|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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**

|  |  |
| --- | --- |
| **Тема Синтаксический разбор с использованием метода рекурсивного спуска**  **Вариант №6 (1)**  **Студент \_Иванов В.А. \_\_\_\_\_\_\_\_**  **Группа \_ИУ7-21М\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель \_Ступников А.А.** |  |

Москва.

2023 г.

**Цель работы**: приобретение навыков синтаксического разбора исходного кода программы на примере метода рекуррентного спуска

**Задачи работы:**

1. Изучить метод.
2. Преобразовать исходную грамматику к LL(1) путём ранее изученных методов.
3. Дополнить грамматику до Си-подобного синтаксиса
4. Разработать и протестировать программу рекуррентного спуска по полученной грамматике.

**Вариант 1. Грамматика G1.**

Грамматика G1 определяется правилами:

<выражение>

<простое выражение> |  
 <простое выражение> <операция отношения> <простое выражение>

<простое выражение>

<терм> |

<знак> <терм> |

<простое выражение> <операция типа сложения> <терм>

<терм>

<фактор> |

<терм> <операция типа умножения> <фактор>

<фактор>

<идентификатор> |

<константа> |

(<простое выражение>) |

not <фактор>

<операция отношения> = | <> | < | <= | > | >=

<знак> + | -

<операция типа сложения> + | - | or

<операция типа умножения> \* | / | div | mod | and

Пусть

<идентификатор> i

<константа> C

Грамматика содержит левую рекурсию, после устранения она приобретает вид:

<выражение>

<простое выражение> |  
 <простое выражение> <операция отношения> <простое выражение>

<простое выражение>

<терм> <простое выражение’> |

<знак> <терм> <простое выражение’>

<простое выражение’>

<операция типа сложения> <терм> <простое выражение’> |

<терм> <фактор><терм’>

<терм’> <операция типа умножения> <фактор> <терм’> |

<фактор>

<идентификатор> |

<константа> |

(<простое выражение>) |

not <фактор>

<операция отношения> = | <> | < | <= | > | >=

<знак> + | -

<операция типа сложения> + | - | or

<операция типа умножения> \* | / | div | mod | and

<идентификатор> i

<константа> C

Вычисление FIRST:

FIRST(<выражение>) = {i, C, (, n, +, -}

FIRST(<простое выражение>) = {i, C, (, n, +, -}

FIRST(<простое выражение’>) = {+, -, o, }

FIRST(<терм>) = {i, C, (, n}

FIRST(<терм’>) = {\*, /, d, m, a, }

FIRST(<фактор>) = {i, C, (, n}

FIRST(<операция отношения>) = {=, <, >}

FIRST(<знак>) = {+, -}

FIRST(<операция типа сложения>) = {+, -, o}

FIRST(<операция типа умножения>) = {\*, /, d, m, a }

FIRST(<идентификатор>) = {i}

FIRST(<константа>) = {C}

Посмотрев на правые части правил, уверенно заявляем, что теперь G1 является LL(1) грамматикой – FIRST от правых частей правил с одинаковыми левыми частями не пересекаются. Можно двигаться дальше.

В качестве расширенной грамматики выбран вариант в стиле Си.

<программа> <блок>

<блок> {<список операторов>}

<список операторов> <оператор> <хвост>

<хвост> ;<оператор> <хвост> |

<оператор> <идентификатор> = <выражение>

Расширенная грамматика также является LL(1) грамматикой.

**Выводы**

В ходе лабораторной работы был изучен метод рекуррентного спуска. Получены практические навыки в преобразовании грамматики к нужной форме. Реализована и протестирована программа, реализующая метод.