Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
Национальный Исследовательский Университет ИТМО
Факультет программной инженерии и компьютерной техники

Отчёт к практическому заданию №1

Выполнил:
Студент 1 курса магистратуры «Системное и прикладное программное обеспечение» группы P4115
Эрнандес Гарсия Даниэль Дельевич

Преподаватель:
к. т. н., доцент ФПИиКТ
Кореньков Юрий Дмитриевич

Санкт-Петербург
2026

Цели: использовать средство синтаксического анализа по выбору, реализовать модуль для разбора текста в соответствии с языком по варианту. Реализовать построение по исходному файлу с текстом синтаксического дерева с узлами, соответствующими элементам синтаксической модели языка. Вывести полученное дерево в файл в формате, поддерживающем просмотр графического представления.

Задачи:

Изучить выбранное средство синтаксического анализа

a. Средство должно поддерживать программный интерфейс, совместимый с языком Си

b. Средство должно параметризоваться спецификацией, описывающей синтаксическую структуру

разбираемого языка

c. Средство может функционировать посредством кодогенерации и/или подключения

необходимых для его работы дополнительных библиотек

d. Средство может быть реализовано с нуля, в этом случае оно должно использовать

обобщённый алгоритм, управляемый спецификацией

2. Изучить синтаксис разбираемого по варианту языка и записать спецификацию для средства

синтаксического анализа, включающую следующие конструкции:

a. Подпрограммы со списком аргументов и возвращаемым значением

b. Операции контроля потока управления – простые ветвления if-else и циклы или аналоги

c. В зависимости от варианта – определения переменных

d. Целочисленные, строковые и односимвольные литералы

e. Выражения численной, битовой и логической арифметики

f. Выражения над одномерными массивами

g. Выражения вызова функции

3. Реализовать модуль, использующий средство синтаксического анализа для разбора языка по варианту

a. Программный интерфейс модуля должен принимать строку с текстом и возвращать структуру,

описывающую соответствующее дерево разбора и коллекцию сообщений ошибке

b. Результат работы модуля – дерево разбора – должно содержать иерархическое

представление для всех синтаксических конструкций, включая выражения, логически

представляющие собой иерархически организованные данные, даже если на уровне средства

синтаксического анализа для их разбора было использовано линейное представление

4. Реализовать тестовую программу для демонстрации работоспособности созданного модуля

a. Через аргументы командной строки программа должна принимать имя входного файла для

чтения и анализа, имя выходного файла записи для дерева, описывающего синтаксическую

структуру разобранного текста

b. Сообщения об ошибке должны выводиться тестовой программной (не модулем, отвечающим

за анализ!) в стандартный поток вывода ошибок

Описание работы: Разработана программа, которая использует tree-sitter-cli (**tree-sitter** библиотеку) с описанием входной грамматики в файле grammar.js в корне проекта для синтаксического разбора исходного кода языка, соответствующего грамматике, и построения на основе выделенной структуры mermaid-диаграммы

Аспекты реализации и результаты: вся основная суть работы заключается в написании правильного **grammar.js** файла, который соответствует грамматики языка, указанного в задании. На его основе можно сгенерировать код (на языке Си) для парсера грамматики используя одно команду npx tree-sitter generate. В дальнейшем нужно было подключить к основному файлу создаваемой программы сгенерированный парсер и исходный код библиотеки tree-sitter, который уже не зависит от грамматики. Написанный таким образом модуль считывает исходный код языка, соответствующего грамматике, использует для его анализа код сгенерированного парсера и основной код библиотеки, получив таким образом структуру данных (из tree sitter), соответствующую всему исходному коду программы, переданной в файле для анализа, и далее на основе этой структуры пишет файл mermaid, где можно увидеть, как программа считала исходный код (какие получились сущности в памяти).

Примеры входного и выходного результата можно посмотреть в репозитории в файлах input.mylang и ast.mmd.

Выводы: в результате проделанной работы были получены навыки написания ПО для синтаксического анализа текстов в основе которых лежат формальные грамматики. Эти навыки позволяют в результате анализа текстов формальных грамматики формировать в памяти компьютеров структуру исходного текста в терминах грамматики и которые в дальнейшем могут быть использованы при написании компиляторов.