**СОДЕРЖАНИЕ**

Contents

[Введение 6](#_Toc132227048)

[1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству 8](#_Toc132227049)

[1.1 Анализ литературных источников 8](#_Toc132227050)

[1.2 Анализ существующих аналогов 11](#_Toc132227051)

Введение

Современное программное обеспечение становится все более сложным, и понимание его работы может представлять трудность даже для опытных разработчиков. В связи с этим возникает потребность в инструментах, которые позволяют облегчить анализ и понимание кода. Один из таких инструментов — это визуальный подход к обучению программированию, который позволяет представлять сложные алгоритмы в графическом виде.

Однако, визуальный подход к программированию может стать еще более эффективным, если используется автоматическая генерация блок-схем на основе исходного кода. Такой подход может значительно упростить понимание и анализ кода, особенно в случае работы с большими проектами.

Для того, чтобы разработать автоматический генератор блок-схем, необходимо решить несколько задач, связанных с лексическим и синтаксическим анализом кода. Важно также разработать алгоритм генерации блок-схем, который должен быть эффективным и точным, чтобы генерировать блок-схемы с минимальным числом ошибок.

Актуальность проблемы генерации блок-схем из исходного кода заключается в том, что это позволяет облегчить понимание кода, особенно в случае работы с большими проектами. Также это может быть полезным для отладки и тестирования программного обеспечения. В современных условиях, когда количество программных продуктов растет с каждым днем, разработка инструментов, которые позволяют упростить анализ кода, становится все более актуальной

Цель данного дипломного проекта заключается в разработке веб-сервиса, который будет автоматически генерировать блок-схемы из исходного кода программ. Генерация блок-схем из кода позволит облегчить понимание и анализ сложных алгоритмов, особенно в случае работы с большими проектами. Разработка веб-сервиса будет иметь большую практическую значимость, поскольку он может быть полезен для студентов, которые учатся программированию, и для опытных разработчиков, которые работают с большими программными проектами. Помимо этого, создание веб-сервиса может быть полезным для отладки и тестирования программного обеспечения. Общая идея проекта заключается в использовании визуального подхода к программированию, чтобы представлять сложные алгоритмы в графическом виде и тем самым облегчить их понимание и анализ.

Разработка данного дипломного проекта позволит получить более глубокие знания в области теории компиляторов и разработки клиент-серверных приложений. В процессе выполнения проекта будет решаться ряд задач, связанных с разработкой синтаксиса собственного языка программирования, алгоритма обхода синтаксического дерева, клиентской части веб-сервиса и тестирования полученного решения.

Разработка синтаксиса собственного языка программирования является одной из ключевых задач проекта, так как язык должен быть достаточно выразительным для описания различных алгоритмов и при этом легко понятным для пользователей. Для решения этой задачи необходимо изучить теорию компиляторов и научиться применять ее в практике.

Алгоритм обхода синтаксического дерева и построения блок-схемы также является важным компонентом проекта. Для его реализации необходимо будет изучить различные алгоритмы обхода деревьев и выбрать оптимальный вариант для данной задачи.

Разработка клиентской и серверной частей веб-сервиса также представляет собой интересный вызов, который позволит изучить принципы разработки клиент-серверных приложений, их взаимодействия и оптимизации.

Таким образом, выполнение данного дипломного проекта позволит не только создать полезный инструмент для программистов и студентов, но и улучшить практические навыки в области теории компиляторов и разработки клиент-серверных приложений.

.

1 Анализ прототипов, литературных источников и формирование требований к проектируемому программному средству

1.1 Анализ литературных источников

**1.1.1** Анализ алгоритмов разбора языка

Компиляция языка программирования — это процесс преобразования исходного кода на одном языке в эквивалентный код на другом языке или в машинный код. Она является важным шагом в создании программного обеспечения и позволяет перевести код, написанный на человеко-читаемом языке, в язык, который может быть исполнен компьютером. В контексте генерации схемы алгоритма, компиляция может быть рассмотрена как процесс преобразования исходного кода в графическое представление алгоритма в виде блок-схемы.

Процесс компиляции включает в себя несколько этапов: лексический анализ, синтаксический анализ, семантический анализ, оптимизацию и генерацию кода. Во время лексического анализа исходный код разбивается на токены, которые представляют отдельные элементы языка, такие как идентификаторы, числа и ключевые слова. Синтаксический анализ проверяет правильность расположения токенов и их соответствие грамматике языка. Семантический анализ проверяет, что код соответствует правилам языка и выполняет необходимые проверки на типы данных, области видимости переменных и другие аспекты, которые могут повлиять на правильность работы программы.

Существует два основных подхода к созданию синтаксических анализаторов для языков программирования: ручной и автоматизированный. В первом случае, разработчик самостоятельно пишет код парсера, который выполняет лексический и синтаксический анализ входного кода. Этот подход позволяет получить более точный контроль над процессом анализа и предоставляет возможность для оптимизации производительности. Однако, ручное написание парсера может быть очень трудоемким и затратным, особенно для языков с богатым и сложным синтаксисом.

Альтернативным подходом является автоматизированное создание парсера с помощью генераторов парсеров. Этот метод основан на грамматиках, описывающих синтаксис языка. Разработчик определяет грамматику языка в специальном формате, который может быть использован генератором парсера для автоматического создания соответствующего кода парсера.

На практике, генераторы парсеров, такие как ANTLR, часто используются для автоматического создания парсеров на основе грамматики языка, что значительно упрощает и ускоряет процесс разработки. Генераторы парсеров позволяют автоматически создавать парсеры для большинства языков программирования, обеспечивая высокую точность и скорость работы. Использование генераторов парсеров позволяет разработчикам сосредоточиться на более важных задачах, таких как разработка лексического анализа и семантического анализа.

Для семантического анализа не существует общих подходов, так как его цель сильно зависит от конкретного языка программирования и контекста, в котором он используется. В разрабатываемом веб-сервисе для генерации блок-схемы из исходного кода нет необходимости проводить полный семантический анализ, так как основной целью является только построение структуры алгоритма в виде графа. Для этого достаточно провести лексический и синтаксический анализ исходного кода, а также выполнить минимальный семантический анализ для определения типа блоков, таких как определение переменных, вызов функций и так далее.

**1.1.2** Анализ языков программирования

Одним из основных факторов при выборе языка программирования является его удобство в использовании. Языки, которые имеют простой и понятный синтаксис, а также богатые возможностями и инструментами, обычно считаются более привлекательными для программистов. Например, язык Python, благодаря своей читабельности и простоте в написании кода, популярен среди начинающих и опытных программистов.

Кроме удобства использования, важным аспектом является поддержка языка сообществом разработчиков и его популярность. Языки программирования с большой и активной пользовательской базой обычно имеют обширную документацию, богатую выборку инструментов и разнообразных библиотек. Например, язык Java, благодаря своей популярности, имеет обширную экосистему, которая включает в себя множество инструментов и библиотек.

Важным критерием при выборе языка программирования для создания веб-сервиса, позволяющего генерировать схему-алгоритма из исходного кода, является его легкость синтаксического и семантического разбора. Для достижения этой цели был проведен анализ существующих языков программирования, популярных в различных областях программирования. Было выяснено, что существуют языки с удобным и легким синтаксисом, который позволяет легко производить семантический и синтаксический анализ кода, но с другой стороны есть ряд правил, которых стоит избегать. Например, некоторые языки программирования используют многословные ключевые слова или сложные правила синтаксиса, как неоднозначности в определении границ выражений или использование разных символов для обозначения одного и того же действия, что затрудняет понимание и анализ кода. Кроме того, использование сложных конструкций, таких как макросы может усложнить семантический анализ, за счет изменения структуры кода на основе параметров.

С учетом этих факторов, при разработке нового языка для создания блок-схем, важно уделить внимание не только удобству синтаксиса, но и точности описания логических конструкций. Необходимо убедиться, что язык позволяет программисту легко выразить свои мысли и идеи в виде блок-схем, без необходимости тратить много времени на поиск нужных инструкций и ключевых слов, но при этом позволяет проводить генерацию схемы-алгоритма с наименьшим количеством проблем.

**1.1.3** Анализ веб-сервиса

Веб-сервис – это программное обеспечение, которое позволяет различным приложениям взаимодействовать друг с другом через сеть, используя стандартные протоколы, например HTTP. Веб-сервис может быть реализован на разных языках программирования и может работать на разных операционных системах.

Клиент-серверная архитектура – это модель, которая разделяет приложение на две части: клиентскую и серверную. Клиентская часть представляет интерфейс пользователя, который взаимодействует с приложением, а серверная часть выполняет операции, обрабатывает запросы клиента и возвращает результаты.

Клиент-серверная архитектура является одной из наиболее популярных архитектур для создания веб-приложений и сервисов. Это связано с тем, что она предоставляет отличный уровень безопасности и масштабируемости. Каждая часть приложения выполняется в отдельном окружении, что предотвращает возможность несанкционированного доступа к чувствительной информации. Кроме того, клиент-серверная архитектура обеспечивает высокую производительность и гибкость в работе с данными.

Существуют различные языки программирования и фреймворки для реализации клиент-серверной архитектуры, некоторые из наиболее популярных языков включают JavaScript, Python, Java, Ruby, PHP и C#. Они используются в сочетании с различными фреймворками, такими как Node.js, Flask, Spring, Ruby on Rails, Laravel, ASP.NET и многими другими.

Spring Kotlin предоставляет разработчикам множество возможностей для создания быстрых и эффективных веб-сервисов. Он основан на Java и позволяет использовать всю мощь Java-фреймворка Spring, который является одним из наиболее популярных в мире, а также генератор парсеров ANTLR. Кроме того, Kotlin предоставляет более простой и понятный синтаксис, что упрощает разработку и поддержку приложения.

Angular обеспечивает высокую производительность, быструю загрузку страниц и хорошую масштабируемость, а также позволяет создавать модульные и переиспользуемые компоненты, что существенно упрощает разработку.

Таким образом, использование Spring Kotlin и Angular для разработки веб-сервиса генерации схемы алгоритма из исходного кода является оптимальным решением. Эти фреймворки обеспечивают высокую скорость разработки, хорошую масштабируемость и простоту в поддержке и сопровождении.

1.2 Анализ существующих аналогов

Произведение анализа существующих аналогов является важным шагом в разработке нового веб-сервиса генерации схемы алгоритма из исходного кода. Это поможет выявить особенности и проблемы имеющихся решений, а также определить возможности для улучшения и дальнейшего развития разрабатываемого веб-сервиса.

Существующие аналоги могут предоставлять различные функциональные возможности, иметь различные методы решения поставленных задач, и в то же время иметь некоторые ограничения и недостатки. Изучение конкурентов позволит определить лучшие практики и опыт, а также учесть ошибки и недочеты в разработке собственного веб-сервиса.

**1.2.1** Веб-приложение «Code2Flow»