**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

К курсовому проектированию

По курсу «Логика и основы алгоритмизации в инженерных задачах»

на тему «Генератор блок-схем используя абстрактно-синтаксическое дерево и его обход в глубину»

Выполнил:

Костин К.А. 23ВВВ4

Принял:

Пенза 2024

**Содержание**

**Оглавление**

[Реферат 5](#_Toc182476855)

[Введение 6](#_Toc182476856)

[Постановка задачи 7](#_Toc182476857)

[Теоретическая часть задания 8](#_Toc182476858)

[Описание алгоритма программы 9](#_Toc182476859)

[Описание программы 10](#_Toc182476860)

# **Реферат**

Отчет хх стр, хх рисунков.

Всяко разные умные слова

Цель исследования ­ разработка программы компилятора, способную компилировать код на заданном языке программирования в формат JSON для веб-сервиса редактора блок-схем, используя tree-sitter для генерации абстрактно-синтаксического дерева и дальнейший обход его в глубину\

В работе рассмотрены правила, на основе которых лексические конструкции языка преобразуются в блоки.

# **Введение**

Тут я буду писать чтото про разные компиляторы

# **Постановка задачи**

Требуется разработать программу-компилятор, которая будет преобразовывать лексические конструкции в блоки

Программу можно запустить с двумя ключами, первый – язык, на котором написан исходный код, второй – относительный путь до файла. Если ключи не заданы, программа запросит все данные в соответствующем диалоге. После завершения работы в папке, из которой запускалась программа, должен создоваться выводной файл в формате .json, с таким же именем, что и входной файл. Этот .json можно поместить в соответствующий сервис для дальнейшего редактирования. Необходимо предусмотреть базовые проверки на правильность написанного кода.

Устройство ввода ­ клавиатура

# **Теоретическая часть задания**

Мне тут теорию компиляторов писать?

# **Описание алгоритма программы**

Для программной реализации алгоритма понадобиться 3 логических блока: блок подготовки, блок предкомпиляции и блок пост компиляции.

**Блок подготовки:** в этом блоке происходит общение с пользователем, чтение исходного кода в оперативную память, подготовка парсера tree-sitter и запуск **блока предкомпиляции**

**Блок предкомпиляции:** в этом блоке происходит парсинг исходного кода в абстрактно-синтаксическое дерево (AST) и анализ этого дерева по определенным правилам, избегая анализа синтаксиса. Результатом работы этого блока будет вектор(Vec) структур, полями которых будут: координата х, координата у, тип блока, текст блока. Этот вектор универсален для всех языков. Или в случае если код написан неверно, в результате пользователь увидит ошибку, с указанием причины, почему именно произошел сбой.

**Блок посткомпиляции:** в случае успешной предкомпиляции вектор передается в этот блок, где происходит преобразование каждого элемента вектора в блоки .json в формате строки (String). Потом эта строка передается в функцию main, где происходит запись в файл и оповещение пользователя об успешном завершении.

Ниже представлен ?псевдокод(можно просто код?)?

# **Описание программы**

Для написания программы использован язык программирования Rust. Язык программирования Rust ­ безопасный в отношении памяти язык программирования, который стал любимым языком перепрограммирования у большинства программистов по версии StackOverflow благодаря своим концепциям владения взаимствования, что позволяет писать приложения, лишенные проблемы утечек памяти.

Проект был создан в виде консольного приложения Linux.

Данная программа является многомодульной, поскольку состоит из нескольких модулей: main.rs, lang\_vec\_stuf.rs, mk\_json\_blocks.rs.