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 курсовой работе**

по дисциплине: «Теория формальных языков и компиляторов»

Вариант № 144211511, файл КР

Выполнил:Проверил:

студент гр. АВТ-918 Малявко А.А.

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

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

(оценка, подпись)

Новосибирск

2022

**Оглавление**

[Введение 3](#_Toc115724167)

[Описание учебного языка 3](#_Toc115724168)

[Тестовая программа 3](#_Toc115724169)

[Лексика учебного языка 4](#_Toc115724170)

[Формальная грамматика учебного языка 5](#_Toc115724171)

[Заключение 8](#_Toc115724172)

# Введение

В данной работе представлены результаты разработки основных элементов транслятора для заданного учебного языка на базе учебного пакета программы автоматизации разработки трансляторов ВебТрансБилдер.

# 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.

## Тестовая программа

Тестовая программа на учебном языке программирования согласно варианту задания:

a1A = void() start

put 0 to a1A;

if (a1A == 0)

then put 1 to a1A;

not put 2 \* 1 to a1A;

exec put 0.3145 to a1B; with a1i from 0 to 10 step 2

choice a1A

option 1 : {put 3 to a1A; fin;}

option 2 : put 4 to a1A;

nooption put 0.1e-10 to a1C;

end

end

## Лексика учебного языка

Ниже представлена система регулярных выражений, определяющая лексики заданного учебного языка.

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 [:]

Управляющая таблица конечного автомата, построенного по разработанной системе правил программой ВебТрансБилдер и легенда к ней представлены в таблицах 1 и 2 соответственно.

Таблица 1 – Управляющая таблица сканера

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| [ EOF ] | **-1** | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ () , ; { } ] | **-2** | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ = ] | 1 | **-6** | **-4** | 3 | **-5** | **-6** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ : ] | **-3** | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ a-d f-z ] | 2 | **-2** | 9 | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | 9 | 15 | ***E*** | ***E*** | **-10** | 15 | 15 | ***E*** | **-13** | 15 | 15 |
| [ % ] | 3 | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [  \d09  \d32 ] | 4 | **-2** | **-4** | 3 | 4 | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ \*+ / ] | **-6** | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ ! ] | 5 | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ < > ] | 6 | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ - ] | **-7** | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | 16 | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ " ] | 7 | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | **-12** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ 0-9 ] | 8 | **-2** | 10 | 3 | **-5** | ***E*** | **-6** | 11 | 8 | **-4** | 14 | ***E*** | 17 | 13 | 18 | **-11** | 17 | 17 | 19 | ***E*** |
| [  \d13  \d10 ] | 4 | **-2** | **-4** | **-9** | 4 | ***E*** | **-6** | 11 | **-8** | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ e ] | 2 | **-2** | 9 | 3 | **-5** | ***E*** | **-6** | 11 | 12 | 9 | 15 | ***E*** | ***E*** | 12 | 15 | 15 | ***E*** | **-13** | 15 | 15 |
| [ . ] | ***E*** | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | 13 | **-4** | ***E*** | ***E*** | ***E*** | **-10** | ***E*** | **-11** | ***E*** | **-13** | ***E*** | ***E*** |
| [ A-Z ] | ***E*** | **-2** | **-4** | 3 | **-5** | ***E*** | **-6** | 11 | **-8** | **-4** | 15 | ***E*** | ***E*** | **-10** | 15 | 15 | ***E*** | **-13** | 15 | 15 |

Таблица 2 – Финальные состояния и действия

|  |
| --- |
| Финальные состояния и действия: |
| |  |  |  |  |  | | --- | --- | --- | --- | --- | | **N** | **Токен** | **Группа слов** |  | **Действие** | | **-1** | -> 0 | **EOF** |  | *Lexem.groupIndex=0;* | | **-2** |  | **Word!** |  |  | | **-3** |  | **switchword** |  |  | | **-4** |  | **keyword** |  |  | | **-5** |  | **space** |  | {ignoreLastWord=true;} | | **-6** |  | **oper** |  |  | | **-7** |  | **minus** |  |  | | **-8** |  | **constdec** |  |  | | **-9** |  | **comment** |  | {ignoreLastWord=true;} | | **-10** |  | **constreal** |  |  | | **-11** |  | **ident** |  |  | | **-12** |  | **constchar** |  |  | | **-13** |  | **constexh** |  |  | |

Кратко алгоритм функционирования данного автомата можно описать следующим образом: каждому символу (или группе символов), поступающих на вход КА, в таблице переходов соответствует строка, каждому состоянию КА – столбец. В ячейке на пересечении строки и столбца содержится состояние, в которое перейдёт автомат при считывании символа, соответствующего данной строке, и находящийся изначально в состоянии, соответствующему данному столбцу. Таким образом, при обнаружении нового символа автоматы производят соответствие между символом и состоянием, в котором автомат находится. Следующее состояние определяется из результата соответствия (ячейки пересечений). Считывание очередного слова заканчивается отрицательным (конечным) состоянием автомата, при этом начальное состояние (при считывании очередного слова) – 0. В конце входной цепочки расположен индикатор EOF.

Запустим первый тестовый пример и приведём результат работы лексического анализатора:

Прочитанные лексемы: Прочитанные лексемы: [1,a1A] [13,=] [32,void] [8,(] [9,)] [28,start] [25,put] [2,0] [31,to] [1,a1A] [12,;] [20,if] [8,(] [1,a1A] [7,==] [2,0] [9,)] [30,then] [25,put] [2,1] [31,to] [1,a1A] [12,;] [23,not] [25,put] [2,2] [7,\*] [2,1] [31,to] [1,a1A] [12,;] [17,exec] [25,put] [3,0.3145] [31,to] [1,a1B] [12,;] [33,with] [1,a1i] [19,from] [2,0] [31,to] [2,10] [29,step] [2,2] [15,choice] [1,a1A] [24,option] [2,1] [11,:] [34,{] [25,put] [2,3] [31,to] [1,a1A] [12,;] [18,fin] [12,;] [35,}] [24,option] [2,2] [11,:] [25,put] [2,4] [31,to] [1,a1A] [12,;] [22,nooption] [25,put] [4,0.1e-10] [31,to] [1,a1C] [12,;] [16,end] [16,end] [0,]

Можно заметить, что все лексемы, определённые в коде тестовой программы, лексическим анализатором были считаны верно – каждое слово присутствует в результате.

## Формальная грамматика учебного языка

Для учебного языка программирования была разработана грамматика класса LL(1). Были разработаны следующие правила:

program func +

func ident "=" ( type | "void" ) "(" (type 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

Для данной грамматики в программе ВебТрансБилдер был построен нисходящий синтаксический анализатор на основе стекового автомата с несколькими состояниями. Приведём фрагмент его управляющей таблицы (таблица 3).

Таблица 3. Фрагмент управляющей таблицы нисходящего парсера с несколькими состоянии

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| N сост. | flag\_A | flag\_S | flag\_R | flag\_E | Переход | Множество выбора | Действие |
| 0 program |  |  |  |  | 73 | ident |  |
| 1 program!0 |  |  |  |  | 75 | ident |  |
| 2 program!1 |  |  |  | 1 | 85 | ident |  |
| 3 program!1 |  |  |  |  | 87 | EOF |  |
| 4 func!0 |  |  |  | 1 | 88 | "char" "int" "real" |  |
| 5 func!0 |  |  |  |  | 89 | "void" |  |
| 6 func!2 |  |  |  | 1 | 90 | "char" "int" "real" |  |
| 7 func!2 |  |  |  |  | 93 | ")" |  |
| 8 func!3 |  |  |  | 1 | 94 | "put" |  |
| 9 func!3 |  |  |  | 1 | 99 | "if" |  |
| 10 func!3 |  |  |  | 1 | 106 | "exec" |  |
| 11 func!3 |  |  |  | 1 | 115 | "char" "int" "real" |  |
| 12 func!3 |  |  |  | 1 | 120 | "fin" |  |
| 13 func!3 |  |  |  | 1 | 122 | "choice" |  |
| 14 func!3 |  |  |  |  | 128 | "return" |  |
| 15 func!4 |  |  |  | 1 | 131 | "char" "choice" "exec" "fin" "if" "int" "put" "real" "return" |  |
| 16 func!4 |  |  |  |  | 133 | "end" |  |
| 17 type |  |  |  | 1 | 134 | "int" |  |
| 18 type |  |  |  | 1 | 135 | "real" |  |
| 19 type |  |  |  |  | 136 | "char" |  |
| 20 func!1 |  |  |  | 1 | 137 | "," |  |
| 21 func!1 |  |  |  |  | 141 | ")" |  |
| 22 expr |  |  |  |  | 142 | minus ident constdec constreal constexh constchar "(" |  |

Принцип работы данного автомата следующий:

Автомат имеет Автомат имеет одномерную управляющую таблицу (представленную выше), текущее состояние, текущий входной символ и стек для хранения номеров состояний. Автомат стартует в состоянии 0, на входе в этот   
момент первый терминал из входной цепочки. В любой момент по текущему терминалу и содержимому полей текущего состояния определяется возможность и способ перехода в следующее состояние (а также номер этого состояния), переход к следующему терминалу, и, возможно, модифицируется содержимое стека. Останов автомата в специально выделенном состоянии свидетельствует о том, что входная цепочка является правильным предложением, в любом другом состоянии – что в цепочке ошибка. При установке флагов a, r, s и e должны быть выполнены следующие действия:

* a – перед переходом из этого состояния нужно прочитать следующий терминал из входной цепочки;
* s – перед переходом из этого состояния номер следующего (номер текущего +1) должен быть занесен в стек;
* r – адрес перехода должен быть снят с верхушки стека;
* e – подавить останов по ошибке.

Проведем тестирование работы данного автомата на тестовом примере. Приведем фрагмент истории разбора автоматной реализации нисходящего синтаксического акцептора представлен в таблице 4.

Таблица 4. История разбора автоматной реализации нисходящего синтаксического акцептора

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | $1func | 0 |  |  |  |  |
| 1 | $1func | 96 |  |  | s |  |
| 2 | $1func | 1 |  |  |  |  |
| 3 | $1func | 98 |  |  |  |  |
| 4 | $1func | 99 | a |  |  |  |
| 5 | := | 100 | a |  |  |  |
| 6 | => | 101 | a |  |  |  |
| 7 | as | 102 |  |  |  |  |
| 8 | as | 103 |  |  | s |  |
| 9 | as | 4 |  |  |  | e |
| 10 | as | 115 | a |  |  |  |
| 11 | int | 116 |  |  |  |  |
| 12 | int | 18 |  |  |  | e |
| 13 | int | 181 | a | r |  |  |
| 14 | ( | 104 | a |  |  |  |
| 15 | int | 105 |  |  | s |  |
| 16 | int | 6 |  |  |  | e |
| 17 | int | 118 |  |  |  |  |
| 18 | int | 119 |  |  | s |  |
| 19 | int | 18 |  |  |  | e |
| 20 | int | 181 | a | r |  |  |
| 21 | $10af | 120 |  |  |  |  |
| 22 | $10af | 121 | a |  |  |  |
| 23 | , | 122 |  |  |  |  |
| 24 | , | 22 |  |  |  | e |
| 25 | , | 185 | a |  |  |  |
| 26 | int | 186 |  |  |  |  |
| 27 | int | 187 |  |  | s |  |
| 28 | int | 18 |  |  |  | e |
| 29 | int | 181 | a | r |  |  |

# Заключение

В ходе выполнения курсовой работы были изучены теоретические основы проектирования трансляторов языков программирования, освоено средство автоматизации построения трансляторов – программа ВебТрансБилдер, а также разработаны лексический и синтаксический анализаторы. Была изучена работа данных компонентов и протестирована на примере нескольких программ на заданном учебном языке.

.