|  |  |
| --- | --- |
| **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»

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 :* Для КС-грамматики это множество терминальных цепочек длиной или меньше, с которых начинаются строки, пораждаемые цепочкой символов . Если из цепочки выводится -последовательность, то тогда множество , так же содержит -цпочку.

*Множество* Для КС-грамматики и некоторого Множество , где является множеством терминальных цепочек длины или меньше, которые встречаются справа от в синтенциальных формах.

*- грамматика*

КС-грамматика является - грамматикой, если для любых двух левых выводов:

-

-

Если , то

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

Исходная грамматика (добавлен блок в стиле С):

<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 (выполнено удаление левой рекурсии, а также приминена левая факторизация):

(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:*

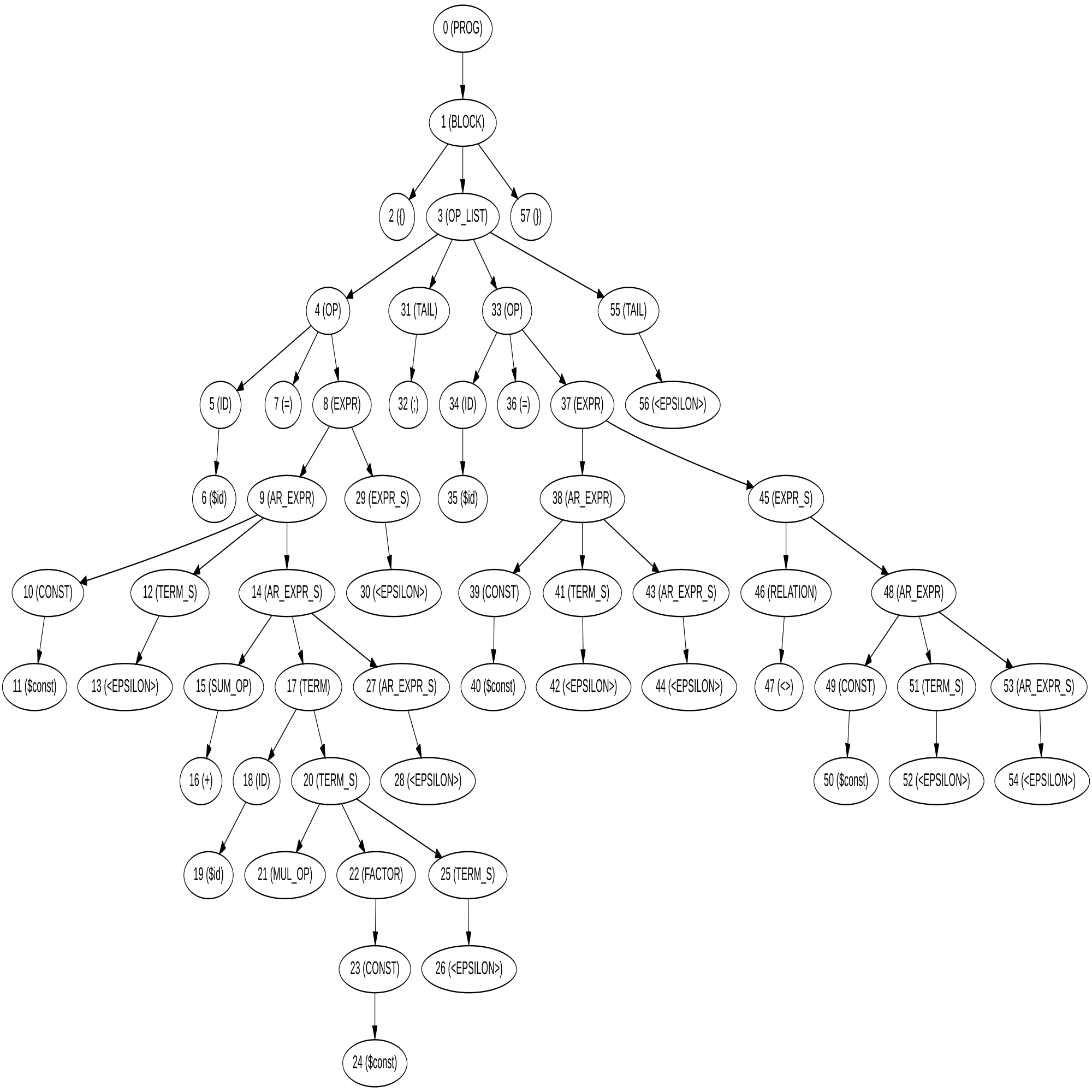
{

$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'}