Token	Expresión Regular	Código del Token	Atributos
TIPO	(“: set of integer” | ”: set of real” | “: set of character” | “: set of boolean” | “: integer” | “: real” | “: character” | “: boolean”)	257	{: set of integer,: set of real, : set of character,: set of boolean : integer,: real,: character,: boolean}
OPBI	(“>” | “>=” | “<” | “<=” | “=” | “¬=” | “<>“ | “and” | “or” | “xor” | “in“ | “*” | “/”)	258	{>, >=, <, <=, =, ¬=, <> ,and, or, xor, in, *, /}
OPUN	(“+” | “-“)	259	{+, -}
CONS	(“<carácter>” | “<booleano>” | “<constanteReal>”)	260	{<carácter>, <booleano>, <constanteReal>}
CONSEN	“<constanteEntera>”	261	{<constanteEntera>}
IDEN	“<identificador>”	262	{<identificador>}
NOT	“not”	263	{not}
CO	“,”	264	{,}
PAA	“ (“	265	{(}
PAC	“)”	266	{)}
SALIDA	“<nomb_salida>”	267	{<nomb_salida>}
FRASE	“<frase>”	268	{<frase>}
COSIM	‘“’	269	{'}
CODOB	‘“‘“’	270	{“}
ENTRADA	“<nomb_entrada>”	271	{<nomb_entrada>}
FOR	“for”	272	{ for }
IGUAL	“:=”	273	{:=}
TO	“to”	274	{ to }
DO	“do”	275	{ do }
WHILE	“while”	276	{ while }
IF	“if”	277	{ if }
THEN	“then”	278	{ then }
ELSE	“else”	279	{ else }
ELIF	“else if”	280	{ else if }
PUNCO	“;”	281	{;}
FUNCION	“function”	282	{ function }
BEGIN	“begin”	283	{ begin }
END	“end”	284	{ end }
VAR	“var”	285	{var}

PROGRAM	“program”	286	{program}
PUN	“.”	287	{.}
PUN2	“..”	288	{..}
CORA	“[“	289	{[}
CORC	“]”	290	{]}

```
<sentencia_for> ::= "for" <identificador>":="<expresion> "to" <expresion>
                     "do" <Sentencias>

<sentencia_entrada> ::= <nomb_entrada> <lista_variables>

<lista_variables> ::= <identificador>
                    | <identificador> "," <lista_variables>

<nomb_entrada> ::= "readln"
                 | "read"

<sentencia_salida> ::= <nomb_salida> <lista_expresiones_o_cadena>

<lista_expresiones_o_cadena> ::= <expresion>
    | <lista_expresiones_o_cadena> "," <expresion>
    | <frase>
    | <lista_expresiones_o_cadena> "," <frase>

 <frase> ::= """<identificador>"""
            |"""<digito><identificador>"""
<nomb_salida> ::= "writeln"
                 |"write"

<funcion> ::= <identificador> "(" <lista_expresiones> ")"

<lista_expresiones> ::= <expresion>
                      | <lista_expresiones> "," <expresion>

<sentencia_return> ::= <sentencia_asignacion>

<expresion> ::= "("<expresion>")"
              | "+"<expresion>
              | "-"<expresion>
              | <expresion>"+"<expresion>
              | <expresion>"-"<expresion>
              | <expresion>"*"<expresion>
              | <expresion>"/"<expresion>
              | <expresion> ">" <expresion>
              | <expresion> ">=" <expresión
              | <expresion> "<" <expresion>
              | <expresion> "<=" <expresion>
              | <expresion> "=" <expresion>
              | <expresion> "¬=" <expresion>
              | <expresion> "<>" <expresion>
              | "not"<expresion>
              | <expresion>"and"<expresion>
              | <expresion>"or"<expresion>
              | <expresion>"xor"<expresion>
              | <expresion>"in"<expresion>
              | <identificador>
              | <constante>
              | <conjunto>
```

```
              | <funcion>

<identificador> ::= <letra> <restoIdentificador>
                  | "_"<restoIdentificador>

<restoIdentificador> ::= <restoIdentificador> <digito>
                       | <restoIdentificador> <letra>
                       | <digito>
                       | <letra>
                       | ""

<conjunto> ::= "["<restoConjunto>"]"

<restoConjunto> ::= <expresion> "," <restoConjunto>
                  | <constante>".."<constante>
                  |<expresion>
                  | ""

<constante> ::= <constanteReal>
              | <constateEntera>
              | <caracter>
              | <booleano>

<constanteReal> ::= <constanteEntera>"."<constanteEntera>

<constanteEntera> ::= <constanteEntera> <digito>
                    | <digito>

<caracter> ::= "'"<letra>"'"

<digito> ::= "0"|"1"|"2"|"3"|"4"|"5"|"6"|"7"|"8"|"9"

<letra> ::= "a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|"l"|"m"
          |"n"|"o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|"w"|"x"|"y"|"z"
          |"A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"
          |"N"|"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z"

<booleano> ::= "true"
             | "false"
```

Lexico

```
%{
#include "tabla.h"
        int numero_linea=0;
%}
blanco [ \t]
letra [a-zA-Z]
digito [0-9]
boolean ("true"|"false")
otros .
%%
{blanco}+
": set of integer" return TIPO;
": set of real" return TIPO;
": set of character" return TIPO;
": set of boolean" return TIPO;
": integer"  return TIPO;
": real" return TIPO;
": character" return TIPO;
": boolean" return TIPO;
">"                 return OPBI;
">="            return OPBI;
"<"                 return OPBI;
"<="            return OPBI;
"="                 return OPBI;
"¬="            return OPBI;
"<>"            return OPBI;
```

```
"and"           return OPBI;
"or"            return OPBI;
"xor"           return OPBI;
"in"            return OPBI;
"*"                 return OPBI;
"/"                 return OPBI;
"+"                     return OPUN;
"-"                     return OPUN;
"not"           return NOT;
","                     return CO;
"("                     return PAA;
")"                     return PAC;
""""                    return COSIM;
\"                      return CODOB;
"for"                   return FOR;
":="                    return IGUAL;
"to"                    return TO;
"do"                    return DO;
"while"                         return WHILE;
"if"                    return IF;
"then"                  return THEN;
"else"                  return ELSE;
"else if"                       return ELIF;
";"                     return PUNCO;
"function"                      return FUNCION;
"begin"                         return BEGIN;
"end"                   return END;
```

```
"var"                   return VAR;
"program"               return PROGRAM;
"."                     return PUN;
".."                    return PUN2;
"["                     return CORA;
"]"                     return CORC;
("writeln"|"write") return SALIDA;
("readln"|"read") return ENTRADA;

("'"{letra}"'"|{boolean}|{digito}+"."{digito}+)        return CONS;
{digito}+    return CONSEN;
({letra}|"_")({letra}|{digito})* return IDEN;
\"({letra}|{digito})+\" return FRASE;
"\n" ++numero_linea;
{otros} printf("\n(Linea %d) Error léxico: Lexema %s\n", numero_linea, yytext);
%%
main () {
int val;
val= yylex() ;
while (val != 0) {
        printf("lexema= %s, codigo de token= %d\n",yytext,val);
        val= yylex() ;
        }
exit (1);
}
int yywrap()
{
```

```
return(1);
}
```