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