PRÁCTICA DE PROCESADORES DE LENGUAJES

Curso 2013 – 2014

Entrega de Febrero

APELLIDOS Y NOMBRE: García Paredero, Francisco Javier

IDENTIFICADOR:

DNI: 51081513

CENTRO ASOCIADO MATRICULADO: Madrid

CENTRO ASOCIADO DE LA SESIÓN DE CONTROL: Madrid

MAIL DE CONTACTO: garcia.paredero@gmail.com

TELÉFONO DE CONTACTO: 617252500

GRUPO (A ó B): B

Analizador léxico

Los tokens se dividen en varias categorías:

1. Palabras reservadas
2. Operadores
3. Delimitadores
4. Otros

Palabras reservadas	Operadores	Delimitadores	Otros
• begin • Boolean • constant • else • end • False • for • function • if • in • Integer • is • loop • or • procedure • Put_line • record • return • then • True • type	• + • >= • := • .	• “ • (•) • .. • , • ; • :	• Números • Cadenas • Identificadores • Comentarios

ANÁLISIS SINTACTICO

Se define la gramatica

$$G=(T,N,A,P)$$

Donde los terminales son los símbolos tokens identificados por el analizador léxico

T = {`BEGIN; BOOLEAN; CONSTANT; ELSE; END; FALSE; FOR; FUNCTION; IF; IN; INTEGER; IS; Token LOOP; OR; PROCEDURE; PUT_LINE; RECORD; RETURN; THEN; TRUE; TYPE; PLUS; MAYOR; IGUAL; ASIGNACION; ACCESO; ABRE_PARENTESIS; CIERRA_PARENTESIS; DELIMITADOR_RANGO; SEPARADOR_COMA; PUNTO_COMA; DOS_PUNTOS; NUMERO; IDENTIFICADOR; CADENA;`}

Los no terminales son:

N = { program; axiom; vacio; cabecera; declaraciones; declaracion_constantes; declaracion_registros; declaracion_variables; declaracion_subprogramas; inicio_declaracion; lista_nombres; constante; valor; registro; lista_campos; campo; tipo; variable; cuerpo; lista_sentencias; sentencia; sentencia_entrada_salida; sentencia_asignacion; sentencia_if; sentencia_for; referencia; acceso_registro; expresion; parametro; llamada_subprograma; lista_parametros; subprograma; funcion; procedimiento; parametros_subprograma; tipo_primitivo; cuerpo_funcion; lista_sentencias_funcion; sentencia_funcion; sentencia_return;}

Las producciones

```
program ::= axiom;

axiom ::= cabecera declaraciones cuerpo;

vacio ::= ;

cabecera ::= PROCEDURE IDENTIFICADOR ABRE_PARENTESIS CIERRA_PARENTESIS IS;

declaraciones ::= declaracion_constantes;

declaracion_constantes ::= constante declaracion_constantes
        | declaracion_registros;

declaracion_registros ::= registro declaracion_registros
        | declaracion_variables;

declaracion_variables ::= variable declaracion_variables
        | declaracion_subprogramas;

declaracion_subprogramas ::= subprograma declaracion_subprogramas
        | vacio;

subprograma ::= funcion | procedimiento;

funcion ::= FUNCTION IDENTIFICADOR ABRE_PARENTESIS parametros_subprograma
CIERRA_PARENTESIS RETURN tipo_primitivo IS declaraciones cuerpo_funcion;

parametros_subprograma ::= lista_nombres DOS_PUNTOS tipo PUNTO_COMA
                           parametros_subprograma
        | lista_nombres DOS_PUNTOS tipo
        | vacio;

procedimiento ::= PROCEDURE IDENTIFICADOR ABRE_PARENTESIS
parametros_subprograma CIERRA_PARENTESIS  IS declaraciones cuerpo;

cuerpo_funcion::= BEGIN lista_sentencias_funcion END IDENTIFICADOR
PUNTO_COMA;

lista_sentencias_funcion ::=  sentencia_funcion PUNTO_COMA
lista_sentencias_funcion
      |  vacio;

sentencia_funcion ::= sentencia | sentencia_return;

sentencia_return ::= RETURN expresion;

constante ::= inicio_declaracion CONSTANT ASIGNACION valor PUNTO_COMA;
```

```
valor ::= TRUE | FALSE | NUMERO;

registro ::= TYPE IDENTIFICADOR IS RECORD lista_campos END RECORD PUNTO_COMA;

lista_campos ::= campo lista_campos | campo;

campo ::= IDENTIFICADOR DOS_PUNTOS tipo PUNTO_COMA;

tipo ::= IDENTIFICADOR | INTEGER | BOOLEAN;

tipo_primitivo ::= INTEGER | BOOLEAN;

variable ::= inicio_declaracion tipo PUNTO_COMA;

inicio_declaracion ::= lista_nombres DOS_PUNTOS;

lista_nombres ::= IDENTIFICADOR SEPARADOR_COMA lista_nombres | IDENTIFICADOR;

cuerpo ::=  BEGIN lista_sentencias END IDENTIFICADOR PUNTO_COMA

lista_sentencias ::=  sentencia PUNTO_COMA lista_sentencias
        |  vacio;

sentencia ::= sentencia_entrada_salida
        | sentencia_if
        | sentencia_for
        | sentencia_asignacion
        | llamada_subprograma;

sentencia_if ::= IF expresion THEN lista_sentencias END IF
        | IF expresion THEN lista_sentencias ELSE lista_sentencias END IF;


sentencia_for ::=  FOR expresion IN expresion DELIMITADOR_RANGO expresion
LOOP lista_sentencias END LOOP;

sentencia_asignacion ::= referencia ASIGNACION expresion;

referencia ::= IDENTIFICADOR
        | acceso_registro;

acceso_registro ::= IDENTIFICADOR ACCESO acceso_registro
        | IDENTIFICADOR ACCESO IDENTIFICADOR;

expresion ::= expresion PLUS expresion
        | expresion MAYOR expresion
        | expresion IGUAL expresion
        | expresion OR expresion
        | ABRE_PARENTESIS expresion CIERRA_PARENTESIS
        | llamada_subprograma
        | referencia
        | CADENA
        | TRUE
        | FALSE
        | NUMERO;

sentencia_entrada_salida ::= PUT_LINE ABRE_PARENTESIS parametro
CIERRA_PARENTESIS;
```

```
parametro ::= CADENA
        | NUMERO
        | referencia;

llamada_subprograma ::= IDENTIFICADOR ABRE_PARENTESIS lista_parametros
CIERRA_PARENTESIS;

lista_parametros ::= expresion SEPARADOR_COMA lista_parametros
        | expresion
    | vacio;
```

Con objeto de eliminar la ambigüedad se insertan explícitamente las reglas de precedencia:

```
precedence left OR;
precedence left IGUAL;
precedence left MAYOR;
precedence left PLUS;
precedence left ACCESO, ABRE_PARENTESIS, CIERRA_PARENTESIS;
```