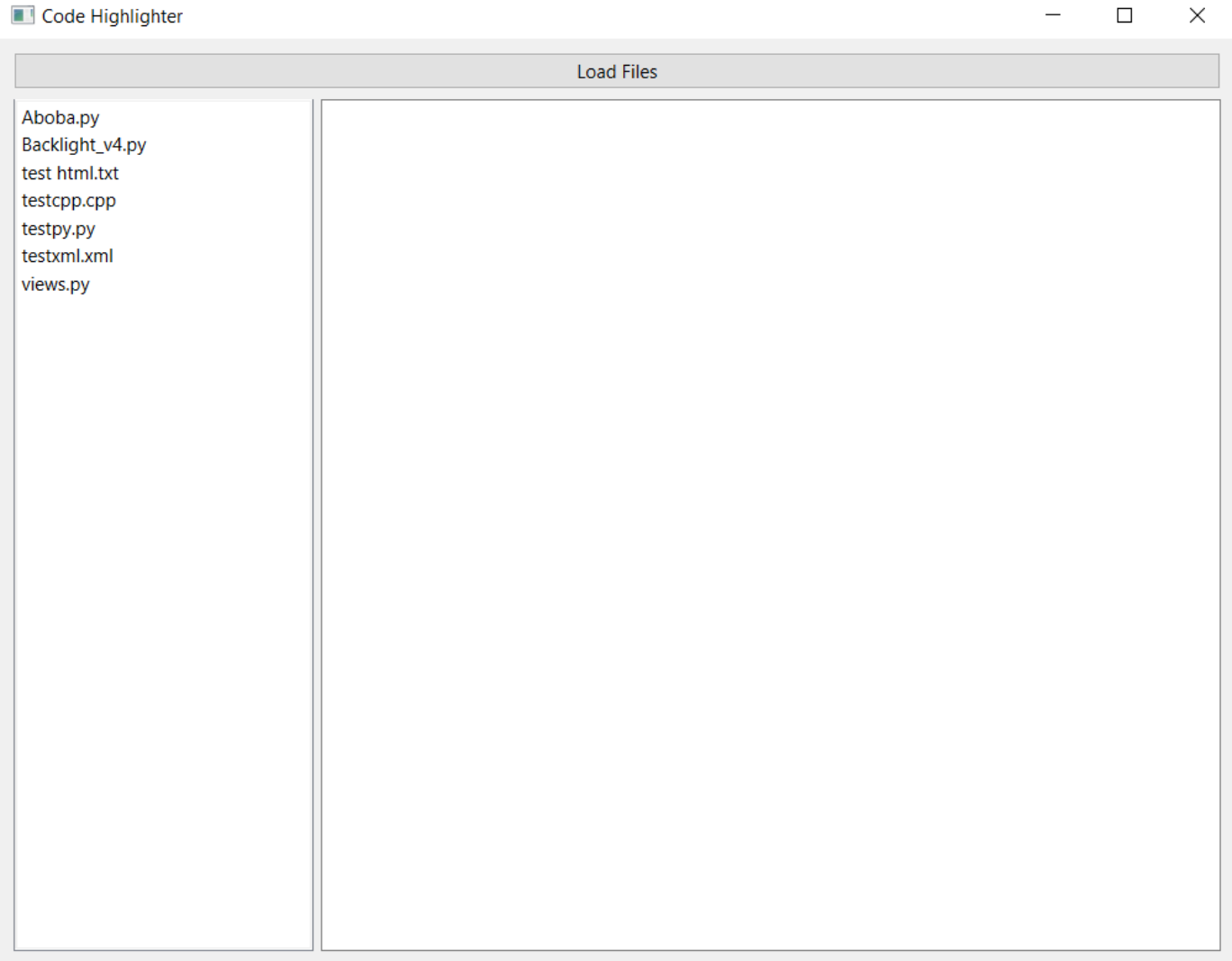

Рисунок 8 – Интерфейс программы

Вывод по разделу три

В данной главе была представлена реализация программы для подсветки синтаксиса алгоритмических языков. Программа была разработана с использованием языка программирования Python и библиотек PyQt6 и Pygments, что обеспечило её функциональность и удобство использования. Были подробно рассмотрены основные компоненты архитектуры системы, их взаимодействие и реализация, включая создание интерфейса, загрузку и чтение файлов, а также применение подсветки синтаксиса. Описанная реализация позволяет создать интуитивно понятный и

функциональный графический интерфейс, который обеспечивает пользователям возможность легко загружать файлы, просматривать их содержимое и видеть подсвеченный синтаксис кода

4. ТЕСТИРОВАНИЕ

В данной главе описываются методы и результаты тестирования программы для подсветки синтаксиса алгоритмических языков. Тестирование проводилось с целью проверки корректности работы всех компонентов системы и обеспечения её надежности и удобства использования.

Тест № 1. Загрузка файлов

Цель тестирования: убедиться, что система корректно загружает текстовые файлы.

Входные данные: пользователь выбирает текстовые файлы форматов .py и .cpp через диалоговое окно.

Ожидаемый результат: система успешно загружает выбранные файлы и отображает их имена в списке загруженных в программу файлов.

Полученный результат: совпал с ожидаемым. Система корректно отобразила имена всех выбранных файлов в списке файлов.

Тест № 2. Подсветка синтаксиса для Python

Цель тестирования: проверить корректность подсветки синтаксиса для языка Python.

Входные данные: файл testpy.py с содержимым на языке Python.

Ожидаемый результат: система распознает язык программирования Python и применяет соответствующую подсветку синтаксиса.

Полученный результат: совпал с ожидаемым. Система корректно определила язык программирования и применила подсветку синтаксиса к содержимому файла.

Результат теста представлен на рисунке 9.

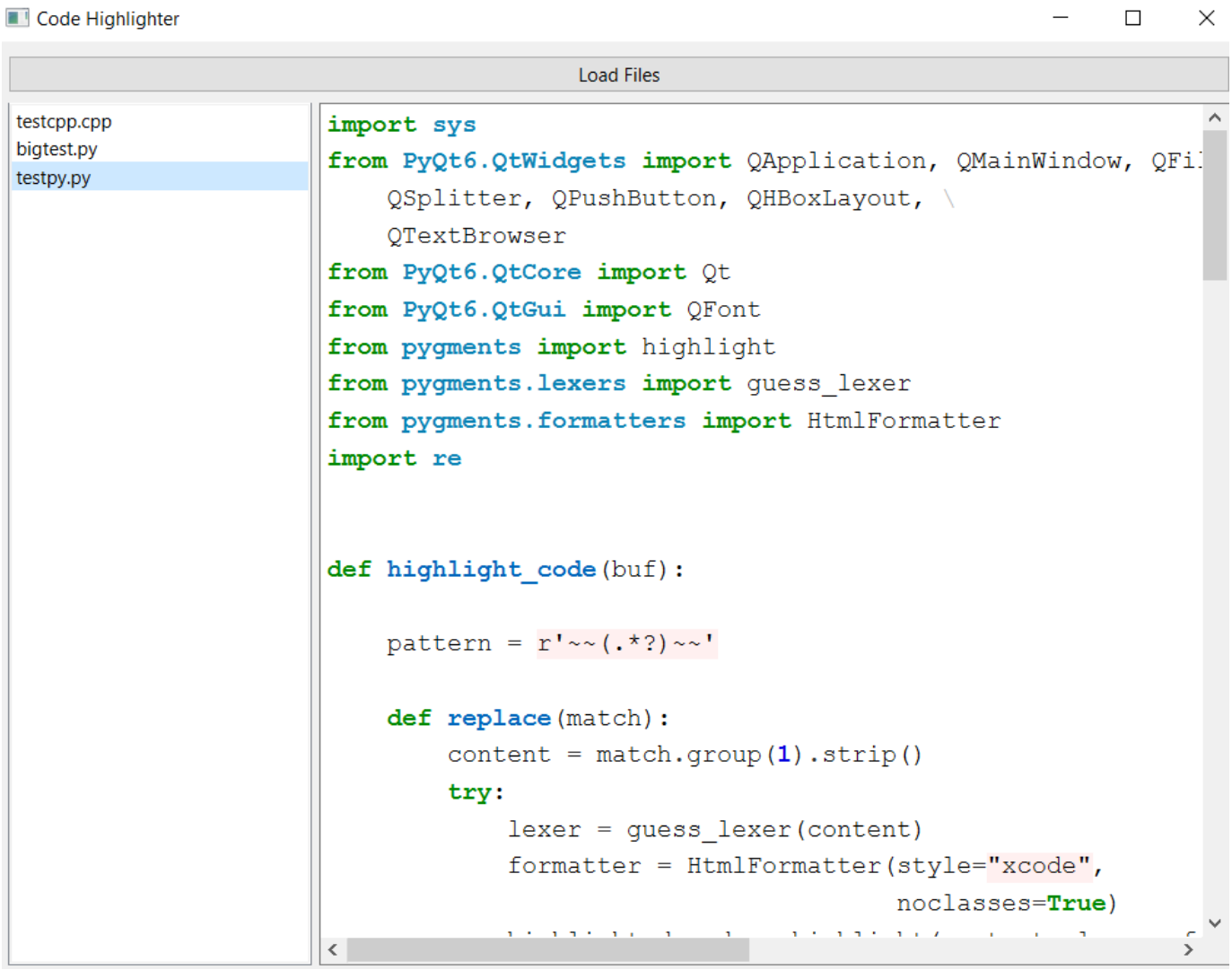

Рисунок 9 – Содержимое файла testpy.py с выделением синтаксиса

Тест № 3. Подсветка синтаксиса для C++

Цель тестирования: проверить корректность подсветки синтаксиса для языка C++.

Входные данные: файл testcpp.cpp с содержимым на языке C++. Ожидаемый результат: система распознает язык программирования C++ и применяет соответствующее выделение синтаксиса. Полученный результат: совпал с ожидаемым. Система корректно определила язык

программирования и применила подсветку синтаксиса к содержимому файла. Результат тестирования представлен на рисунке 10.

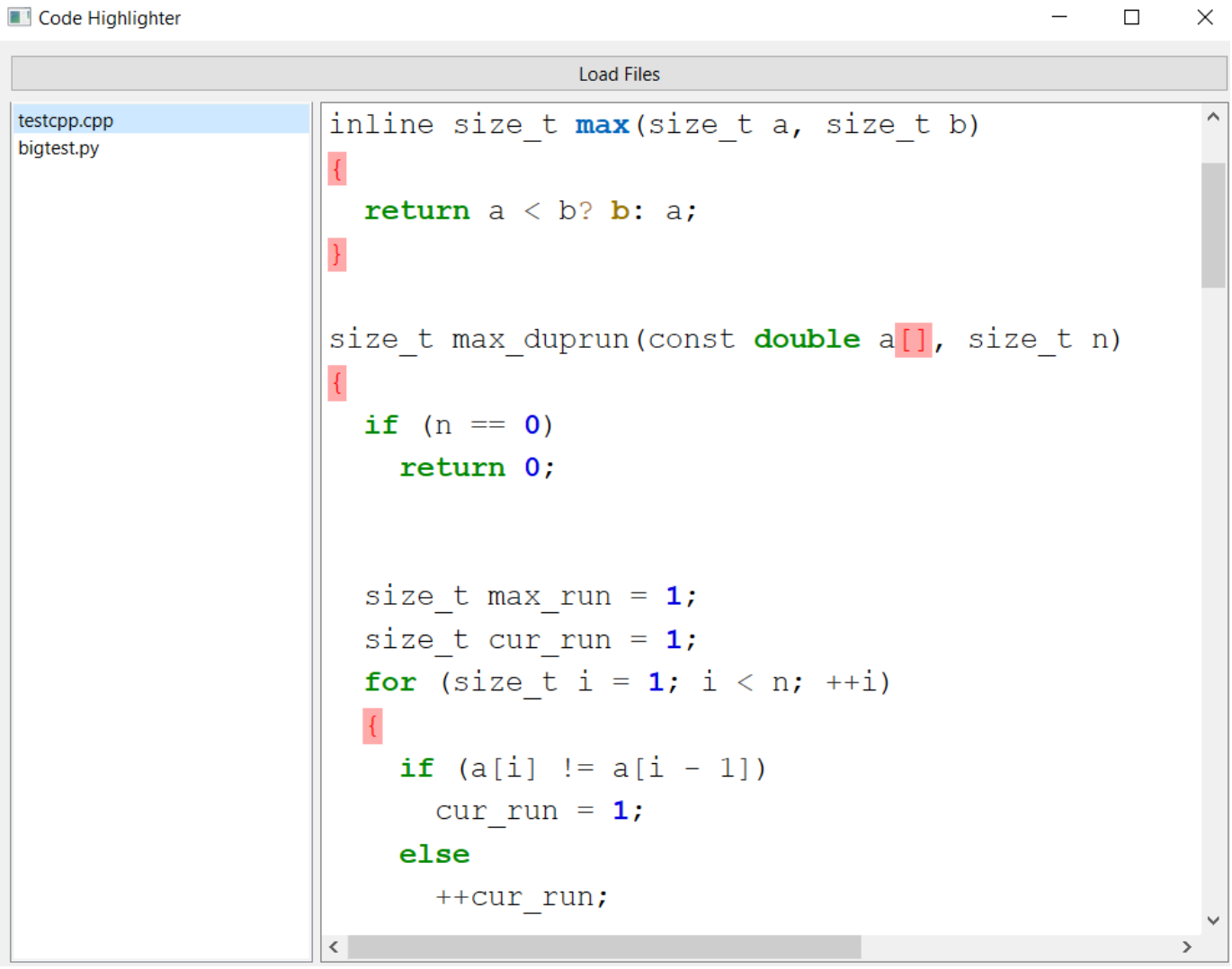

Рисунок 10 – Содержимое файла testcpp.cpp с выделением синтаксиса

Тест № 4. Отображение больших файлов

Цель тестирования: убедиться, что система корректно обрабатывает и отображает большие файлы.
Входные данные: файл большого размера.
Ожидаемый результат: система успешно загружает и отображает содержимое файла, применяя подсветку синтаксиса, без значительных задержек и сбоев.

Полученный результат: совпал с ожидаемым. Система корректно обработала и отобразила содержимое большого файла с применением подсветки синтаксиса.

Тест № 5. Масштабирование текста

Цель тестирования: убедиться, что система корректно масштабирует текст при изменении размера окна.
Входные данные: пользователь изменяет размер окна программы.
Ожидаемый результат: текст с подсветкой синтаксиса корректно масштабируется и отображается в измененном окне.
Полученный результат: совпал с ожидаемым. Текст с подсветкой синтаксиса корректно масштабируется и отображается при изменении размера окна. Результат тестирования представлен на рисунке 11.

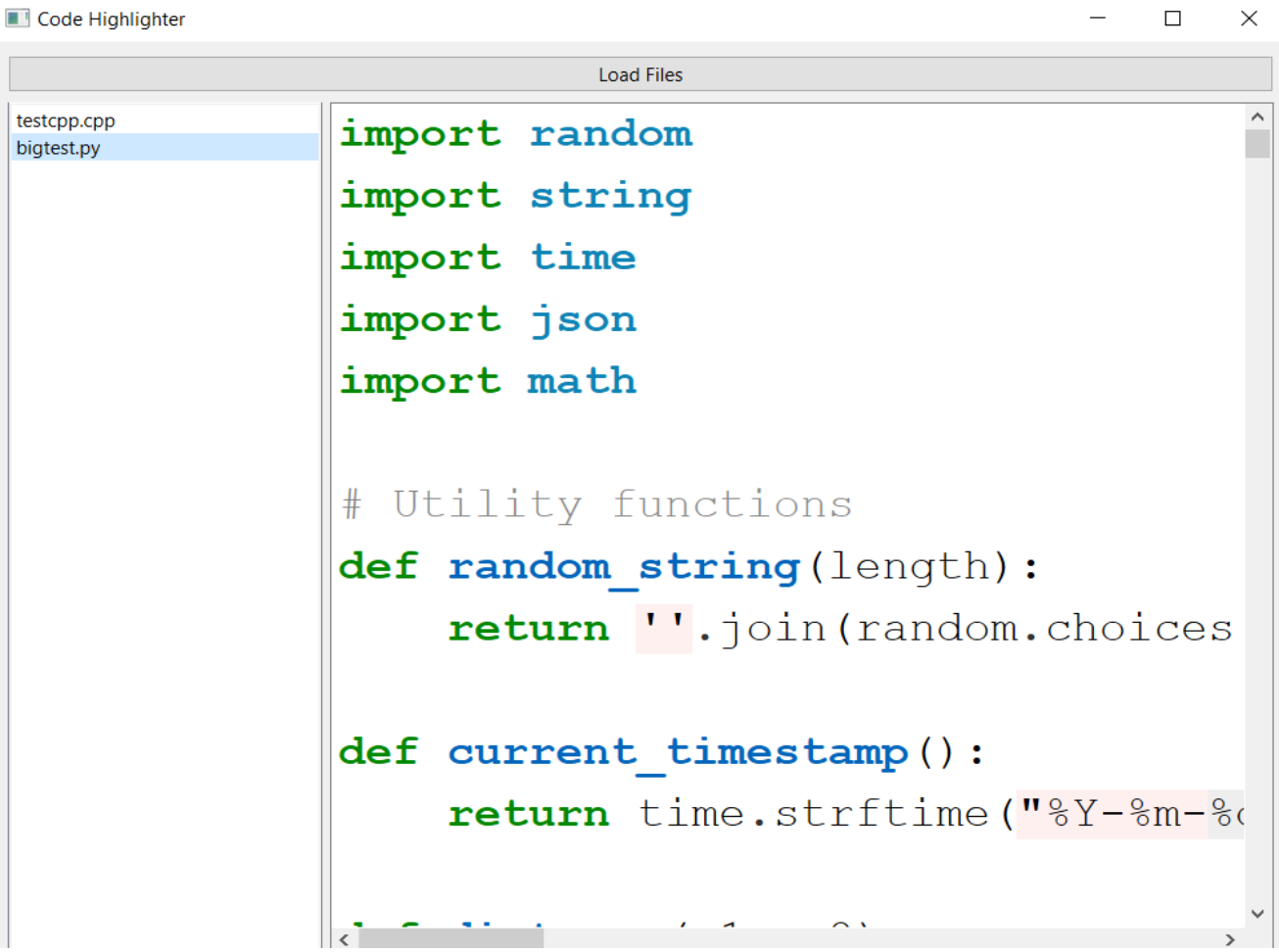

Рисунок 11 – Текст файла bigtest.py с увеличенным масштабом

Вывод по разделу четыре

Проведенные тесты показали, что программа для подсветки синтаксиса алгоритмических языков работает корректно и соответствует заявленным требованиям. Основные результаты тестирования следующие:

1) Загрузка файлов: Система успешно загружает текстовые файлы форматов .py и .cpp, корректно отображая их имена в списке файлов.
2) Подсветка синтаксиса для Python: Программа правильно определяет язык программирования Python и корректно применяет соответствующую подсветку синтаксиса.
3) Подсветка синтаксиса для C++: Программа успешно распознает язык программирования C++ и корректно применяет подсветку синтаксиса к содержимому файла.
4) Отображение больших файлов: Система справляется с обработкой и отображением файлов большого размера, обеспечивая подсветку синтаксиса без значительных задержек и сбоев.
5) Масштабирование текста: Программа корректно масштабирует текст с подсветкой синтаксиса при изменении размера окна, обеспечивая удобство просмотра.

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной выпускной квалификационной работы была разработана программа для подсветки синтаксиса алгоритмических языков в образовательных целях. Программа представляет собой инструмент, который позволяет студентам и преподавателям более наглядно и эффективно работать с исходным кодом, а также улучшает восприятие и понимание программных конструкций.

В процессе работы были достигнуты следующие основные результаты:

1) разработан интерфейс приложения, включающий функции загрузки и отображения файлов, подсветки синтаксиса и удобное переключение между различными языками программирования;
2) использованы современные технологии, такие как PyQt6 для создания графического интерфейса и Pygments для подсветки синтаксиса, что обеспечило высокую производительность и качество отображения;
3) проанализированы существующие подходы к подсветке синтаксиса и выбраны оптимальные решения для реализации функциональности приложения;
4) проведено тестирование приложения, в результате чего подтверждена его работоспособность и стабильность.

В дальнейшем планируется расширение функциональности приложения, внедрение дополнительных возможностей для адаптации к различным образовательным потребностям и улучшение пользовательского интерфейса для повышения удобства использования.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Python Documentation. [Электронный ресурс] URL: https://docs.python.org/3/ (дата обращения: 11.03.2024 г.)
2. C++ Documentation. [Электронный ресурс] URL: https://en.cppreference.com/w/ (дата обращения: 11.03.2024 г.)
3. SQL Documentation. [Электронный ресурс] URL: https://www.postgresql.org/docs/ (дата обращения: 16.04.2024 г.)
4. PyQt6 Documentation. [Электронный ресурс] URL: https://www.riverbankcomputing.com/static/Docs/PyQt6/ (дата обращения: 17.04.2024 г.)
5. Pygments Documentation. [Электронный ресурс] URL: https://pygments.org/docs/ (дата обращения: 17.04.2024 г.)
6. HTML Documentation. [Электронный ресурс] URL: https://developer.mozilla.org/docs/Web/HTML (дата обращения: 17.04.2024 г.)
7. CSS Documentation. [Электронный ресурс] URL: https://developer.mozilla.org/docs/Web/CSS (дата обращения: 17.04.2024 г.)
8. Вигерс Карл. Разработка требований к программному обеспечению. – Москва: Издательско-торговый дом Русская редакция, 2004. – 576 с.
9. Старолетов С.М. Основы тестирования и верификации программного обеспечения. – Санкт-Петербург: Изд-во Лань, 2020. – 344 с.
10. Дронов В.А. Python 3. Самое необходимое. – БХВ-Петербург, 2015. – 464 с.
11. Любанович Б. Простой Python. Современный стиль программирования, 2-е издание. – Прогресс книга, 2021. – 132 с.
12. Анатолий П. Разработка кроссплатформенных мобильных и настольных приложений на Python. Практическое пособие. – Литрес, 2022. – 580 с.
13. Прохоренок Н.А., Дронов В.А. Python 3 и PyQt 5. Разработка приложений. 2-е издание. – БХВ-Петербург, 2019. – 832 с.

14. Edward Chang. Fresher PyQt6 A Beginner's Guide to PyQt6. – Amazon Digital Services LLC – Kdp, 2022. – 144 с.
15. Дэвид Бизли, Брайан К. Джонс. Python. Книга рецептов. – ДМК Пресс, 2019. – 648 с.