

Compiler Design Principles The PARSER

Dr. Mohammad Reza Razzazi

*Amirkabir University of Technology*

*SPRING 2017*



aLI gHOLAMI - 9531504

aiDA DOOST MOHAMMADI - 9431550

**Grammar (.y) File**

%{

package chronicle;

import java.io.\*;

%}

%token ID NUMCONST REALCONST CHARCONST BOOLCONST SHARP\_KW MOD\_KW DIV\_KW MUL\_KW SUB\_KW ADD\_KW SINGLE\_QUOTE\_KW DOT\_KW LTE\_KW GTE\_KW NEQ\_KW EQ\_KW GT\_KW LT\_KW COMMA\_KW CLOSEPARENTHESIS\_KW OPENPARENTHESIS\_KW CLOSEACCOLADE\_KW OPENACCOLADE\_KW CLOSEBRACKET\_KW OPENBRACKET\_KW ASSIGN\_KW COLON\_KW SEMICOLON\_KW NOT\_KW OR\_KW AND\_KW DOWNTO\_KW UPTO\_KW EXIT\_KW RETURN\_KW FOR\_KW WHEN\_KW END\_KW DEFAULT\_KW CASE\_KW SWITCH\_KW WHILE\_KW DO\_KW ELSE\_KW THEN\_KW IF\_KW PROCEDURE\_KW BOOLEAN\_KW CHARACTER\_KW REAL\_KW INTEGER\_KW MAIN\_KW PROGRAM\_KW DIGIT NONZERO\_DIGIT LETTER

%code {

static PrintStream writer;

public static void main(String args[]) throws IOException, FileNotFoundException {

YYParser yyparser;

final Yylex lexer;

writer = new PrintStream(new File("yacc\_tool\_output.txt"));

lexer = new Yylex(new InputStreamReader(new FileInputStream(".\\Global\_Test\\globalTest2.shl")));

yyparser = new YYParser(new Lexer() {

@Override

public int yylex() {

int yyl\_return = -1;

try {

yyl\_return = lexer.yylex();

} catch (IOException e) {

System.err.println("IO error :" + e);

}

return yyl\_return;

}

@Override

public void yyerror(String error) {

System.err.println("Error : " + error);

}

@Override

public Object getLVal() {

return null;

}

});

yyparser.parse();

return;

}

}

// Precedences go increasing, so "then" < "else".

%nonassoc THEN\_KW

%nonassoc ELSE\_KW

%nonassoc DOT\_KW

%%

program:

PROGRAM\_KW ID MAIN\_KW block {

System.out.println("Rule 1.2: " +

"program -> PROGRAM\_KW ID MAIN\_KW block");

}

| PROGRAM\_KW ID declarations\_list MAIN\_KW block {

System.out.println("Rule 1.3: " +

"program -> PROGRAM\_KW ID declarations\_list MAIN\_KW block");

}

| PROGRAM\_KW ID procedure\_list MAIN\_KW block {

System.out.println("Rule 1.4: " +

"program -> PROGRAM\_KW ID procedure\_list MAIN\_KW block");

}

| PROGRAM\_KW ID declarations\_list procedure\_list MAIN\_KW block {

System.out.println("Rule 1.5: " +

"program -> PROGRAM\_KW ID declarations\_list procedure\_list MAIN\_KW block");

}

| PROGRAM\_KW ID MAIN\_KW {

System.out.println("Rule 1.1: " +

"program -> PROGRAM\_KW ID MAIN\_KW");

}

declarations\_list:

declarations\_list declarations {

System.out.println("Rule 2.1: " +

"declarations\_list -> declarations\_list declarations");

}

| declarations {

System.out.println("Rule 2.2: " +

"declarations\_list -> declarations");

}

declarations:

type\_specifiers declarator\_list SEMICOLON\_KW {

System.out.println("Rule 3.1: " +

"declarations -> type\_specifiers declarator\_list SEMICOLON\_KW");

}

type\_specifiers:

INTEGER\_KW {

System.out.println("Rule 4.1: " +

"type\_specifiers -> INTEGER\_KW");

}

| REAL\_KW {

System.out.println("Rule 4.2: " +

"type\_specifiers -> REAL\_KW");

}

| CHARACTER\_KW {

System.out.println("Rule 4.3: " +

"type\_specifiers -> CHAR\_KW");

}

| BOOLEAN\_KW {

System.out.println("Rule 4.4: " +

"type\_specifiers -> BOOLEAN\_KW");

}

declarator\_list:

declarator {

System.out.println("Rule 5.1: " +

"declarator\_list -> declarator");

}

| declarator\_list COMMA\_KW declarator {

System.out.println("Rule 5.2: " +

"declarator\_list -> declarator\_list COMMA\_KW declarator");

}

declarator:

dec {

System.out.println("Rule 6.1: " +

"declarator -> dec");

}

| dec ASSIGN\_KW initializer {

System.out.println("Rule 6.2: " +

"declarator -> dec ASSIGN\_KW initializer");

}

dec:

ID {

System.out.println("Rule 7.1: " +

"dec -> ID");

}

| ID OPENBRACKET\_KW range CLOSEBRACKET\_KW {

System.out.println("Rule 7.2: " +

"dec -> ID OPENBRACKET\_KW range CLOSEBRACKET\_KW");

}

| ID OPENBRACKET\_KW NUMCONST CLOSEBRACKET\_KW {

System.out.println("Rule 7.3: " +

"dec -> ID OPENBRACKET\_KW NUMCONST CLOSEBRACKET\_KW");

}

range:

ID DOT\_KW ID {

System.out.println("Rule 8.1: " +

"range -> ID DOT\_KW ID");

}

| NUMCONST DOT\_KW NUMCONST {

System.out.println("Rule 8.2: " +

"range -> NUMCONST DOT\_KW NUMCONST");

}

| arithmetic\_expressions DOT\_KW arithmetic\_expressions {

System.out.println("Rule 8.3: " +

"range -> arithmetic\_expressions DOT\_KW arithmetic\_expressions");

}

initializer:

constant\_expressions {

System.out.println("Rule 9.1: " +

"initializer -> constant\_expressions");

}

| OPENACCOLADE\_KW initializer\_list CLOSEACCOLADE\_KW {

System.out.println("Rule 9.2: " +

"initializer -> OPENACCOLADE\_KW initializer CLOSEACCOLADE\_KW");

}

initializer\_list:

constant\_expressions COMMA\_KW initializer\_list {

System.out.println("Rule 10.1: " +

"initializer\_list -> constant\_expressions COMMA\_KW initializer\_list");

}

| constant\_expressions {

System.out.println("Rule 10.2: " +

"initializer\_list -> constant\_expressions");

}

procedure\_list:

procedure\_list procedure {

System.out.println("Rule 11.1: " +

"procedure\_list -> procedure\_list procedure");

}

| procedure {

System.out.println("Rule 11.2: " +

"procedure\_list -> procedure");

}

procedure:

PROCEDURE\_KW ID parameters OPENACCOLADE\_KW block CLOSEACCOLADE\_KW SEMICOLON\_KW {

System.out.println("Rule 12.1: " +

"procedure -> PROCEDURE\_KW ID parameters OPENACCOLADE\_KW block CLOSEACCOLADE\_KW SEMICOLON\_KW");

}

|PROCEDURE\_KW ID parameters OPENACCOLADE\_KW declarations\_list block CLOSEACCOLADE\_KW SEMICOLON\_KW {

System.out.println("Rule 12.2: " +

"procedure -> PROCEDURE\_KW ID parameters OPENACCOLADE\_KW declarations\_list block CLOSEACCOLADE\_KW SEMICOLON\_KW");

}

parameters:

OPENPARENTHESIS\_KW declarations\_list CLOSEPARENTHESIS\_KW {

System.out.println("Rule 13.1: " +

"parameters -> OPENPARENTHESIS\_KW declarations\_list CLOSEPARENTHESIS\_KW");

}

block:

OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW {

System.out.println("Rule 14.1: " +

"block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW");

}

statement\_list:

statement SEMICOLON\_KW {

System.out.println("Rule 15.1: " +

"statement\_list -> statement SEMICOLON\_KW");

}

| statement\_list statement SEMICOLON\_KW {

System.out.println("Rule 15.2: " +

"statement\_list -> statement\_list statement SEMICOLON\_KW");

}

| SEMICOLON\_KW {

System.out.println("Rule 15.3: " +

"statement\_list -> SEMICOLON\_KW");

}

| statement\_list SEMICOLON\_KW {

System.out.println("Rule 15.4: " +

"statement\_list -> statement\_list SEMICOLON\_KW");

}

statement:

ID ASSIGN\_KW expressions {

System.out.println("Rule 16.1: " +

"statement -> ID ASSIGN\_KW expressions");

}

| IF\_KW bool\_expressions THEN\_KW statement {

System.out.println("Rule 16.2: " +

"statement -> IF\_KW bool\_expressions THEN\_KW statement");

}

| IF\_KW bool\_expressions THEN\_KW statement ELSE\_KW statement {

System.out.println("Rule 16.3: " +

"statement -> IF\_KW bool\_expressions THEN\_KW statement ELSE\_KW statement");

}

| DO\_KW statement WHILE\_KW bool\_expressions {

System.out.println("Rule 16.4: " +

"statement -> DO\_KW statement WHILE\_KW bool\_expressions");

}

| FOR\_KW ID ASSIGN\_KW counter DO\_KW statement {

System.out.println("Rule 16.5: " +

"statement -> FOR\_KW ID ASSIGN\_KW counter DO\_KW statement");

}

| SWITCH\_KW expressions case\_element default END\_KW {

System.out.println("Rule 16.6: " +

"statement -> SWITCH\_KW expressions case\_element default END\_KW");

}

| ID OPENPARENTHESIS\_KW arguments\_list CLOSEPARENTHESIS\_KW {

System.out.println("Rule 16.7: " +

"statement -> ID OPENPARENTHESIS\_KW arguments\_list CLOSEPARENTHESIS\_KW");

}

| ID OPENBRACKET\_KW expressions CLOSEBRACKET\_KW ASSIGN\_KW expressions {

System.out.println("Rule 16.8: " +

"statement -> IDENTIFIER OPENBRACKET\_KW expressions CLOSEBRACKET\_KW ASSIGN\_KW expressions");

}

| RETURN\_KW expressions {

System.out.println("Rule 16.9: " +

"statement -> RETURN\_KW expressions");

}

| EXIT\_KW WHEN\_KW bool\_expressions {

System.out.println("Rule 16.10: " +

"statement -> EXIT\_KW WHEN\_KW bool\_expressions");

}

| block {

System.out.println("Rule 16.11: " +

"statement -> block");

}

| ID OPENPARENTHESIS\_KW CLOSEPARENTHESIS\_KW {

System.out.println("Rule 16.12: " +

"statement -> ID OPENPARENTHESIS\_KW CLOSEPARENTHESIS\_KW");

}

| SWITCH\_KW expressions case\_element END\_KW {

System.out.println("Rule 16.13: " +

"statement -> SWITCH\_KW expressions case\_element END\_KW");

}

arguments\_list:

multi\_arguments {

System.out.println("Rule 17.1: " +

"arguments\_list -> multi\_arguments");

}

multi\_arguments:

multi\_arguments COMMA\_KW expressions {

System.out.println("Rule 18.1: " +

"multi\_arguments -> multi\_arguments COMMA\_KW expressions");

}

| expressions {

System.out.println("Rule 18.2: " +

"multi\_arguments -> expressions");

}

counter:

NUMCONST UPTO\_KW NUMCONST {

System.out.println("Rule 19.1: " +

"counter -> NUMCONST UPTO\_KW NUMCONST");

}

| NUMCONST DOWNTO\_KW NUMCONST {

System.out.println("Rule 19.2: " +

"counter -> NUMCONST DOWNTO\_KW NUMCONST");

}

case\_element:

CASE\_KW NUMCONST COLON\_KW block {

System.out.println("Rule 20.1: " +

"case\_element -> CASE\_KW NUMCONST COLON\_KW block");

}

| case\_element CASE\_KW NUMCONST COLON\_KW block {

System.out.println("Rule 20.2: " +

"case\_element -> case\_element CASE\_KW NUMCONST COLON\_KW block");

}

default:

DEFAULT\_KW COLON\_KW block {

System.out.println("Rule 21.1: " +

"default -> DEFAULT\_KW COLON\_KW block");

}

expressions:

constant\_expressions {

System.out.println("Rule 22.1: " +

"expressions -> constant\_expressions");

}

| bool\_expressions {

System.out.println("Rule 22.2: " +

"expressions -> bool\_expressions");

}

| arithmetic\_expressions {

System.out.println("Rule 22.3: " +

"expressions -> arithmetic\_expressions");

}

| ID {

System.out.println("Rule 22.4: " +

"expressions -> ID");

}

| ID OPENBRACKET\_KW expressions CLOSEBRACKET\_KW {

System.out.println("Rule 22.5: " +

"expressions -> ID OPENBRACKET\_KW expressions CLOSEBRACKET\_KW");

}

| ID OPENPARENTHESIS\_KW arguments\_list CLOSEPARENTHESIS\_KW {

System.out.println("Rule 22.6: " +

"expressions -> ID OPENPARENTHESIS\_KW arguments\_list CLOSEPARENTHESIS\_KW");

}

| OPENPARENTHESIS\_KW expressions CLOSEPARENTHESIS\_KW {

System.out.println("Rule 22.7: " +

"expressions -> OPENPARENTHESIS\_KW expressions CLOSEPARENTHESIS\_KW");

}

| ID OPENPARENTHESIS\_KW CLOSEPARENTHESIS\_KW {

System.out.println("Rule 22.8: " +

"expressions -> ID OPENPARENTHESIS\_KW CLOSEPARENTHESIS\_KW");

}

constant\_expressions:

NUMCONST {

System.out.println("Rule 23.1: " +

"constant\_expressions -> NUMCONST");

}

| REALCONST {

System.out.println("Rule 23.2: " +

"constant\_expressions -> REALCONST");

}

| CHARCONST {

System.out.println("Rule 23.3: " +

"constant\_expressions -> CHARCONST");

}

| BOOLEAN\_KW {

System.out.println("Rule 23.4: " +

"constant\_expressions -> BOOLEAN\_KW");

}

bool\_expressions:

LT\_KW pair {

System.out.println("Rule 24.1: " +

"bool\_expressions -> LT\_KW pair");

}

| LTE\_KW pair {

System.out.println("Rule 24.2: " +

"bool\_expressions -> LTE\_KW pair");

}

| GT\_KW pair {

System.out.println("Rule 24.3: " +

"bool\_expressions -> GT\_KW pair");

}

| GTE\_KW pair {

System.out.println("Rule 24.4: " +

"bool\_expressions -> GTE\_KW pair");

}

| EQ\_KW pair {

System.out.println("Rule 24.5: " +

"bool\_expressions -> EQ\_KW pair");

}

| NEQ\_KW pair {

System.out.println("Rule 24.6: " +

"bool\_expressions -> NEQ\_KW pair");

}

| AND\_KW THEN\_KW pair {

System.out.println("Rule 24.7: " +

"bool\_expressions -> AND\_KW THEN\_KW pair");

}

| OR\_KW ELSE\_KW pair {

System.out.println("Rule 24.8: " +

"bool\_expressions -> OR\_KW ELSE\_KW pair");

}

arithmetic\_expressions:

ADD\_KW pair {

System.out.println("Rule 25.1: " +

"arithmetic\_expressions -> ADD\_KW pair");

}

| SUB\_KW pair {

System.out.println("Rule 25.2: " +

"arithmetic\_expressions -> SUB\_KW pair");

}

| MUL\_KW pair {

System.out.println("Rule 25.3: " +

"arithmetic\_expressions -> MUL\_KW pair");

}

| DIV\_KW pair {

System.out.println("Rule 25.4: " +

"arithmetic\_expressions -> DIV\_KW pair");

}

| MOD\_KW pair {

System.out.println("Rule 25.5: " +

"arithmetic\_expressions -> MOD\_KW pair");

}

| SUB\_KW expressions {

System.out.println("Rule 25.6: " +

"arithmetic\_expressions -> SUB\_KW expressions");

}

pair:

OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW {

System.out.println("Rule 26.1: " +

"pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW");

}

**Test Program (Coded by ourselves)**

program test  
 int a;  
 int b:=#3;  
 real f;  
 real k:=#0.4;  
 real l:=#3.5000;  
 char c:='w';  
 char h;  
 bool s;  
 bool r;  
 int array[#2]:={#5,#6};  
 procedure func (int input;) {  
 int x:=#1;  
 int y:=#2;  
 {  
 if < (x,y)  
 then  
 x:= +(x, #1)  
 else  
 y:= -(y, #1);  
  
 do  
 x:=+(x,#1)  
 while <(x, #1);  
  
 for i:=#1 upto #10  
 do d[#1] := +(j,+(i,#2));  
 ;}  
 };  
 procedure func (int input;) {  
 int x:=#1;  
 int y:=#2;  
 {  
 if >(input, #0) then {u:=#89;}  
 else return \*(-#1, input)  
 ;}  
 };  
main {  
 switch k  
 case #3: { for i:=#1 upto #10  
 do d[#1] := +(j,+(i,#2));}  
 case #6: {do var7:= #9 while <=((var9),#8);}  
 default:{  
 {  
 if >(input, #10) then {return input;}  
 ;}  
 ;}  
 end;  
 exit when <=(+(l,k), -(k) );  
 \*((\*(-(#6,#6),#7)),#5);  
}

**Final Results (Parser Output)**

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.1: dec -> ID

Rule 6.1: declarator -> dec

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.2: declarations\_list -> declarations

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.1: dec -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 9.1: initializer -> constant\_expressions

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.2: type\_specifiers -> REAL\_KW

Rule 7.1: dec -> ID

Rule 6.1: declarator -> dec

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.2: type\_specifiers -> REAL\_KW

Rule 7.1: dec -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 9.1: initializer -> constant\_expressions

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.2: type\_specifiers -> REAL\_KW

Rule 7.1: dec -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 9.1: initializer -> constant\_expressions

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.3: type\_specifiers -> CHAR\_KW

Rule 7.1: dec -> ID

Rule 23.3: constant\_expressions -> CHARCONST

Rule 9.1: initializer -> constant\_expressions

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.3: type\_specifiers -> CHAR\_KW

Rule 7.1: dec -> ID

Rule 6.1: declarator -> dec

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.4: type\_specifiers -> BOOLEAN\_KW

Rule 7.1: dec -> ID

Rule 6.1: declarator -> dec

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.4: type\_specifiers -> BOOLEAN\_KW

Rule 7.1: dec -> ID

Rule 6.1: declarator -> dec

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.3: dec -> ID OPENBRACKET\_KW NUMCONST CLOSEBRACKET\_KW

Rule 23.1: constant\_expressions -> NUMCONST

Rule 23.1: constant\_expressions -> NUMCONST

Rule 10.2: initializer\_list -> constant\_expressions

Rule 10.1: initializer\_list -> constant\_expressions COMMA\_KW initializer\_list

Rule 9.2: initializer -> OPENACCOLADE\_KW initializer CLOSEACCOLADE\_KW

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.1: dec -> ID

Rule 6.1: declarator -> dec

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.2: declarations\_list -> declarations

Rule 13.1: parameters -> OPENPARENTHESIS\_KW declarations\_list CLOSEPARENTHESIS\_KW

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.1: dec -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 9.1: initializer -> constant\_expressions

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.2: declarations\_list -> declarations

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.1: dec -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 9.1: initializer -> constant\_expressions

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 22.4: expressions -> ID

Rule 22.4: expressions -> ID

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 24.1: bool\_expressions -> LT\_KW pair

Rule 22.4: expressions -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.1: arithmetic\_expressions -> ADD\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 16.1: statement -> ID ASSIGN\_KW expressions

Rule 22.4: expressions -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.2: arithmetic\_expressions -> SUB\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 16.1: statement -> ID ASSIGN\_KW expressions

Rule 16.3: statement -> IF\_KW bool\_expressions THEN\_KW statement ELSE\_KW statement

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Rule 22.4: expressions -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.1: arithmetic\_expressions -> ADD\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 16.1: statement -> ID ASSIGN\_KW expressions

Rule 22.4: expressions -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 24.1: bool\_expressions -> LT\_KW pair

Rule 16.4: statement -> DO\_KW statement WHILE\_KW bool\_expressions

Rule 15.2: statement\_list -> statement\_list statement SEMICOLON\_KW

Rule 19.1: counter -> NUMCONST UPTO\_KW NUMCONST

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 22.4: expressions -> ID

Rule 22.4: expressions -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.1: arithmetic\_expressions -> ADD\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.1: arithmetic\_expressions -> ADD\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 16.8: statement -> IDENTIFIER OPENBRACKET\_KW expressions CLOSEBRACKET\_KW ASSIGN\_KW expressions

Rule 16.5: statement -> FOR\_KW ID ASSIGN\_KW counter DO\_KW statement

Rule 15.2: statement\_list -> statement\_list statement SEMICOLON\_KW

Rule 15.4: statement\_list -> statement\_list SEMICOLON\_KW

Rule 14.1: block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW

Rule 12.2: procedure -> PROCEDURE\_KW ID parameters OPENACCOLADE\_KW declarations\_list block CLOSEACCOLADE\_KW SEMICOLON\_KW

Rule 11.2: procedure\_list -> procedure

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.1: dec -> ID

Rule 6.1: declarator -> dec

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.2: declarations\_list -> declarations

Rule 13.1: parameters -> OPENPARENTHESIS\_KW declarations\_list CLOSEPARENTHESIS\_KW

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.1: dec -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 9.1: initializer -> constant\_expressions

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.2: declarations\_list -> declarations

Rule 4.1: type\_specifiers -> INTEGER\_KW

Rule 7.1: dec -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 9.1: initializer -> constant\_expressions

Rule 6.2: declarator -> dec ASSIGN\_KW initializer

Rule 5.1: declarator\_list -> declarator

Rule 3.1: declarations -> type\_specifiers declarator\_list SEMICOLON\_KW

Rule 2.1: declarations\_list -> declarations\_list declarations

Rule 22.4: expressions -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 24.3: bool\_expressions -> GT\_KW pair

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 16.1: statement -> ID ASSIGN\_KW expressions

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Rule 14.1: block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW

Rule 16.11: statement -> block

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 25.6: arithmetic\_expressions -> SUB\_KW expressions

Rule 22.3: expressions -> arithmetic\_expressions

Rule 22.4: expressions -> ID

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.3: arithmetic\_expressions -> MUL\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 16.9: statement -> RETURN\_KW expressions

Rule 16.3: statement -> IF\_KW bool\_expressions THEN\_KW statement ELSE\_KW statement

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Rule 14.1: block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW

Rule 12.2: procedure -> PROCEDURE\_KW ID parameters OPENACCOLADE\_KW declarations\_list block CLOSEACCOLADE\_KW SEMICOLON\_KW

Rule 11.1: procedure\_list -> procedure\_list procedure

Rule 22.4: expressions -> ID

Rule 19.1: counter -> NUMCONST UPTO\_KW NUMCONST

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 22.4: expressions -> ID

Rule 22.4: expressions -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.1: arithmetic\_expressions -> ADD\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.1: arithmetic\_expressions -> ADD\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 16.8: statement -> IDENTIFIER OPENBRACKET\_KW expressions CLOSEBRACKET\_KW ASSIGN\_KW expressions

Rule 16.5: statement -> FOR\_KW ID ASSIGN\_KW counter DO\_KW statement

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Rule 14.1: block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW

Rule 20.1: case\_element -> CASE\_KW NUMCONST COLON\_KW block

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 16.1: statement -> ID ASSIGN\_KW expressions

Rule 22.4: expressions -> ID

Rule 22.7: expressions -> OPENPARENTHESIS\_KW expressions CLOSEPARENTHESIS\_KW

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 24.2: bool\_expressions -> LTE\_KW pair

Rule 16.4: statement -> DO\_KW statement WHILE\_KW bool\_expressions

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Rule 14.1: block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW

Rule 20.2: case\_element -> case\_element CASE\_KW NUMCONST COLON\_KW block

Rule 22.4: expressions -> ID

Rule 23.1: constant\_expressions -> NUMCONST

Rule 22.1: expressions -> constant\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 24.3: bool\_expressions -> GT\_KW pair

Rule 22.4: expressions -> ID

Rule 16.9: statement -> RETURN\_KW expressions

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Rule 14.1: block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW

Rule 16.11: statement -> block

Rule 16.2: statement -> IF\_KW bool\_expressions THEN\_KW statement

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Rule 14.1: block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW

Rule 16.11: statement -> block

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Error : syntax error

Rule 14.1: block -> OPENACCOLADE\_KW statement\_list CLOSEACCOLADE\_KW

Rule 21.1: default -> DEFAULT\_KW COLON\_KW block

Rule 16.6: statement -> SWITCH\_KW expressions case\_element default END\_KW

Rule 15.1: statement\_list -> statement SEMICOLON\_KW

Rule 22.4: expressions -> ID

Rule 22.4: expressions -> ID

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.1: arithmetic\_expressions -> ADD\_KW pair

Rule 22.3: expressions -> arithmetic\_expressions

Rule 22.4: expressions -> ID

Rule 22.7: expressions -> OPENPARENTHESIS\_KW expressions CLOSEPARENTHESIS\_KW

Rule 25.6: arithmetic\_expressions -> SUB\_KW expressions

Rule 22.3: expressions -> arithmetic\_expressions

Rule 26.1: pair: OPENPARENTHESIS\_KW expressions COMMA\_KW expressions CLOSEPARENTHESIS\_KW

Rule 24.2: bool\_expressions -> LTE\_KW pair

Rule 16.10: statement -> EXIT\_KW WHEN\_KW bool\_expressions

Rule 15.2: statement\_list -> statement\_list statement SEMICOLON\_KW