МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра программной инженерии

**Отчет по лабораторной работе**

по дисциплине «Теория языков программирования и методы трансляции»

на тему: «Реализация синтаксического анализатора методом рекурсивного спуска»

Студент Бажин М.И. Шифр:160582

Институт приборостроения, автоматизации и информационных технологий

Специальность 09.03.01 «Информатика и вычислительная техника»

Группа 61-ИВТ

Руководитель Гордиенко А.П.

Орел 2018

**ГРАММАТИКА**

GOAL → SECTION

SECTION → DECLARATION IMPLEMENTATION

DECLARATION → var\_term VAR\_DECL\_INSTR ; VAR\_DECLARATION

VAR\_DECLARATION → VAR\_DECL\_INSTR ; VAR\_DECLARATION

→ E

VAR\_DECL\_INSTR → id ID\_DECL : type\_term

ID\_DECL → , id ID\_DECL

→ E

IMPLEMENTATION → begin\_term LIST\_INSTRUCTION end\_term .

LIST\_INSTRUCTION → ASSIGNMENT\_INSTRUCTION ; LIST\_INSTRUCTION

→ READ\_INSTR ; LIST\_INSTRUCTION

→ WRITE\_INSTR ; LIST\_INSTRUCTION

→ BRANCH\_INSTR ; LIST\_INSTRUCTION

→ E

ASSIGNMENT\_INSTRUCTION → id := EXPR

EXPR → TERM EXPR1

EXPR1 → + TERM EXPR1

→ - TERM EXPR1

→ E

TERM → FACTOR TERM1

TERM1 → \* FACTOR TERM1

→ / FACTOR TERM1

→ div FACTOR TERM1

→ mod FACTOR TERM1

→ E

FACTOR → ( EXPR )

→ round\_term ( EXPR )

→ num

→ realnum

→ id

READ\_INSTR → read\_term ( id )

WRITE\_INSTR → write\_term ( id )

→ write\_term ( “ string\_term “ )

→ write\_term ( “ id “ )

BRANCH\_INSTR → if\_term CONDITION CONSEQUENCE

CONDITION → BOOLEXPR CONDITION1

→ EXPR BOOLOP EXPR

CONDITION1 → or\_term BOOLEXPR CONDITION1

→ E

BOOLEXPR → BOOLTERM BOOLEXPR1

BOOLEXPR1 → and\_term BOOLTERM BOOLEXPR1

→ E

BOOLTERM → ( CONDITION )

→ ( EXPR BOOLOP EXPR )

BOOLOP → =

→ <

→ >

→ boolop\_term

CONSEQUENCE → then\_term CONSEQUENCE1 ELSE

CONSEQUENCE1 → ASSIGNMENT\_INSTRUCTION

→ READ\_INSTR

→ WRITE\_INSTR

→ BRANCH\_INSTR

→ begin LIST\_INSTRUCTION end

ELSE → else\_term CONSEQUENCE1

→ E

Листинг программы:

function TForm1.decloration: boolean;

begin

if token = var\_term

then

begin

token:=yylex;

if var\_decl\_instr and (token = ord(';'))

then

begin

token:=yylex;

result := var\_decloration

end

else result := false;

end

else result := false;

end;

function TForm1.var\_decloration: boolean;

var res: boolean;

begin

case token of

id:

begin

if var\_decl\_instr and (token = ord(';'))

then

begin

token:=yylex;

result := var\_decloration();

end

else result := false;

end;

else result := true; //Эпсилон (var\_decloration -> E)

end;

end;

function TForm1.var\_decl\_instr: boolean;

begin

if token = id then

begin

token := yylex;

if id\_decl then

begin

if token = ord(':') then

begin

token:=yylex;

if token = type\_term then

begin

token := yylex;

result := true;

end

else result := false;

end

else result := false;

end

else result := false;

end

else result := false;

end;

function TForm1.id\_decl:boolean;

begin

if token = ord(',') then

begin

token := yylex;

if token = id then

begin

token := yylex;

result := id\_decl();

end

else result := false;

end

else result:=true; //Эпсилон (id\_decl -> E)

end;

function TForm1.assigment\_inctruction: boolean;

begin

if (yylex = ord(':')) and (yylex = ord('=')) //!!!! заменить : и = на оператор :=

then

begin

token:=yylex;

result:=expr();

end

else result:=false;

end;

function TForm1.expr: boolean;

begin

if term and expr1

then result:=true

else result:=false;

end;

function TForm1.expr1: boolean;

begin

case token of

ord('+'):begin

token:=yylex;

if term and expr1()

then result:=true

else result:=false;

end;

ord('-'):begin

token:=yylex;

if term and expr1()

then result:=true

else result:=false;

end;

else result:=true; //Эпсилон (expr1 -> E)

end;

end;

function TForm1.term: boolean;

begin

if factor and term1

then result:=true

else result:=false;

end;

function TForm1.term1: boolean;

begin

case token of

ord('\*'):begin

token:=yylex;

if factor and term1()

then result:=true

else result:=false;

end;

ord('/'):begin

token:=yylex;

if factor and term1()

then result:=true

else result:=false;

end;

div\_term:begin

token:=yylex;

if factor and term1()

then result:=true

else result:=false;

end;

mod\_term:begin

token:=yylex;

if factor and term1()

then result:=true

else result:=false;

end;

else result:=true; //Эпсилон (term1 -> E)

end;

end;

function TForm1.factor: boolean;

begin

case token of

num:begin

result:=true

end;

dec:begin

result:=true

end;

id:begin

result:=true

end;

round\_term:begin

if (yylex = ord('('))

then

begin

token:=yylex;

result:=expr and (token = ord(')'))

end

else result:=false;

end;

ord('('):begin

token:=yylex;

result := expr() and (token = ord(')'));

end;

else result:=false;

end;

token:=yylex;

end;

=====================================================================

function TForm1.branch\_inctruction: boolean;

begin

token:=yylex;

if condition and consequence

then result := true

else result := false;

end;

function TForm1.condition: boolean;

begin

if (token = ord('('))

then

begin

result := boolexpr and condition1;

end

else

begin

result := expr and boolop and expr;

end;

end;

function TForm1.condition1: boolean;

begin

if (token = or\_term) then

begin

token:=yylex;

if boolexpr and condition1

then result := true

else result := false

end

else result := true; //Эпсилон (condition1 -> E)

end;

function TForm1.boolexpr: boolean;

begin

if boolterm and boolexpr1

then result := true

else result := false

end;

function TForm1.boolexpr1: boolean;

begin

if (token = and\_term) then

begin

token:=yylex;

if boolterm and boolexpr1

then result := true

else result := false

end

else result := true; //Эпсилон (boolexpr1 -> E)

end;

function TForm1.boolterm: boolean;

begin

if (token = ord('('))

then

begin

token:=yylex;

if (token = ord('(')) then

begin

result := condition;

end

else result := expr and boolop and expr and (token = ord(')'));

token:=yylex;

end

else result:=false;

end;

function TForm1.boolop: boolean;

begin

case token of

ord('='):begin

result:=true

end;

ord('<'):begin

result:=true

end;

ord('>'):begin

result:=true

end;

boolop\_term:begin

result:=true

end;

else result:=false;

end;

token:=yylex;

end;

function TForm1.consequence: boolean;

begin

if token = then\_term then

begin

token:=yylex;

result:=consequence1 and elsecons;

end

else result:=false;

end;

function TForm1.consequence1: boolean;

begin

case token of

id:begin

result:=assigment\_inctruction;

end;

read\_term:begin

result:=read\_inctruction;

end;

write\_term:begin

result:=write\_inctruction;

end;

if\_term:begin

result:=branch\_inctruction;

end;

begin\_term:begin

if list\_inctruction and (token=end\_term)

then

begin

token:=yylex;

result:=true;

end

else result:=false;

end;

else result := false;

end;

end;

function TForm1.elsecons: boolean;

begin

if token = else\_term then

begin

token:=yylex;

result:=consequence1;

end

else result:=true; //Эпсилон (elsecons -> E)

end;