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»
ИУ-7

Лабораторная работа №3 «Синтаксический разбор с использованием метода рекурсивного спуска»

Выполнил студент: Агеев Алексей Владимирович

Группа: ИУ7-22М

Проверил: Андрей Алексеевич Ступников

Оценка: _____ Дата: _____ Подпись: _____

2020 год.

Цель. Приобретение практических навыков реализации важнейших элементов лексических анализаторов на примере распознавания цепочек регулярного языка.

Задачи

- Разобраться с методом синтаксическго разбора “Метод рекурсивного спуска”
- Реализовать “Метод рекусивного спуска” для граматики предложенное вариантом. Перед реализацией дополнить грамматику одним из видов блочного элемента. На выбор:
    - Блок в ситиле Паскаль
    - Блок в стиле С

Основная 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. Синтаксические анализаторы, работающие по методу рекурсивного спуска без возврата, могут быть построены для класса грамматик, называющегося LL(1). Первая буква L в названии связана с тем, что входная цепочка читается слева направо, вторая буква L означает, что строится левый вывод входной цепочки, 1
означает, что на каждом шаге для принятия решения используется один символ непрочитанной части входной цепочки. Для строгого определения LL(1) грамматики потребуются две функции - FIRST и FOLLOW.

Множество $FIRST_k(a)$ *:* Для КС-грамматики G это множество терминальных цепочек длиной k или меньше, с которых начинаются строки, пораждаемые цепочкой символов a . Если из цепочки a выводится ε -последовательность, то тогда множество $FIRST_k(a)$, так же содержит ε-цпочку.

$$FIRST_k(a) =$$
$$\{w|\ a \vdash_g^* w\beta, |w| = k\ or\ a \vdash_G^* w, |w| < k\} \cup \{\varepsilon | a \vdash_G^* \varepsilon\}$$

Множество $FOLLOW_k(\beta)$: Для КС-грамматикиG и некоторого $k > 0$ Множество $\mathrm{FOLLOW_k}(\beta)$, где $\beta \in (N \cup \Sigma)^*$ является множеством терминальных цепочек длины k или меньше, которые встречаются справа от β в синтенциальных формах.

$$FOLLOW_k(\beta) =$$
$$\{w|\ S \vdash_G^* \alpha\beta\gamma,\ w \in FIRST_k(\gamma)\} \cup \{\varepsilon | S \vdash_G^* \alpha\beta\}$$

$LL(k)$ *- грамматика*

КС-грамматика G является $LL(k)$ - грамматикой, если для любых двух левых выводов:

- $S \vdash_G^* xA\delta_1 \vdash_G^* xw_1\delta_1 \vdash_G^* xy_1$
- $S \vdash_G^* xA\delta_2 \vdash_G^* xw_2\delta_2 \vdash_G^* xy_2$

Если $FIRST_k(y_1) == FIRST_k(y_2)$, то $w_1 == w_2$

Задание (Варинат 2). Необходимо дополнить следующую грамматику блоком и реализовать для нее синтаксический разбор методом рекрсивного спуска.
Исходная грамматика:

```
<prog> := <block>
<block> := {<op_list>}
<op_list> := <op><tail>
<op> := <id>=<expr>
     | <block>
<tail> := ;<op><tail>
        | <epsilon>
<expr> := <ar_expr><relation><ar_exp>
       | <ar_expr>
<ar_expr> := <ar_exp><sum_op><term>
          | <term>
<term> := <term><mul_op><factor>
       |<factor>
<factor> := <id>|<const>|(<ar_expr>)
<relation> := <|<=|==|<>|>|>=
<sum_op> := +|-
<mul_op> := *|/
<id> := $id
<const> = $const
```

Преобразованная грамматика (добавлен блок в стиле C):

```
(OBRACKET, “{”) (CBRACKET, “}”) (ORBRACKET, “(”) (CRBRACKET, “)”)
(SEMICOLON, “;”) (RELATION, “<”), (RELATION, “<=”), (RELATION,
“>”),
(RELATION, “>=”), (RELATION, “<>”), (RELATION, “==”), (ASSIGN,
“=”), (ID, “$id”), (CONST, “$const”)

<prog> := <block>
<block> := ‘{’<op_list>’}’
<op_list> := <op><tail>
<op> := <id>’=’<expr>
     | ‘{’<op_list>’}’
<tail> := ‘;’<op><tail>
     | <epsilon>
<expr> := <ar_expr><expr`>
<expr`> := <relation><ar_expr>
        | <epsilon>

<factor> := <id>|<const>|’(’<ar_expr>’)’
<term> := <id><term`>
       | <const><term`>
       | ‘(’<ar_expr>’)’<term`>
<term`> := <mul_op><factor><term`>
       | <epsilon>
<ar_expr> := <id><term`><ar_expr`>
        | <const><term’><ar_expr’>
```

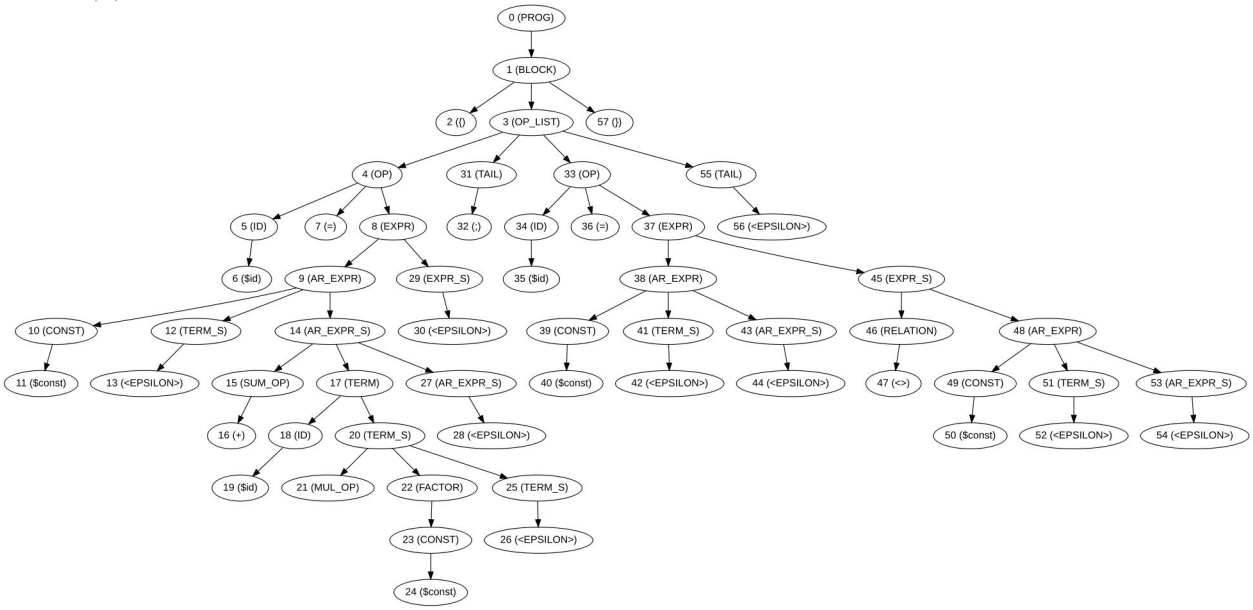
```
        | ‘(’<ar_expr>’)’ <term’><ar_expr’>
<ar_expr`> := <sum_op><term><ar_expr`>
        | <epsilon>
<relation> :=  ‘<’|’<=’|’>’|’>=’|’<>’
<mul_op> := ‘*’|’\’
<sum_op> := ‘+’|’-’
<id> := ‘$id’
<const> := ‘$const’
```

Демонстрация работы программы:

ВХОД:

```
{
    $id = $const + $id*$const;
    $id = $const <> $const
}
```

ВЫХОД:



ВХОД: (с ошибкой)

```
{
    $id = $const + $id*$const;
    $id = const <> $const
}
```

ВЫХОД:

Parsing Error: In line: 3; column: 8;
one of the following characters was expected: ID, CONST, ORBRACKET; But the actual type of symbol:{UNDEFINED, 'c'}