Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

**«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт космических и информационных технологий

институт

Программная инженерия

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ** **РАБОТЕ №5**

Синтаксический анализ контекстно-свободных языков

тема

Преподаватель А. С. Кузнецов

подпись, дата инициалы, фамилия

Студент КИ23-17/1Б, 032320072 М. А. Мальцев

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2025

# Цель

Исследование контекстно-свободных грамматик и алгоритмов синтаксического анализа контекстно-свободных языков.

# Задания

Часть 1.

Необходимо с использованием системы JFLAP, построить LL(1)-грамматику, описывающую заданный язык, или формально доказать невозможность этого. Полученная грамматика не должна повторять SLR(1)-грамматику, конструируемую в части 3. Ассоциативность операций на усмотрение разработчика.

Вариант 2.

Язык оператора присваивания, в правой части которого задано логическое выражение. Элементами выражений являются целочисленные константы в шестнадцатеричной системе счисления, имена переменных из одного символа (от g до k), знаки операций и скобки для изменения порядка вычисления подвыражений. Операции (в сторону уменьшения приоритета): отрицание, мультипликативные, аддитивные, присваивание.

Часть 2.

Предложить программную реализацию метода рекурсивного спуска для распознавания строк заданного языка. Представить формальное доказательство принадлежности к классу LL(1)-грамматики, лежащей в основе синтаксического анализа заданного языка. Во всех случаях язык должен состоять из последовательностей выражений. В качестве разделителя может выступать символ новой строки, точка с запятой или любой другой символ, не задействованный в других лексемах. Ассоциативность операций на усмотрение разработчика. Результатом работы синтаксического анализатора является выдача сообщения «Accepted» или «Rejected».

Вариант 2.

Язык логических выражений, элементами которых являются целочисленные константы в шестнадцатеричной, двоичной или десятичной системах счисления, имена переменных из 1-2 символов, знаки операций и скобки для изменения порядка вычисления подвыражений. Операции (в сторону уменьшения приоритета): отрицание, мультипликативные, аддитивные, присваивание

Часть 3.

Необходимо с использованием системы JFLAP, построить SLR(1)-грамматику, описывающую заданный язык, или формально доказать невозможность этого. Во всех случаях реализуется язык, состоящий из последовательностей операторов присваивания. В качестве разделителя может выступать символ новой строки, точка с запятой или любой другой символ, не задействованный в прочих лексемах. В качестве L-значения оператора присваивания выступает только имя переменной. В правой части оператора присваивания указывается выражение, элементы которых оговариваются в каждом варианте задания. Ассоциативность операций на усмотрение разработчика. Полученная грамматика не должна повторять LL(1)-грамматику, конструируемую в части 1.

Вариант 2.

Элементами логического выражения являются целочисленные константы в 8- и 16-чной системах счисления, имена переменных из одного символа (от g до k), знаки операций и скобки для изменения порядка вычисления подвыражений. Операции (в сторону уменьшения приоритета): отрицание, мультипликативные, аддитивные, присваивание.

# Ход выполнения

## Создание LL(1)-грамматики

Сначала была сделана первая часть задания практической работы. За знаки операций были приняты следующие: «~» (отрицание), «|» (аддитивная, ИЛИ), «&» (мультипликативная, И) и «=» (присваивание). Были использованы круглые скобки. Составленная LL(1)-грамматика в JFLAP показана на рисунке 1.

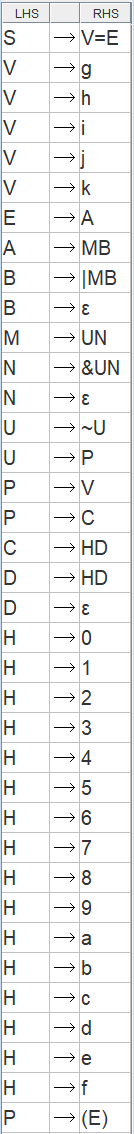


Рисунок 1 – Составленная LL(1)-грамматика

Затем был выполнен «Build LL(1) Parse» и получено множество первых порождаемых символов и символов последователей и таблица синтаксического анализа. Они показаны на рисунке 2 и 3.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 2 – Множество первых порождаемых символов и символов последователей составленной грамматики

Изображение выглядит как текст, снимок экрана, число, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.Рисунок 3 – Таблица синтаксического анализа составленной грамматики

Как можно заметить, в каждой ячейке таблицы не более одной записи, что говорит о принадлежности грамматики LL(1)-грамматике.

Дальше было проведено распознавание тестовых цепочек, путём нажатия на кнопку «Parse» и ввода 6 тестовых цепочек для проверки. Результаты теста показаны на рисунках 4, 5, 6, 7, 8 и 9.

Изображение выглядит как текст, снимок экрана, число, Параллельный

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 4 – Тест для цепочки «g=~(h|a2)&c2»

Изображение выглядит как текст, снимок экрана, число, Параллельный

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 5 – Тест для цепочки «g=h»

Изображение выглядит как текст, снимок экрана, число, Параллельный

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 6 – Тест для цепочки «k=~a23»

Изображение выглядит как текст, снимок экрана, число, Параллельный

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 7 – Тест для цепочки «j=(k|12))»

Изображение выглядит как текст, снимок экрана, линия, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 8 – Тест для цепочки «~(12|k=2)»

Изображение выглядит как текст, снимок экрана, число, Параллельный

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 9 – Тест для цепочки «h=~j&»

В итоге все тесты были успешно пройдены, и полученная грамматика правильно определяет исходный язык.

## Создание SLR(1)-грамматики

Затем была выполнена 3 часть задания практической работы. В качестве знаков операций были приняты те же знаки, что и в прошлой части задания. В качестве разделителя использовался «;». Составленная SLR(1)-грамматика в JFLAP показана на рисунке 10.

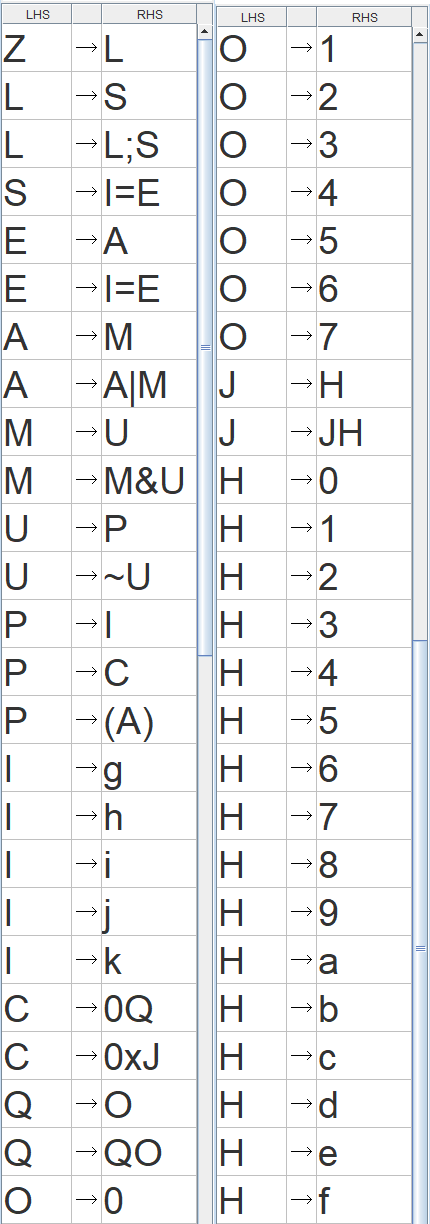


Рисунок 10 – Составленная RLS(1)-грамматика

Затем был выполнен «Build SLR(1) Parse» и получено множество первых порождаемых символов и символов последователей, канонический набор LR(0)-ситуаций и таблица синтаксического анализа. Они показаны на рисунке 11, 12 и 13.

Изображение выглядит как текст, снимок экрана, число, дисплей

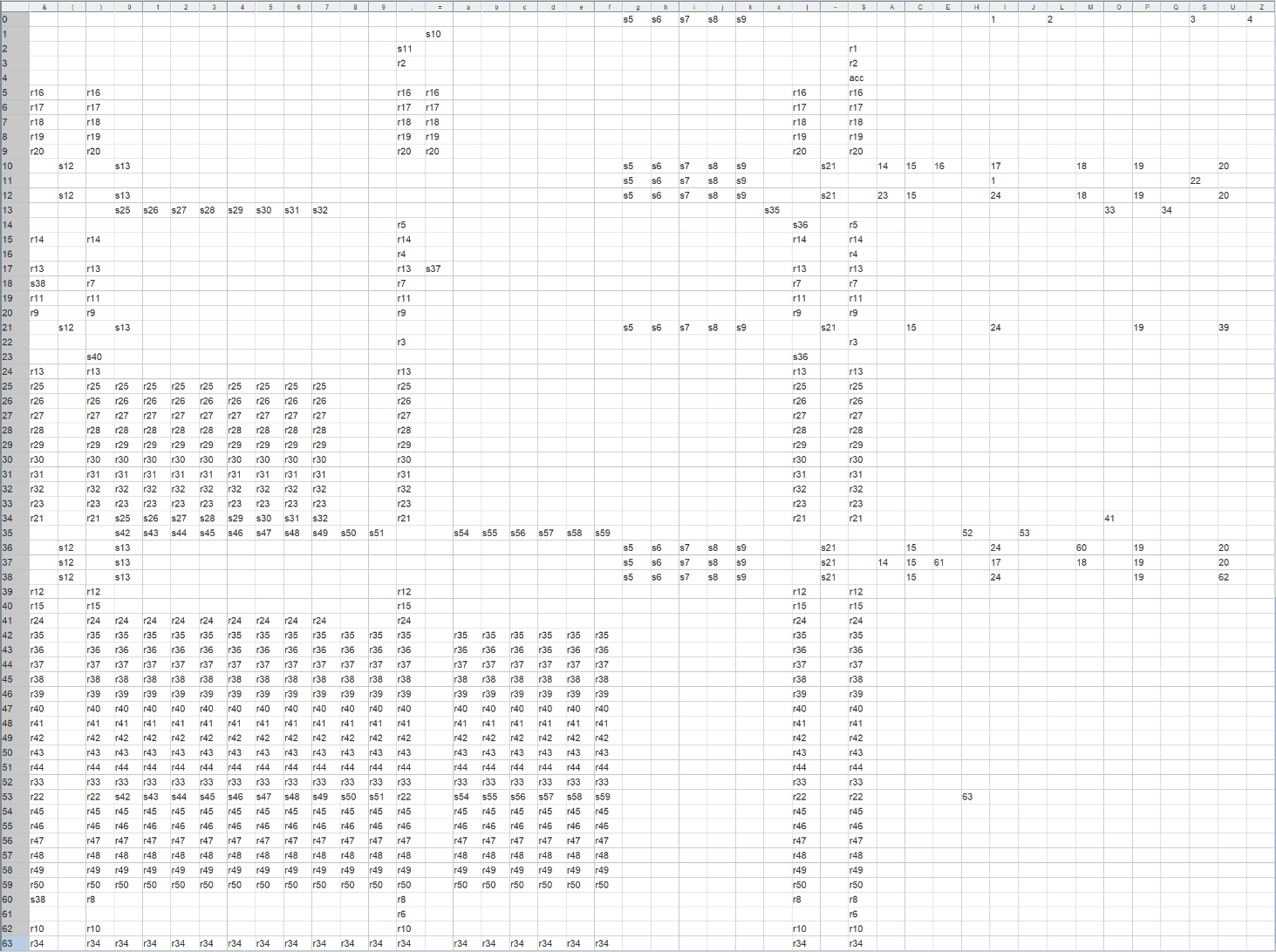
Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 11 – Множество первых порождаемых символов и символов последователей составленной грамматики

Изображение выглядит как диаграмма, линия, карта, рисунок

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 12 – Канонический набор LR(0)-ситуаций

Рисунок 13 – Таблица синтаксического анализа составленной грамматики

Дальше было проведено распознавание тестовых цепочек, путём нажатия на кнопку «Parse» и ввода 6 тестовых цепочек для проверки. Результаты теста показаны на рисунках 14, 15, 16, 17, 18 и 19.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 14 – Тест для цепочки «g=~(h|071)&0xa2f»

Изображение выглядит как текст, число, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 15 – Тест для цепочки «k=j=02&g»

Изображение выглядит как текст, снимок экрана, число, линия

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 16 – Тест для цепочки «k=~h|0x1;j=k&07»

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 17 – Тест для цепочки «g=h;i=02;j=0xa»

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 17 – Тест для цепочки «g=~k;i»

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 17 – Тест для цепочки «g=~j=k»

В итоге все тесты были успешно пройдены, и полученная грамматика правильно определяет исходный язык.

## Программная реализация синтаксического анализатора методом рекурсивного спуска

Затем была выполнена 2 часть задания практической работы. Сначала была составлена LL(1)-грамматика, определяющая язык, описанный во 2 части задания, которая затем легла в основу разработанного синтаксического анализатора. Описание составленной грамматики показано на рисунке 18.

Изображение выглядит как текст, снимок экрана, меню

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 18 – Составленная LL(1)-грамматика

Также было доказано, что данная грамматика принадлежит LL(1)-грамматикам. Для этого она была написана в JFLAP и проанализирована.

Во-первых, данная грамматика не имеет левой рекурсии и неоднозначности. Во-вторых, вычисленные множества FIRST для каждого нетерминала не пересекаются. В-третьих, вычисленные множества FOLLOW для каждого нетерминала выполняли равенство: «Если для продукции A → α есть ε ∈ FIRST(α), то FOLLOW(A) ∩ FIRST(β) = ∅ для всех других продукций A → β». В-четверых, в таблице синтаксического анализа каждая ячейка содержит не больше одной продукции, как показано на рисунке 19.

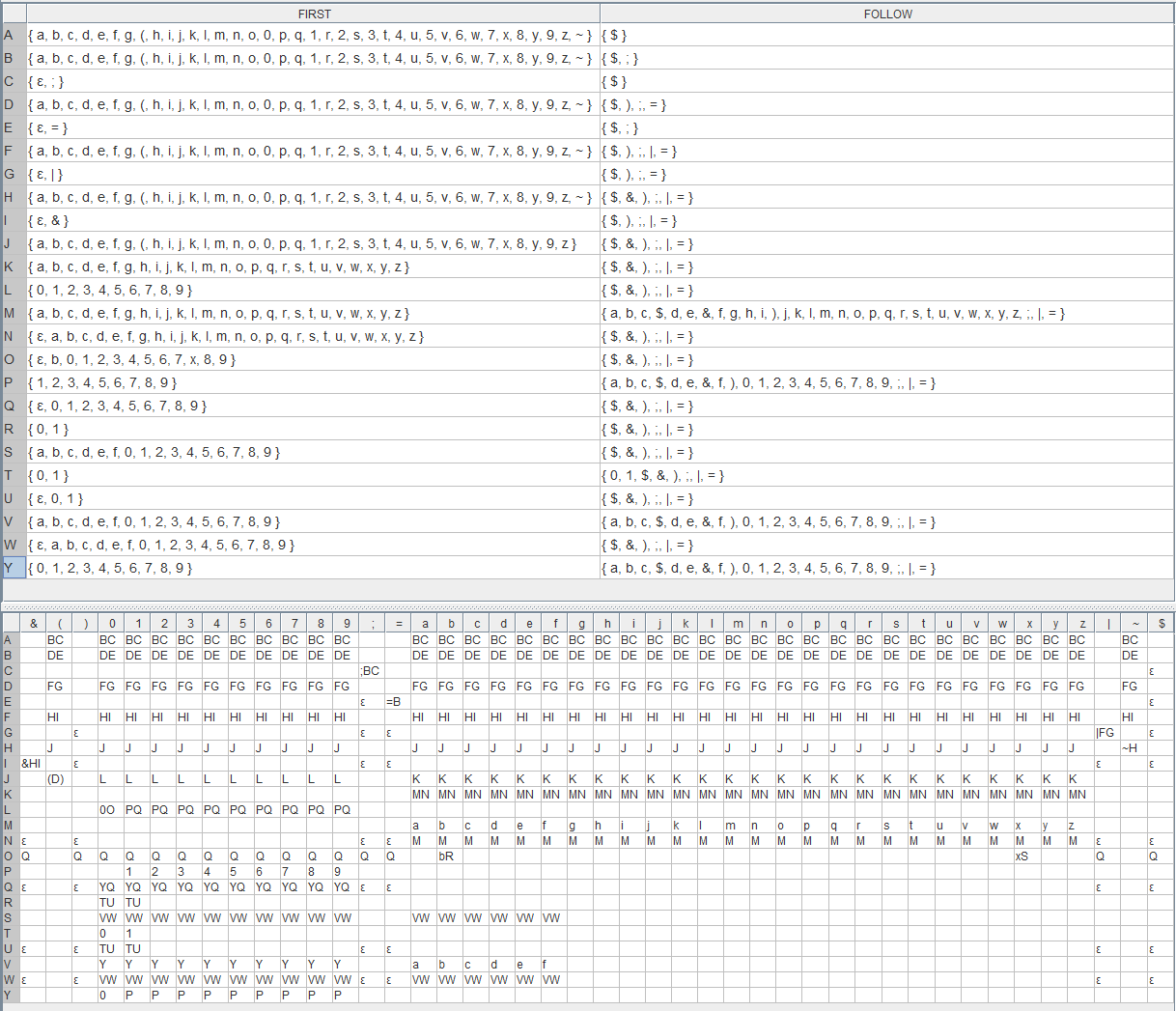


Рисунок 19 – Таблица синтаксического анализа и множество первых порождаемых символов и символов последователей составленной грамматики

После составления грамматики был написан код программы на С++, реализующей синтаксический анализатор методом рекурсивного спуска на основе составленной LL(1)-грамматики. Он показан на рисунках 20, 21 и 22.

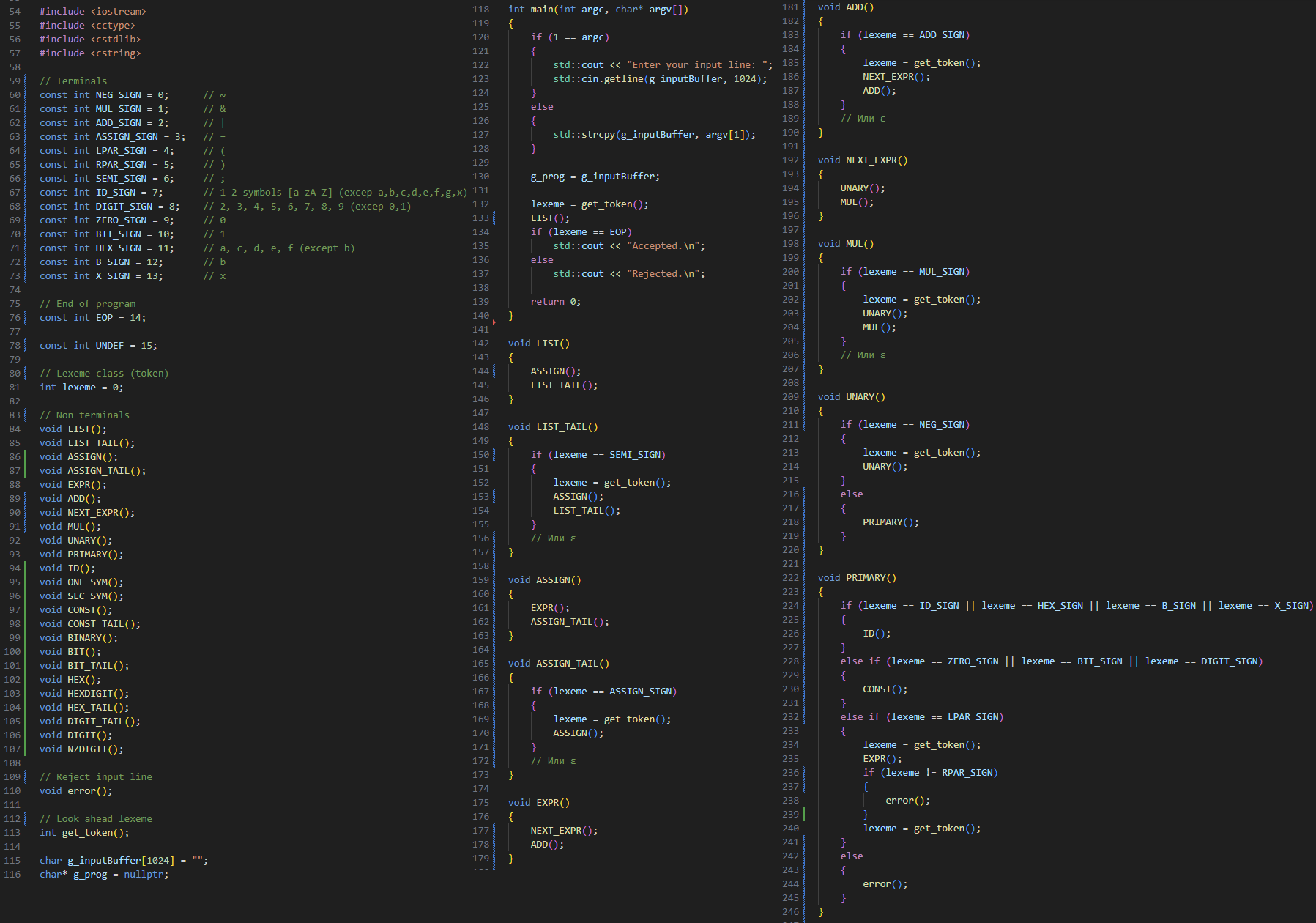


Рисунок 20 – Первая часть кода реализованной программы на С++

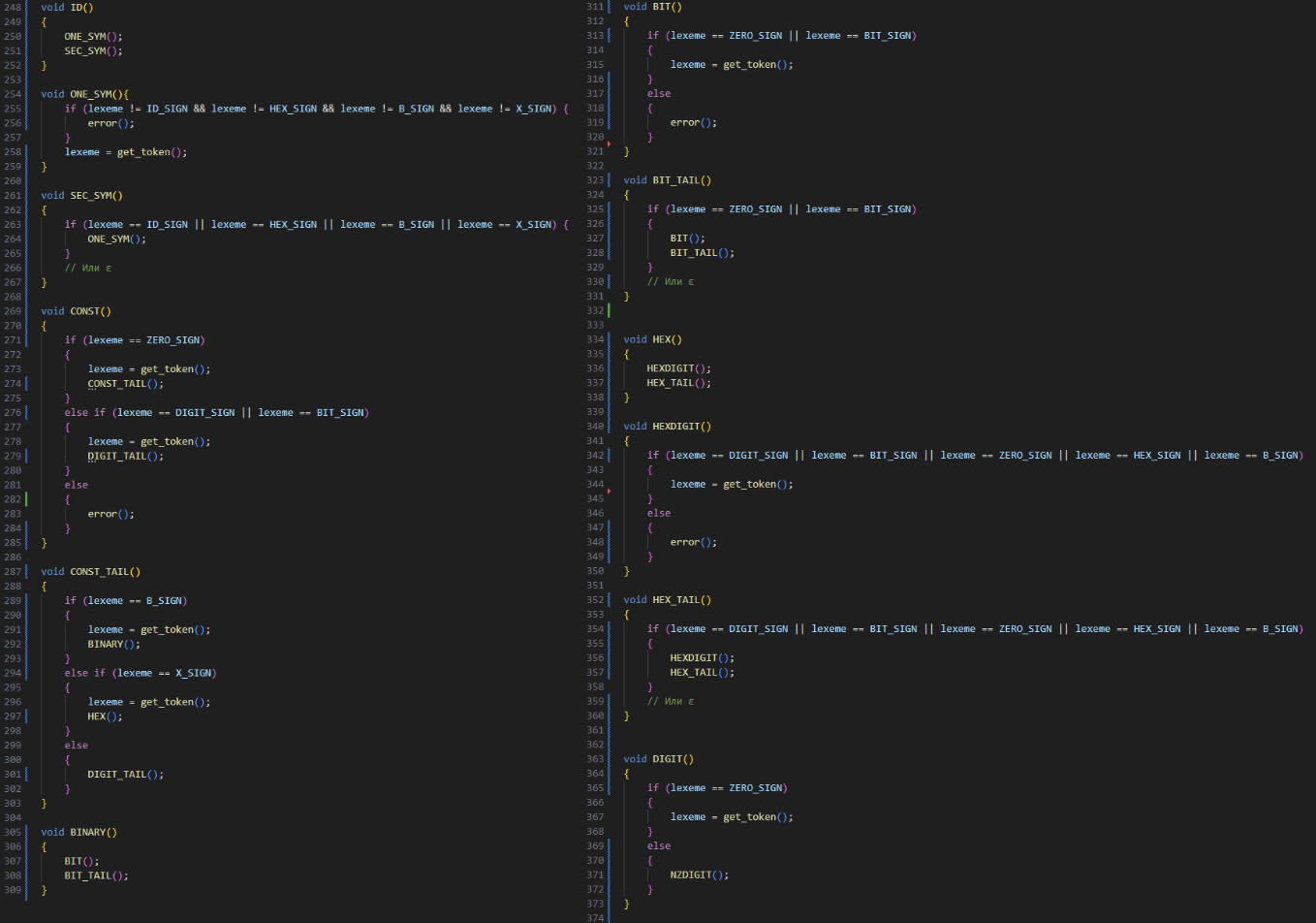


Рисунок 21 – Вторая часть кода реализованной программы на С++

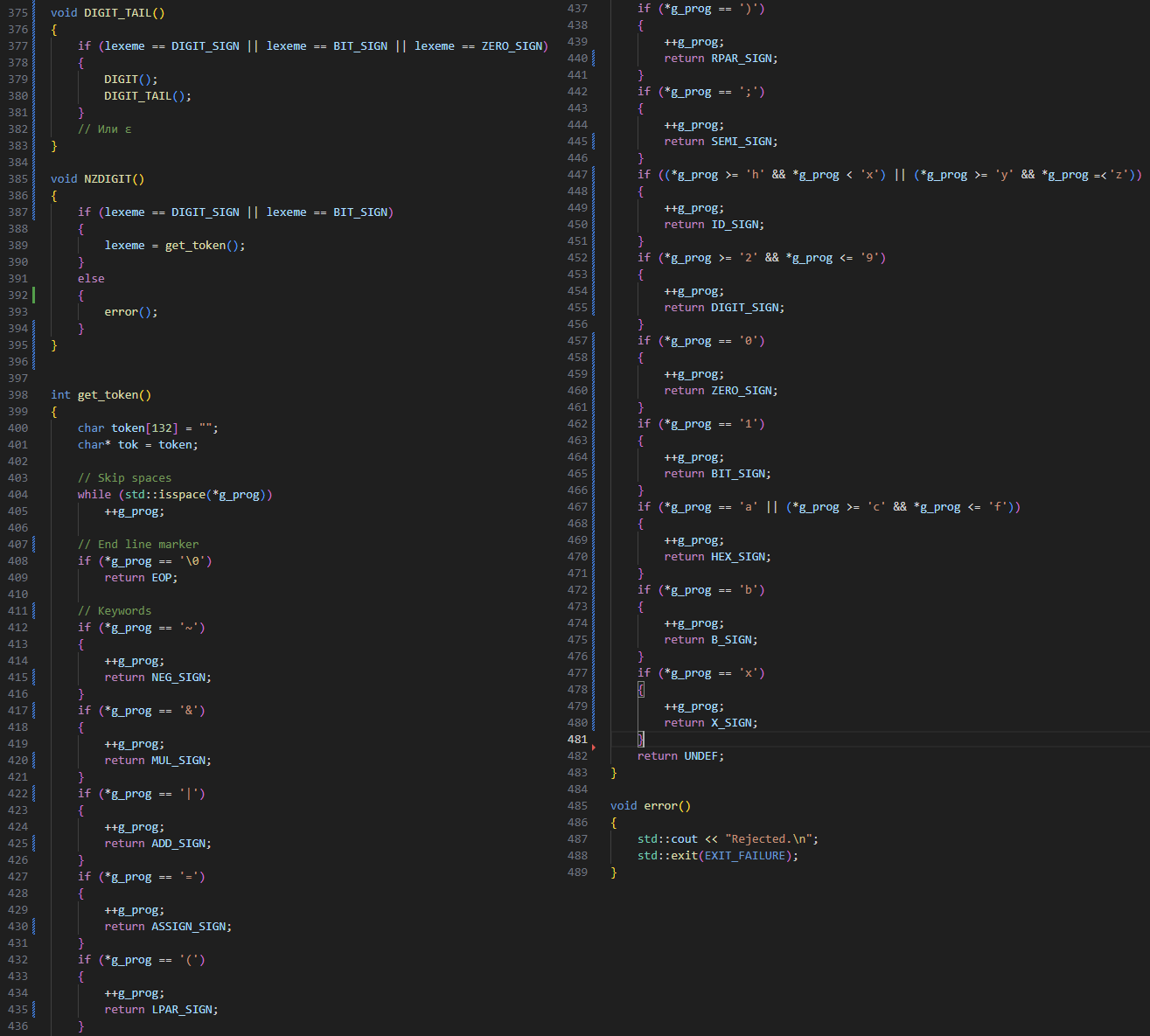


Рисунок 22 – Третья часть кода реализованной программы на С++

На рисунке 23 показаны тестовые примеры работы программы на тестовых цепочках.

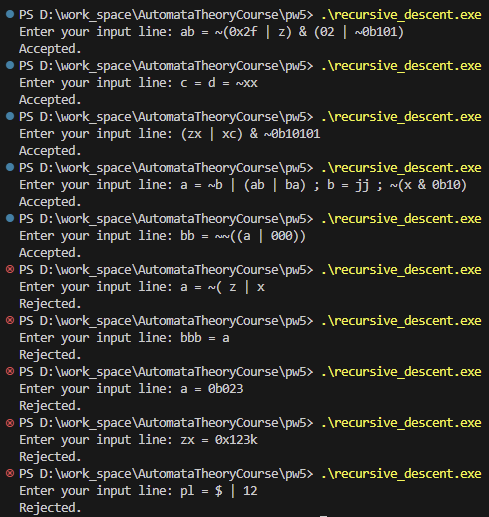


Рисунок 23 – Тест программы на произвольных цепочках

Все тесты были пройдены программой успешно. Принимаются только логические выражения и есть возможность разделения логических выражения с помощью точки с запятой.

# Выводы

В ходе данной практической работы были исследованы свойства универсальных алгоритмов синтаксического анализа контекстно-свободных языков, построены LL(1) и SLR(1) грамматики, а также изучены шаги формального доказательства принадлежности грамматики к LL(1)-грамматике. Также мы узнали про синтаксический анализатор, работающий методом рекурсивного спуска.