Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 3

Дисциплина: Конструирование компиляторов

Студент: Платонова Ольга

Вариант: 3

Группа: ИУ7-22М

Преподаватель: Ступников А. А.

Москва, 2023 г.

Цель работы: приобретение практических навыков реализации метода рекурсивного спуска для синтаксического разбора.

Задачи работы:

1) Принять к сведению соглашения об обозначениях, принятые в литературе по теории формальных языков и грамматик и кратко описанные в приложении.
2) Познакомиться с основными понятиями и определениями теории формальных языков и грамматик.
3) Детально разобраться в алгоритме метода рекурсивного спуска.
4) Разработать, тестировать и отладить программу нисходящего синтаксического анализа с использованием метода рекурсивного спуска в соответствии с предложенным вариантом.

Теоретическая часть

Одним из наиболее простых и потому одним из наиболее популярных методов нисходящего синтаксического анализа является метод рекурсивного спуска (*recursive descent method*). Метод основан на «зашивании» правил грамматики непосредственно в управляющие конструкции распознавателя.

В методе рекурсивного спуска полностью сохраняются идеи нисходящего разбора, принятые в LL(1)-грамматиках:

- происходит последовательный просмотр входной строки слева-направо;
- очередной символ входной строки является основанием для выбора одной из правых частей правил группы при замене текущего нетерминала;
- терминальные символы входной строки и правой части правила «взаимно уничтожаются»;
- обнаружение нетерминала в правой части рекурсивно повторяет этот же процесс.

Вариант 3. Грамматика G3

Рассматривается грамматика выражений отношения с правилами

<выражение> ->
    <арифметическое выражение> <знак операции отношения> <арифметическое выражение>

<арифметическое выражение> ->
    <терм> |
    <знак операции типа сложения> <терм> |
    <арифметическое выражение> <знак операции типа сложения> <терм>

<терм> ->
    <множитель> |
    <терм> <знак операции типа умножения> <множитель>

<множитель> ->
    <первичное выражение> |
    <множитель> ^ <первичное выражение>

<первичное выражение> ->
    <число> |
    <идентификатор> |
    (<арифметическое выражение>)

<знак операции типа сложения> ->
    + | -

<знак операции типа умножения> ->
    * | / | %

<знак операции отношения> ->
    < | <= | = | >= | > | <>

Вариант в стиле Си.

<программа> ->
    <блок>

<блок> ->
    { <список операторов> }

<список операторов>
    <оператор> <хвост>

<хвост> ->
    ; <оператор> <хвост> | ε

Первый вариант содержит левую рекурсию, которая должна быть устранена. Второй вариант не содержит левую рекурсию, но имеет ε-правило. В обоих вариантах точка с запятой (;) ставится между операторами. Теперь начальным символом грамматики становится нетерминал <программа>. Оба варианта содержат цепное правило <программа> -> <блок>. Можно начальным символом грамматики назначить нетерминал <блок>. А можно <блок> считать оператором, т. е.

<оператор> ->
    <идентификатор> = <выражение> |
    <блок>

Результаты работы

```
{
    a = 1;
    b = a + 3;
    c = 6 - b;
    d = 3 <> c;
    e = d / c;
}
```

Выражение валидно