Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное учреждение высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Кафедра «Информационных технологий и систем»

**«Проектирование синтаксического анализатора»**

Лабораторная работа

по дисциплине:

«Теория языков программирования и методы трансляции»

Проверил:

\_\_\_\_\_\_\_\_ / Макаров В. А.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_ 2020 г.

Выполнил

студент группы 8091:

\_\_\_\_\_\_/ Лехновский А. Д.

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2020 г.

**Великий Новгород**

**2020**

**Постановка задачи**

**Формулировка**

Разработать синтаксический анализатор. Синтаксический блок компилятора необходимо объединить с лексическим блоком. На входе - программа на определенном заданием языке, разработанная в соответствии с правилами грамматики. На выходе - сообщение о том, что текст соответствует грамматике (программа без ошибок), либо сообщение об ошибке.

**Грамматика**

1. STAT -> ITERATION

2) STAT -> ASSIGNMENT\_EXP ;

3) ITERATION -> while ( EXP ) STAT

4) EXP -> ASSIGNMENT\_EXP

5) EXP -> EXP , ASSIGNMENT\_EXP

6) ASSIGNMENT\_EXP -> id = ADDITIVE\_EXP

7) ADDITIVE\_EXP -> MULT\_EXP

8) ADDITIVE\_EXP -> ADDITIVE\_EXP additive\_operator MULT\_EXP

9) MULT\_EXP -> CAST\_EXP

10) MULT\_EXP -> MULT\_EXP mult\_operator CAST\_EXP

11) CAST\_EXP -> type\_name CAST\_EXP

12) CAST\_EXP -> UNARY\_EXP

13) UNARY\_EXP -> id

14) UNARY\_EXP -> int

15) UNARY\_EXP -> float

16) UNARY\_EXP -> prefix\_operator UNARY\_EXP

**Пояснение к данной грамматике**

prefix\_operator = { ++ -- }

int = целые числа

float = вещественные числа

id = идентификатор

type\_name = { int double long short }

mult\_operator = { \* / %}

additive\_operator = { + - }

**Описание реализации**

Для реализации синтаксического анализатора был выбран метод “перенос-опознание”. Данный метод распознаёт LR грамматики, не содержащие аннулирующего правила <A> -> ε.

Для выбранной грамматики найдем множества первых символов:



Для данной грамматики найдем множества следующих символов:



Используя полученные результаты, а также принцип ПЕРЕНОСА и принцип СВЕРТЫВАНИЯ построим управляющую таблицу, в которую будут входить данные из таблицы свертывания и таблицы переноса. Надо отметить, что в управляющей таблице будет вызов процедуры ОПОЗНАТЬ, которая далее будет подробно описана.

**Управляющая таблица**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ; | while | ( | ) | , | id | = | additive\_  operator | mult\_  operator | type\_  name | int | float | prefix\_  operator | -| |
| STAT |  |  |  |  |  |  |  |  |  |  |  |  |  | **О(1)** |
| ITERATION |  |  |  |  |  |  |  |  |  |  |  |  |  | **О(2)** |
| ASSIGNMENT\_EXP | **П** |  |  | **О**  **(3)** | **О**  **(3)** |  |  |  |  |  |  |  |  |  |
| EXP |  |  |  | **П** | **П** |  |  |  |  |  |  |  |  |  |
| ADDITIVE\_EXP | **О (4)** |  |  | **О (4)** | **О (4)** |  |  | **П** |  |  |  |  |  |  |
| MULT\_EXP | **О (5)** |  |  | **О (5)** | **О (5)** |  |  | **О (5)** | **П** |  |  |  |  |  |
| CAST\_EXP | **О (6)** |  |  | **О (6)** | **О (6)** |  |  | **О (6)** | **О (6)** |  |  |  |  |  |
| UNARY\_EXP | **О (7)** |  |  | **О (7)** | **О (7)** |  |  | **О (7)** | **О (7)** |  |  |  |  |  |
| ; |  |  |  |  |  |  |  |  |  |  |  |  |  | **О (8)** |
| while |  |  | **П** |  |  |  |  |  |  |  |  |  |  |  |
| ( |  |  |  |  |  | **П** |  |  |  |  |  |  |  |  |
| ) |  | **П** |  |  |  | **П** |  |  |  |  |  |  |  |  |
| , |  |  |  |  |  | **П** |  |  |  |  |  |  |  |  |
| id | **О (9)** |  |  | **О (9)** | **О (9)** |  | **П** | **О (9)** | **О (9)** |  |  |  |  |  |
| = |  |  |  |  |  | **П** |  |  |  | **П** | **П** | **П** | **П** |  |
| additive\_operator |  |  |  |  |  | **П** |  |  |  | **П** | **П** | **П** | **П** |  |
| mult\_operator |  |  |  |  |  | **П** |  |  |  | **П** | **П** | **П** | **П** |  |
| type\_name |  |  |  |  |  | **П** |  |  |  | **П** | **П** | **П** | **П** |  |
| int | **О (10)** |  |  | **О (10)** | **О (10)** |  |  | **О (10)** | **О (10)** |  |  |  |  |  |
| float | **О (11)** |  |  | **О (11)** | **О (11)** |  |  | **О (11)** | **О (11)** |  |  |  |  |  |
| prefix\_operator |  |  |  |  |  | **П** |  |  |  |  | **П** | **П** | **П** |  |
| ∇ |  | **П** |  |  |  | **П** |  |  |  |  |  |  |  |  |

**П – перенос**

**О – опознать**

**Описание процедуры ОПОЗНАТЬ**

**Опознать (1):**

If вершина = ∇ STAT

Then допустить

Else вершина = while ( EXP ) STAT

Then свертка (3)

Else отвергнуть

**Опознать (2):**

If вершина = ITERATION

Then свертка (1)

Else отвергнуть

**Опознать (3):**

If вершина = EXP , ASSIGNMENT\_EXP

Then свертка (5)

Else if вершина = ASSIGNMENT\_EXP

Then свертка (4)

**Опознать (4):**

If вершина = id = ADDITIVE\_EXP

Then свертка (6)

Else отвергнуть

**Опознать (5):**

If вершина = ADDITIVE\_EXP additive\_operator MULT\_EXP

Then свертка (8)

Else if вершина = MULT\_EXP

Then свертка (7)

**Опознать (6):**

If вершина = MULT\_EXP mult\_operator CAST\_EXP

Then свертка (10)

Else if вершина = type\_name CAST\_EXP

Then свертка (11)

Else if вершина = CAST\_EXP

Then свертка (9)

**Опознать (7):**

If вершина = prefix\_operator UNARY\_EXP

Then свертка (16)

Else if вершина = UNARY\_EXP

Then свертка (12)

**Опознать (8):**

If вершина = ASSIGNMENT\_EXP ;

Then свертка (2)

Else отвергнуть

**Опознать (9):**

If вершина = id

Then свертка (13)

Else отвергнуть

**Опознать (10):**

If вершина = int

Then свертка (14)

Else отвергнуть

**Опознать (11):**

If вершина = float

Then свертка (15)

Else отвергнуть

**Описание процедуры ПЕРЕНОС**

Перенос: втолкнуть (текущий входной символ), сдвиг.

**Описание процедуры СВЕРТКА**

**Свертка (1):** вытолкнуть, втолкнуть (STAT)

**Свертка (2):** вытолкнуть, вытолкнуть, втолкнуть (STAT)

**Свертка (3):** вытолкнуть, вытолкнуть, вытолкнуть, вытолкнуть, вытолкнуть, втолкнуть (ITERATION)

**Свертка (4):** вытолкнуть, втолкнуть (EXP)

**Свертка (5):** вытолкнуть, вытолкнуть, вытолкнуть, втолкнуть (EXP)

**Свертка (6):** вытолкнуть, вытолкнуть, вытолкнуть, втолкнуть (ASSIGNMENT\_EXP)

**Свертка (7):** вытолкнуть, втолкнуть (ADDITIVE\_EXP)

**Свертка (8):** вытолкнуть, вытолкнуть, вытолкнуть, втолкнуть (ADDITIVE\_EXP)

**Свертка (9):** вытолкнуть, втолкнуть (MULT\_EXP)

**Свертка (10):** вытолкнуть, вытолкнуть, вытолкнуть, втолкнуть (MULT\_EXP)

**Свертка (11):** вытолкнуть, вытолкнуть, втолкнуть (CAST\_EXP)

**Свертка (12):** вытолкнуть, втолкнуть (CAST\_EXP)

**Свертка (13):** вытолкнуть, втолкнуть (UNARY\_EXP)

**Свертка (14):** вытолкнуть, втолкнуть (UNARY\_EXP)

**Свертка (15):** вытолкнуть, втолкнуть (UNARY\_EXP)

**Свертка (16):** вытолкнуть, вытолкнуть, втолкнуть (UNARY\_EXP)

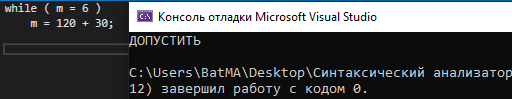
**Методика тестирования**

При тестировании используется файл *input.txt*. Программа выводит на экран ДОПУСТИТЬ, если входная цепочка соответствует правилам данной грамматики, в противном случае – ОТВЕГНУТЬ.

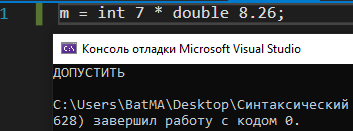
*Скриншоты работы программы.*

***Примеры с корректной входной цепочкой:***

*Пример №1*

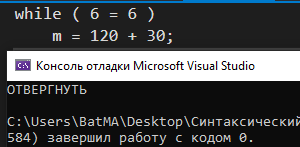
**

*Пример №2*

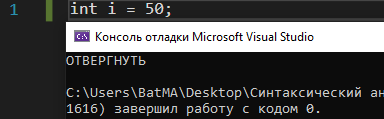
**

***Примеры с некорректной входной цепочкой:***

*Пример №1*

**

*Пример №2*

******

**Вывод**

В ходе лабораторной работы на основе рассматриваемой грамматики

был разработан синтаксический анализатор.