МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Факультет автоматики и вычислительной техники

Кафедра вычислительной техники



**ОТЧЁТ**

**по лабораторной работе №5**

**«Синтаксис языков программирования. Восходящий синтаксический анализ, автоматная реализация»**

по дисциплине: «Теория формальных языков и компиляторов»

Вариант № 144211511, файл Lab5

Выполнил:Проверил:

студент гр. АВТ-918 Малявко А.А.

Ванин Константин «\_\_\_» \_\_\_\_\_\_\_\_\_ 2022 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка, подпись)

Новосибирск

2022

# Цель работы

Изучить основные идеи и понятия восходящего синтаксического анализа, свойства формальных грамматик, определяющих принадлежность грамматики к одному из классов **LR**, получить навыки построения автоматной реализации восходящего анализатора, исследовать поведение восходящих синтаксических акцепторов.

# Ход работы

## Реализация LL(1)-грамматики для учебного языка программирования:

Видимые/редактируемые:

program func +

func ident "=" ( type | "void" ) "(" ( ident ( "," type ident ) \* ) ? ")" "start" operator + "end"

type "int" | "real" | "char"

operator "put" expr "to" ident ";"

operator "if" "(" expr ")" "then" blockoroperator partnot

operator "exec" blockoroperator "with" ident "from" const "to" const ( "step" const ) ?

operator type ( "put" const "to" ) ? ident ( "," ( "put" const "to" ) ? ident ) \* ";"

operator "fin" ";"

operator "choice" expr ("option" ( minus ) ? const ":" blockoroperator) + ( "nooption" blockoroperator ) ? "end"

operator "return" expr ";"

expr ( minus ) ? begexpr ( endexpr ) ?

blockoroperator operator | ( "{" operator + "}" )

partnot "not" blockoroperator

partnot ~ "not"

begexpr ident ( "(" expr ")" ) ?

begexpr const

begexpr "(" expr ")"

const constdec | constreal | constexh | constchar

endexpr sign expr

sign oper | minus

ident [a-z] [0-9] {1,4} [a-zA-Z] +

minus [-]

constdec [0-9] +

constreal [0-9] + [.] [0-9] \*

constexh [0-9] + ( [.] [0-9] \* ) ? [e] [-] ? [0-9] +

constchar ["] [] ["]

oper ([-+/\*])|([!=][=])|([<>][=]?)

space [ \n\r\t] + {ignoreLastWord=true;}

comment [%][]\*[\n\r]{ignoreLastWord=true;}

keyword [a-z]+

switchword [:]

## Описание учебного языка

1. Идентификаторы: <б><пЦ><пБ> - одна маленькая буква; последовательность цифр длины от 1 до 4, непустая последовательность букв. Примеры: a1928A, b0CD;

2. Константы:

целые по основанию 10 - представляет собой число любой длины, состоящее из цифр 0-9. Примеры: 0, 25;

вещественные - представляет собой число любой длины, состоящее из цифр 0-9, целая и дробная части которой разделены точкой. Примеры: 0., 0.125;

экспоненциальные - представляет собой число любой длины, состоящее из цифр 0-9, экспоненциальная часть которой начинается с буквы e. Отрицательный показатель начинается со знака минус. Примеры: 10e2, 10e-2, 0.1e-2;

символьные – представляет собой любой символ, заключенный в кавычки. Пример: “c”;

3. Знаки операций: -, +, /, \*;

4. Знаки сравнения: !=, ==, >, >=, <, <=;

5. Разделителем является знак «;»;

6. Комментарии начинаются с символа % и могут быть только однострочными;

7. Оператор присваивания задаётся следующим образом: **put** <В> **to**<И>**;**, где < В> – произвольное выражение, <И> – идентификатор, жирным шрифтом помечены ключевые слова;

8. Условный присваивания задаётся следующим образом: **if (**<ЛВ> **)** **then**<ОБ>**[ not** <ОБ> **]** где <ЛВ> – логическое выражение, <ОБ>**–** оператор или блок операторов. В квадратных скобках выделена часть конструкции, которая не является обязательной;

9. Оператор цикла задаётся следующим образом: **exec**<ОБ>**with**<И> **from** <К> **to** <К>**[ step**<K> **]** где <К> – константа;

10. Оператор переключателя задаётся следующим образом: **choice**<В>**option** <К> **:** <ОБ> **[ fin; ]** **[ option … ] … [ nooption** <ОБ> **]** **end**, где троеточие следует за конструкцией, которое можно повториться неограниченное число раз;

11. Объявление функций происходит следующим образом:  **[**<Тип>**] ( [**<АргЛист>**] )**<Блок> где <Тип>**–** ключевое слово типа, <АргЛист> – список формальных аргументов функции, <Блок> – блок операторов.

**Фрагменты историй работы** восходящего синтаксического акцептора

Приведём фрагменты истории работы для конечного автомата со стековой памятью и несколькими состояниями, для конечного автомата со стековой памятью и одним состоянием и для программной реализации нисходящего рекурсивного спуска.

В качестве правильного тестового примера возьмём следующий код:

a1A = void() start put 0 to a1B; end

В качестве ошибочного:

a1A = void(put 0 to a1C) start put 0 to a1B; end

Истории работы представлены в таблицах 1, 2.

Таблица 1 – История работы автоматной реализации восходящего синтаксического акцептора для правильного тестового примера

|  |  |  |
| --- | --- | --- |
| 0 | a1A | 0 |
| 1 | = | 3 | 0 |
| 2 | void | 6 | 3 | 0 |
| 3 | ( | 9 | 6 | 3 | 0 |
| 4 | ( | 6 | 3 | 0 |  |
| 5 | ( | 8 | 6 | 3 | 0 |
| 6 | ) | 11 | 8 | 6 | 3 | 0 |
| 7 | ) | 11 | 8 | 6 | 3 | 0 |
| 8 | ) | 12 | 11 | 8 | 6 | 3 | 0 |
| 9 | start | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 10 | put | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 11 | 0 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 12 | 0 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 13 | 0 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 14 | to | 56 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 15 | to | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |
| 16 | to | 54 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 17 | to | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |
| 18 | to | 52 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 19 | to | 52 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 20 | to | 80 | 52 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 21 | to | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |
| 22 | to | 30 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |
| 23 | a1B | 51 | 30 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |
| 24 | ; | 79 | 51 | 30 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 25 | end | 110 | 79 | 51 | 30 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 26 | end | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |  |  |
| 27 | end | 19 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |  |
| 28 | end | 19 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |  |
| 29 | end | 28 | 19 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |
| 30 |  | 49 | 28 | 19 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |
| 31 |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| 32 |  | 2 | 0 |  |  |  |  |  |  |  |  |  |  |  |
| 33 |  | 2 | 0 |  |  |  |  |  |  |  |  |  |  |  |
| 34 |  | 4 | 2 | 0 |  |  |  |  |  |  |  |  |  |  |
| 35 |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| 36 |  | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |

Таблица 2 – История работы автоматной реализации восходящего синтаксического акцептора для ошибочного тестового примера

|  |  |  |
| --- | --- | --- |
| 0 | a1A | 0 |
| 1 | = | 3 | 0 |
| 2 | void | 6 | 3 | 0 |
| 3 | ( | 9 | 6 | 3 | 0 |
| 4 | ( | 6 | 3 | 0 |  |
| 5 | ( | 8 | 6 | 3 | 0 |
| 6 | put | 11 | 8 | 6 | 3 | 0 |

Таблица 3 – История работы процедурной реализации восходящего синтаксического акцептора для правильного тестового примера

|  |  |  |
| --- | --- | --- |
| 0 | a1A | 0 |
| 1 | = | 3 | 0 |
| 2 | void | 6 | 3 | 0 |
| 3 | ( | 9 | 6 | 3 | 0 |
| 4 | ( | 6 | 3 | 0 |  |
| 5 | ( | 8 | 6 | 3 | 0 |
| 6 | ) | 11 | 8 | 6 | 3 | 0 |
| 7 | ) | 11 | 8 | 6 | 3 | 0 |
| 8 | ) | 12 | 11 | 8 | 6 | 3 | 0 |
| 9 | start | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 10 | put | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 11 | 0 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 12 | 0 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 13 | 0 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 14 | to | 56 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 15 | to | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |
| 16 | to | 54 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 17 | to | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |
| 18 | to | 52 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 19 | to | 52 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 20 | to | 80 | 52 | 31 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 21 | to | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |
| 22 | to | 30 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |
| 23 | a1B | 51 | 30 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |
| 24 | ; | 79 | 51 | 30 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 25 | end | 110 | 79 | 51 | 30 | 20 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |
| 26 | end | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |  |  |
| 27 | end | 19 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |  |
| 28 | end | 19 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |  |
| 29 | end | 28 | 19 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |  |
| 30 |  | 49 | 28 | 19 | 17 | 14 | 12 | 11 | 8 | 6 | 3 | 0 |  |  |
| 31 |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| 32 |  | 2 | 0 |  |  |  |  |  |  |  |  |  |  |  |
| 33 |  | 2 | 0 |  |  |  |  |  |  |  |  |  |  |  |
| 34 |  | 4 | 2 | 0 |  |  |  |  |  |  |  |  |  |  |
| 35 |  | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| 36 |  | 1 | 0 |  |  |  |  |  |  |  |  |  |  |  |

Таблица 4 – История работы процедурной реализации восходящего синтаксического акцептора для ошибочного тестового примера

|  |  |  |
| --- | --- | --- |
| 0 | a1A | 0 |
| 1 | = | 3 | 0 |
| 2 | void | 6 | 3 | 0 |
| 3 | ( | 9 | 6 | 3 | 0 |
| 4 | ( | 6 | 3 | 0 |  |
| 5 | ( | 8 | 6 | 3 | 0 |
| 6 | put | 11 | 8 | 6 | 3 | 0 |

# Выводы

В данной лабораторной работе были изучены основные идеи и понятия восходящего синтаксического анализа, свойства формальных грамматик, определяющих принадлежность грамматики к одному из классов **LR**, получены навыки построения автоматной реализации восходящего анализатора, исследованы поведение восходящих синтаксических акцепторов.

.