Минобрнауки России

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«САРАТОВСКИЙ национальный исследовательский  
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ**

**Н. Г. ЧЕРНЫШЕВСКОГО»**

Факультет компьютерных наук и информационных технологий

**Основы трансляции языков программирования.**

Лабораторная работа №1. Построение синтаксического анализатора методом рекурсивного спуска.

Студента 4 курса 441 группы

направления 02.03.03—Математическое обеспечение и администрирование

информационных систем

факультета КНиИТ

Поповича Алексея Сергеевича

Саратов 2024

# Задание на лабораторную работу.

С использованием метода рекурсивного спуска реализовать синтаксический анализатор языка, цепочки которого имеют вид, указанный в задании на первую лабораторную работу. Синтаксический анализатор должен принимать на вход последовательность лексем, сформированную лексическим анализатором и восстанавливать дерево разбора цепочки. В случае ошибок во входной цепочке анализатор должен дать пользователю информативное сообщение с указанием причины и места возникновения ошибки.

# Описание грамматики исходного языка.

# Описание всех преобразований исходной грамматики и грамматику в окончательном виде, представленную в расширенной бэкусовой нормальной форме.

**1. Устранение левой рекурсии**

# Контрольные вопросы.

**1) Определение контекстно-свободной грамматики (CFG):**

Контекстно-свободная грамматика — это формальная грамматика, в которой правила вывода имеют вид 𝐴→𝛼, где A — это один нетерминальный символ, а α — произвольная строка, состоящая из терминальных и/или нетерминальных символов. Такие грамматики определяют язык, где структура выражений не зависит от контекста, в котором они встречаются.

**2) Класс грамматик, допускающий построение анализаторов методом рекурсивного спуска:**

Метод рекурсивного спуска может быть применен к грамматикам класса LL(1), то есть к контекстно-свободным грамматикам, которые можно анализировать слева направо (Left-to-right), с выводом с левым упорядочением (Leftmost derivation), и где анализатор принимает решение на основе одного символа вперед (1 lookahead).

**Терминал** — это символ, который не может быть заменён на другие символы в процессе вывода по правилам грамматики. Терминальные символы являются конечными "строительными блоками" строк языка, которые грамматика порождает. В контексте синтаксического анализа терминалы представляют собой конкретные символы или лексемы (например, ключевые слова, операторы, идентификаторы), которые встречаются в исходном коде или входных данных программы.

Пример терминалов для языка программирования может включать такие символы, как if, else, +, ;, = и т. д. Терминалы не подлежат дальнейшему разбору или разложению — они воспринимаются анализатором как единые целые.

Противоположностью терминалов являются нетерминальные символы, которые можно разложить на терминалы и другие нетерминалы, согласно правилам грамматики.

**3) Определение функции First1(…):**

Функция First1(X) для символа X (либо терминала, либо нетерминала) — это множество терминалов, которые могут быть первыми символами в любой строке, порожденной символом X. Если X — терминал, то First1(X) равно самому терминалу X. Если X — нетерминал, то First1(X) — это множество всех терминалов, которые могут быть первыми символами строк, порождаемых этим нетерминалом.

**Пример**: Для правила A→aB∣bCA \rightarrow aB \mid bCA→aB∣bC, First1(A) будет {a,b}\{a, b\}{a,b}.

**4) Как строится анализатор методом рекурсивного спуска**:

Анализатор методом рекурсивного спуска представляет собой набор рекурсивных функций, каждая из которых соответствует одному правилу грамматики. Эти функции вызывают друг друга для анализа соответствующих частей входной строки, рекурсивно обрабатывая каждый нетерминал. Для каждого нетерминала функция проверяет текущее входное выражение и пытается сопоставить его с одной из альтернатив грамматических правил. Если сопоставление успешно, анализ продолжается, в противном случае возвращается ошибка.

**5)** **Элементы, используемые в расширенной бэкусовой нормальной форме (EBNF)**:

Расширенная бэкусовая нормальная форма (EBNF) — это расширение обычной Бэкусовой нормальной формы (BNF), позволяющее более компактно записывать правила грамматики. В EBNF используются следующие элементы:

* Квадратные скобки [ ] — для обозначения необязательных элементов.
* Фигурные скобки { } — для обозначения повторяющихся элементов (ноль или более раз).
* Альтернатива ∣ — для обозначения вариантов.
* Символы группировки ( ) — для группирования выражений.