PRÁCTICA DE PROCESADORES DE LENGUAJES

Curso 2013 – 2014

Entrega de Febrero

APELLIDOS Y NOMBRE: **Remirez Ruiz, Paula**

IDENTIFICADOR: **premirez1**

DNI: **72680972Y**

CENTRO ASOCIADO MATRICULADO: **Pamplona - NAVARRA**

CENTRO ASOCIADO DE LA SESIÓN DE CONTROL: **Pamplona**

MAIL DE CONTACTO: **pauremirez@gmail.com**

TELÉFONO DE CONTACTO: **657603358**

GRUPO (A ó B): **A**

1. El analizador léxico

[…]

```
//Macros necesarias (Directivas)

LETRA = [A-Za-z]
DIGITO = [0-9]
ESPACIO = [\ \t\r\n]+
CARACTERESCADENA = [^\"]*
ENTERO = {DIGITO}+
ID = {LETRA}({LETRA}|{DIGITO})*
COMENTARIOLINEA=--.*\r\n

%%

<YYINITIAL>
{
"array"{Token token = new Token (sym.ARRAY);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"begin"{Token token = new Token (sym.BEGIN);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"Boolean"{Token token = new Token (sym.BOOLEAN);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"constant"{Token token = new Token (sym.CONSTANT);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"else"{Token token = new Token (sym.ELSE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"end"{Token token = new Token (sym.END);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"false"{Token token = new Token (sym.FALSE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"function"{Token token = new Token (sym.FUNCTION);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"if"{Token token = new Token (sym.IF);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"Integer"{Token token = new Token (sym.INTEGER);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“is"{Token token = new Token (sym.IS);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“loop"{Token token = new Token (sym.LOOP);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“of"{Token token = new Token (sym.OF);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“or"{Token token = new Token (sym.OR);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“out"{Token token = new Token (sym.OUT);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“procedure"{Token token = new Token (sym.PROCEDURE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“Put_line"{Token token = new Token (sym.PUTLINE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“return"{Token token = new Token (sym.RETURN);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“then"{Token token = new Token (sym.THEN);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“True"{Token token = new Token (sym.TRUE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“type"{Token token = new Token (sym.TYPE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“while"{Token token = new Token (sym.WHILE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}

“\""{Token token = new Token (sym.COMILLASDOBLES);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“("{Token token = new Token (sym.PARENTESISIZQ);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“)"{Token token = new Token (sym.PARENTESISDER);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“.."{Token token = new Token (sym.PUNTOPUNTO);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“--"{Token token = new Token (sym.INICIOCOMENTARIO);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“\n"{Token token = new Token (sym.SALTOLINEA);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“,"{Token token = new Token (sym.COMA);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“;"{Token token = new Token (sym.PUNTOYCOMA);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
“:"{Token token = new Token (sym.DOSPUNTOS);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
```

```
"-"{Token token = new Token (sym.MENOS);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"+"{Token token = new Token (sym.MAS);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"<"{Token token = new Token (sym.MENORQUE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
">"{Token token = new Token (sym.MAYORQUE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"/="{Token token = new Token (sym.DISTINTOQUE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"="{Token token = new Token (sym.IGUALQUE);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}

"and"{Token token = new Token (sym.AND);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
":="{Token token = new Token (sym.ASIGNACION);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
"."{Token token = new Token (sym.PUNTO);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}

\"{CARACTERESCADENA}\"
{Token token = new Token (sym.CARACTERESCADENA);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ()); return token;}
{ID}
{Token token = new Token (sym.ID);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
{ENTERO}
{Token token = new Token (sym.ENTERO);token.setLine (yyline + 1);token.setColumn (yycolumn + 1);token.setLexema (yytext ());return token;}
{ESPACIO}
{ } //ESPACIOS EN BLANCO (NO HACEMOS NADA) //
{COMENTARIOLINEA}
{ } //NO HACEMOS NADA
[^]
{LexicalError error = new LexicalError ();error.setLine (yyline + 1);error.setColumn (yycolumn + 1);error.setLexema (yytext
());lexicalErrorManager.lexicalError (error);} }//Fin YYNITIAL
```

2. El analizador sintáctico

```
//-------------------------
// Declaracion de terminales
//-------------------------
terminal Token MAS;
terminal Token ARRAY;
terminal Token BEGIN;
terminal Token BOOLEAN;
terminal Token CONSTANT;
terminal Token ELSE;
terminal Token END;
terminal Token FALSE;
terminal Token FUNCTION;
terminal Token IF;
terminal Token INTEGER;
terminal Token IS;
terminal Token LOOP;
terminal Token OF;
terminal Token OR;
terminal Token OUT;
terminal Token PROCEDURE;
terminal Token PUTLINE;
terminal Token RETURN;
terminal Token THEN;
terminal Token TRUE;
terminal Token TYPE;
terminal Token WHILE;
terminal Token COMILLASDOBLES;
terminal Token PARENTESISIZQ;
terminal Token PARENTESISDER;
terminal Token INICIOCOMENTARIO;
terminal Token SALTOLINEA;
terminal Token COMA;
terminal Token PUNTOYCOMA;
terminal Token DOSPUNTOS;
terminal Token MENOS;
terminal Token MENORQUE;
terminal Token MAYORQUE;
terminal Token IGUALQUE;
terminal Token DISTINTOQUE;
terminal Token AND;
terminal Token ASIGNACION;
terminal Token PUNTO;
terminal Token ID; //Identificador
terminal Token ENTERO;
terminal Token PUNTOPUNTO;
terminal Token CARACTERESCADENA;
```

```
//----------------------------
// Declaracion de no terminales
//----------------------------
non terminal Axiom           axiom;
non terminal                 procedure;
non terminal                 seccionTipos;
non terminal                 seccionVariables;
non terminal                 seccionSubProgramas;
non terminal                 seccionConstantesSimbolicas;
non terminal                 declaracionConstanteSimbolica;
non terminal                 declaracionTipo;
non terminal                 tipoPrimitivo;
non terminal                 declaracionVariable;
non terminal                 declaracionSubPrograma;
non terminal                 funcion;
non terminal                 procedimiento;
non terminal                 parametros;
non terminal                 idParametros;
non terminal                 listaSentencias;
non terminal                 sentencia;
non terminal                 sentenciaProcedimiento;
non terminal                 sentenciaPutLine;
non terminal                 sentenciaWhile;
non terminal                 sentenciaIf;
non terminal                 sentenciaAsignacion;
non terminal                 idTipos;
non terminal                 expresion;
non terminal                 tipoBooleano;
non terminal                 vacio;
non terminal                 cuerpoFuncion;
non terminal                 cabecera;
non terminal                 cuerpo;
non terminal                 sentenciaFuncion;
non terminal                 lista_parametros_llamada;
non terminal                 tipos;
non terminal                 parametro;
non terminal                 modo;
non terminal                 parametroPutLine;
non terminal                 cadenaCaracteres;
non terminal                 vector;

//---------------------------------------
// Declaracion de relaciones de precedencia
//---------------------------------------
precedence left MAS, MENOS;
precedence nonassoc ASIGNACION;
precedence left MENORQUE, MAYORQUE, DISTINTOQUE, IGUALQUE;
precedence left AND, OR;
precedence left PARENTESISIZQ, PARENTESISDER, PUNTO;
precedence right ELSE;
```

3. Conclusiones

He tenido bastantes problemas a la hora de implementar el analizador sintáctico. Uno de los principales retos fue el de ubicar el elemento “vacio” en el punto correcto para no tener fallos durante la compilación y posterior ejecución.

La documentación me ha parecido algo escasa y en determinados momentos difícil de seguir. Al ser una asignatura que está en extinción tampoco he tenido una referencia en el centro asociado para resolver dudas tanto teóricas como prácticas, por lo que esta asignatura, y en especial esta primera parte práctica, me ha resultado en general muy laboriosa y compleja.

Una vez resueltas las grandes dudas respecto a la asignatura, he podido implementar la práctica con “relativa” solvencia.

Esta primera parte me ha parecido muy curiosa, ya que implementar un compilador te hace ver realmente los pasos y procesos que se ejecutan cada vez que programamos. A nivel conceptual me parece muy interesante.

4. Gramática

```
// ----------------------------------------
// Declaracion de las reglas de producción
// ----------------------------------------
axiom::= PROCEDURE ID PARENTESISIZQ PARENTESISDER IS cabecera cuerpo | PROCEDURE ID PARENTESISIZQ PARENTESISDER IS cuerpo;
cabecera::= seccionConstantesSimbolicas seccionTipos seccionVariables seccionSubProgramas |
       seccionConstantesSimbolicas seccionVariables seccionSubProgramas | seccionConstantesSimbolicas seccionTipos seccionSubProgramas |
       seccionConstantesSimbolicas seccionTipos seccionVariables | seccionConstantesSimbolicas seccionSubProgramas |
       seccionConstantesSimbolicas seccionVariables | seccionConstantesSimbolicas | seccionTipos seccionVariables seccionSubProgramas |
       seccionTipos seccionVariables | seccionTipos seccionSubProgramas | seccionTipos |
       seccionVariables seccionSubProgramas | seccionVariables | seccionSubProgramas;
cuerpo::= BEGIN listaSentencias END ID PUNTOYCOMA | BEGIN END ID PUNTOYCOMA;
seccionConstantesSimbolicas::= declaracionConstanteSimbolica | seccionConstantesSimbolicas declaracionConstanteSimbolica ;
seccionTipos::= declaracionTipo | seccionTipos declaracionTipo;
seccionVariables::= declaracionVariable | seccionVariables declaracionVariable;
seccionSubProgramas::= declaracionSubPrograma | seccionSubProgramas declaracionSubPrograma;
declaracionConstanteSimbolica::= ID DOSPUNTOS CONSTANT ASIGNACION tipoPrimitivo PUNTOYCOMA;
declaracionTipo::= TYPE ID IS ARRAY PARENTESISIZQ ENTERO PUNTOPUNTO ENTERO PARENTESISDER OF tipos PUNTOYCOMA;
declaracionVariable::= idParametros tipos PUNTOYCOMA;
declaracionSubPrograma::= procedimiento | funcion;
funcion::= FUNCTION ID PARENTESISIZQ parametros PARENTESISDER RETURN tipos IS cabecera cuerpoFuncion |
       FUNCTION ID PARENTESISIZQ PARENTESISDER RETURN tipos IS cabecera cuerpoFuncion |
       FUNCTION ID PARENTESISIZQ parametros PARENTESISDER RETURN tipos IS cuerpoFuncion |
       FUNCTION ID PARENTESISIZQ PARENTESISDER RETURN tipos IS cuerpoFuncion;
cuerpoFuncion::= BEGIN listaSentencias RETURN expresion PUNTOYCOMA END ID PUNTOYCOMA | BEGIN RETURN expresion PUNTOYCOMA END ID PUNTOYCOMA;
procedimiento::= PROCEDURE ID PARENTESISIZQ parametros PARENTESISDER IS cabecera cuerpo | PROCEDURE ID PARENTESISIZQ parametros PARENTESISDER IS cuerpo;
listaSentencias::= sentencia | listaSentencias sentencia ;
sentencia::=sentenciaAsignacion | sentenciaIf | sentenciaWhile | sentenciaPutLine | sentenciaProcedimiento;
sentenciaPutLine::= PUTLINE PARENTESISIZQ parametroPutLine PARENTESISDER PUNTOYCOMA;
parametroPutLine::= cadenaCaracteres | expresion;
cadenaCaracteres::= CARACTERESCADENA;
sentenciaWhile::= WHILE expresion LOOP listaSentencias END LOOP PUNTOYCOMA;
sentenciaIf::= IF expresion THEN listaSentencias END IF PUNTOYCOMA | IF expresion THEN listaSentencias ELSE listaSentencias END IF PUNTOYCOMA;
sentenciaAsignacion::= idTipos ASIGNACION expresion PUNTOYCOMA;
idTipos::= ID | ID PUNTO ID | ID PUNTO ID PUNTO ID | vector;
vector::= ID PARENTESISIZQ ENTERO PARENTESISDER;
sentenciaFuncion ::= ID PARENTESISIZQ lista_parametros_llamada PARENTESISDER PUNTOYCOMA;
lista_parametros_llamada::= expresion | expresion COMA lista_parametros_llamada;
sentenciaProcedimiento::= idTipos ASIGNACION ID PARENTESISIZQ lista_parametros_llamada PARENTESISDER PUNTOYCOMA |
                          ID PARENTESISIZQ lista_parametros_llamada PARENTESISDER PUNTOYCOMA;
expresion::= expresion MENOS expresion | expresion MAS expresion | expresion OR expresion | expresion AND expresion | expresion MAYORQUE expresion |
             expresion MENORQUE expresion | expresion IGUALQUE expresion | expresion DISTINTOQUE expresion | vector |
             PARENTESISIZQ expresion PARENTESISDER | ID | ENTERO | ID PUNTO ID | ID PUNTO ID PUNTO ID | sentenciaFuncion | tipoBooleano;
tipoPrimitivo::= ENTERO | tipoBooleano;
tipoBooleano::= TRUE | FALSE;
vacio::=;
tipos::= INTEGER | BOOLEAN | ID;
idParametros::= ID DOSPUNTOS | ID COMA idParametros;
parametros::= parametro | parametro PUNTOYCOMA parametros;
parametro::= idParametros modo tipos;
modo::= OUT | vacio;
```