/\*

PRG ::= "program" identifier ";" BLQ "."

BLQ ::= DCLLIST "begin" SENTLIST "end"

DCLLIST ::= lambda | DCLLIST ";" DCL

SENTLIST ::= SENT | SENTLIST ";" SENT

La zona de declaraciones es una lista de declaraciones de constantes, tipos, variables, procedimientos y/o funciones.

DCL ::= DEFCTE | DEFVAR | DEFPROC | DEFFUN

DEFCTE ::= "const" CTELIST

CTELIST ::= identifier "=" SIMPVALUE ";"

| CTELIST identifier "=" SIMPVALUE ";"

SIMPVALUE ::= numeric\_integer\_const | numeric\_real\_const | string\_const

DEFVAR ::= "var" DEFVARLIST

DEFVARLIST ::= VARLIST ":" TBAS ";"

| VARLIST ":" TBAS ";" DEFVARLIST

VARLIST ::= identifier | identifier "," VARLIST

DEFPROC ::= "procedure" identifier FORMAL\_PARAMLIST ";" BLQ ";"

DEFFUN ::= "function" identifier FORMAL\_PARAMLIST ":" TBAS ";" BLQ ";"

FORMAL\_PARAMLIST ::= lambda | "(" FORMAL\_PARAM ")"

FORMAL\_PARAM ::= VARLIST ":" TBAS

| VARLIST ":" TBAS ";" FORMAL\_PARAM

TBAS ::= "INTEGER" | "REAL" | "CHARACTER"

La zona de sentencias es una lista de sentencias como asignaciones, sentencias de flujo, llamadas a procedimientos y bloques de ejecución anónimos:

SENT ::= ASIG | PROC\_CALL | EXEBLQ

ASIG ::= ID ":=" EXP

ID := identifier

EXP ::= EXP OP EXP | FACTOR

OP ::= OPCOMP | OPLOG | OPARIT

OPCOMP ::= "<" | ">" | "<=" | ">=" | "=" | "<>"

OPARIT ::= "+" | "-" | "\*" | "div" | "mod"

OPLOG ::= "or" | "and"

FACTOR ::= SIMPVALUE | "not" FACTOR

| "(" EXP ")" | identifier SUBPPARAMLIST

SUBPPARAMLIST ::= lambda | "(" EXPLIST ")"

EXPLIST ::= EXP | EXP "," EXPLIST

PROC\_CALL ::= identifier SUBPPARAMLIST

EXEBLQ ::= DCLLIST\_BLQ "begin" SENTLIST "end"

DCLLIST\_BLQ ::= lambda | DCLLIST\_BLQ ";" DCL\_BLQ

DCL\_BLQ ::= DEFCTE | DEFTYPE | DEFVAR

\*/

/\*aqui comienza el programa\*/

import java\_cup.runtime.\*;

terminal program, identifier, begin, end, punto, punto\_coma;

terminal integer, real, char\_, const\_, op\_igual, var, dos\_puntos, coma;

terminal numeric\_integer\_const, numeric\_integerHex\_const, numeric\_real\_const, numeric\_realHex\_const, string\_const;

terminal procedure, function, abrir\_paren, cerrar\_paren, abrir\_corchete, cerrar\_corchete;

terminal op\_asignacion, op\_menor, op\_mayor, op\_menorIgual, op\_mayorIgual, op\_distinto;

terminal op\_suma, op\_resta, op\_mult, op\_divEntero, op\_mod, op\_or, op\_and, op\_not, op\_divReal;

terminal if\_, then, else\_, while\_, do\_, for\_, to, case\_, of, type, array, record;

non terminal PRG, BLQ, DCLLIST, DCL, SENTLIST, SENT;

non terminal ALLTYPES, DEFCTE, CTELIST, SIMPVALUE, DEFTYPE, TYPELIST, UDTYPE, DEFVAR, DEFVARLIST, VARLIST, TBAS, DEFPROC, FORMAL\_PARAMLIST, FORMAL\_PARAM, DEFFUN;

non terminal ASIG, ID, EXP, OP, OPCOMP, OPARIT, OPLOG, FACTOR, SUBPPARAMLIST, EXPLIST, PROC\_CALL, EXEBLQ, DCLLIST\_BLQ;

non terminal COND, ELSECOND, WLOOP, FLOOP, CASE, CASELIST, DCL\_BLQ;

precedence left op\_menor;

precedence left op\_mayor;

precedence left op\_mayorIgual;

precedence left op\_menorIgual;

precedence left op\_igual;

precedence left op\_distinto;

precedence left op\_suma;

precedence left op\_resta;

precedence left op\_mod;

precedence left op\_mult;

precedence left op\_divEntero;

precedence left op\_divReal;

precedence left op\_not;

precedence left op\_or;

precedence left op\_and;

precedence left else\_;

precedence left if\_;

start with PRG;

PRG ::= program identifier punto\_coma BLQ punto;

BLQ ::= DCLLIST begin SENTLIST end | begin SENTLIST end;

DCLLIST ::= DCLLIST DCL | DCL;

SENTLIST ::= SENT | SENTLIST SENT;

DCL ::= DEFCTE | DEFTYPE | DEFVAR | DEFPROC | DEFFUN;

DEFCTE ::= const\_ CTELIST;

CTELIST ::= identifier op\_igual SIMPVALUE punto\_coma | CTELIST identifier op\_igual SIMPVALUE punto\_coma;

DEFTYPE ::= type TYPELIST;

TYPELIST ::= identifier op\_igual UDTYPE punto\_coma | identifier op\_igual UDTYPE punto\_coma TYPELIST;

UDTYPE ::= array abrir\_corchete SIMPVALUE punto punto SIMPVALUE cerrar\_corchete of ALLTYPES | record DEFVARLIST punto\_coma end;

SIMPVALUE ::= numeric\_integer\_const | numeric\_integerHex\_const | numeric\_real\_const | numeric\_realHex\_const | string\_const;

DEFVAR ::= var DEFVARLIST punto\_coma;

DEFVARLIST ::= VARLIST dos\_puntos ALLTYPES | DEFVARLIST punto\_coma VARLIST dos\_puntos ALLTYPES;

VARLIST ::= identifier | identifier coma VARLIST;

DEFPROC ::= procedure identifier FORMAL\_PARAMLIST punto\_coma BLQ punto\_coma

| procedure identifier punto\_coma BLQ punto\_coma;

DEFFUN ::= function identifier FORMAL\_PARAMLIST dos\_puntos ALLTYPES punto\_coma BLQ punto\_coma

| function identifier dos\_puntos ALLTYPES punto\_coma BLQ punto\_coma;

FORMAL\_PARAMLIST ::= abrir\_paren FORMAL\_PARAM cerrar\_paren;

FORMAL\_PARAM ::= VARLIST dos\_puntos ALLTYPES | VARLIST dos\_puntos ALLTYPES punto\_coma FORMAL\_PARAM;

TBAS ::= integer | real | char\_;

ALLTYPES ::= TBAS | identifier;

SENT ::= ASIG punto\_coma | PROC\_CALL punto\_coma | EXEBLQ | COND | WLOOP | FLOOP | CASE;

COND ::= if\_ EXP then SENT| if\_ EXP then SENT ELSECOND;

ELSECOND ::= else\_ SENT;

WLOOP ::= while\_ EXP do\_ SENT;

FLOOP ::= for\_ identifier op\_asignacion EXP to EXP do\_ SENT;

CASE ::= case\_ EXP of CASELIST end punto\_coma;

CASELIST ::= EXP dos\_puntos SENT | EXP dos\_puntos SENT CASELIST;

ASIG ::= ID op\_asignacion EXP;

ID ::= identifier | identifier abrir\_corchete EXP cerrar\_corchete | identifier punto identifier;

EXP ::= EXP OP EXP | FACTOR;

OP ::= OPCOMP | OPLOG | OPARIT;

OPCOMP ::= op\_menor | op\_mayor | op\_menorIgual | op\_mayorIgual | op\_igual | op\_distinto;

OPARIT ::= op\_suma | op\_resta | op\_mult | op\_divEntero | op\_mod | op\_divReal;

OPLOG ::= op\_or | op\_and;

FACTOR ::= SIMPVALUE | op\_not FACTOR | abrir\_paren EXP cerrar\_paren | identifier SUBPPARAMLIST

| identifier abrir\_corchete EXP cerrar\_corchete | identifier | identifier punto identifier;

SUBPPARAMLIST ::= abrir\_paren EXPLIST cerrar\_paren;

EXPLIST ::= EXP | EXP coma EXPLIST;

PROC\_CALL ::= identifier SUBPPARAMLIST | identifier;

EXEBLQ ::= DCLLIST\_BLQ begin SENTLIST end punto\_coma| begin SENTLIST end punto\_coma;

DCLLIST\_BLQ ::= DCLLIST\_BLQ punto\_coma DCL\_BLQ | punto\_coma DCL\_BLQ;

DCL\_BLQ ::= DEFCTE | DEFTYPE | DEFVAR;