ANALIZADOR LÉXICO

GRAMÁTICA

S → lA | dB | “C | /D | |E | = | , | ; | (|) | { | } | < | EOF | del S

A → lA | dA | λ

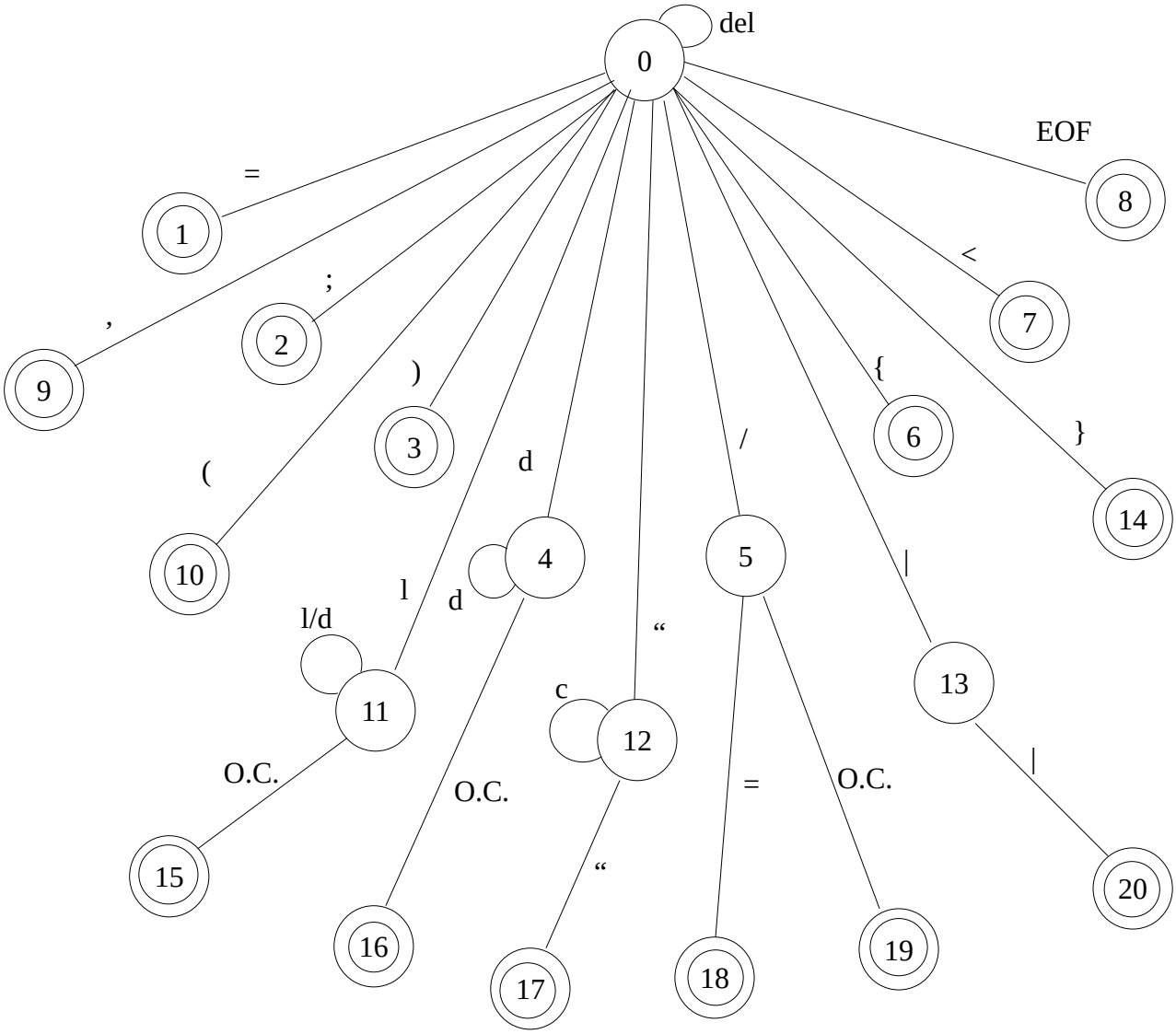
B → dB | λ

C → cC | “

D → = | λ

E → |

AFD



TOKENS

<TYPEBOOL,- >
<LOOPDOW,- >
<DECLFUNC,- >
<CONDIF,- >
<IOIN,- >
<TYPEINT,- >
<IOOUT,- >
<RET,- >
<TYPESTR,- >
<DECLVAR,- >
<LOOPWHL,- >
<ID,num >
<OPASSIGN,- >
<SEPCOM,- >
<SEPSCOM,- >
<SEPLPAR,- >
<SEPRPAR,- >
<SEPLCURL,- >
<SEPRCURL,- >
<OPLT,- >
<VALINT,num>
<SEPSTR,-”c*”>
<OPDIVASSIGN,- >
<OPDIV,- >
<OPOR,- >
<DELEOF,- >

ACCIONES SEMÁNTICAS

0:0 leer()

0:1 GenToken(OPASSIGN, -)
0:9 GenToken(SEPCOM, -)
0:2 GenToken(SEPSCOM, -)
0:10 GenToken(SEPLPAR, -)
0:3 GenToken(SEPRPAR, -)
0:6 GenToken(SEPLCURL, -)
0:14 GenToken(SEPRCURL, -)
0:7 GenToken(OPLT, -)
0:8 GenToken(DELEOF, -)

0:4 num = d; leer()
4:4 num = num *10 + d; leer()
4:16 GenToken(VALINT, num)

0:12 cadena = “
12:12 cadena = cadena + c; leer()
12:17 cadena = cadena + c;
GenToken(SEPSTR, cadena)

```
0:5     leer()
5:18    GenToken(OPDIVASSIGN, -)
5:19    GenToken(OPDIV, -)

0:13    leer()
13:20   GenToken(OPOR, -)

0:11    lexema = l; leer()
11:11   lexema = lexema + l/d; leer()
11:15   p := buscarTS(lexema)
        If (lexema = “boolean”)
                Then GenToken(TYPEBOOL, -)

        else If (lexema = “do”)
                Then GenToken(LOOPDOW, -)

        else If (lexema = “function”)
                Then GenToken(DECLFUNC, -)

        else If (lexema = “if”)
                Then GenToken(CONDIF, -)

        else If (lexema = “input”)
                Then GenToken(IOIN, -)

        else If (lexema = “int”)
                Then GenToken(TYPEINT, -)


        else If (lexema = “print”)
                Then GenToken(IOOUT, -)

        else If (lexema = “return”)
                Then GenToken(RET, -)

        else If (lexema = “string”)
                Then GenToken(TYPESTR, -)

        else If (lexema = “var”)
                Then GenToken(DECLVAR, -)

        else If (lexema = “while”)
                Then GenToken(LOOPWHL, -)

        else If (p = NULL)
                Then insertarTS(lexema)

        GenToken(ID, p)
```

PRUEBAS

PRUEBA1

```
var string texto;
function imprime (string msg)
{
	print ("Mensage introducido:");
	print (msg);
}
function pideTexto ()
{
	print ("Introduce un texto");
	input (texto);
}
pideTexto();
var string textoAux;
textoAux = texto;
imprime (textoAux);
```

------TOKENS------

```
<DECLVAR, >
<TYPESTR, >
<ID, 0>
<SEPSCOM, >
<DECLFUNC, >
<ID, 1>
<SEPLPAR, >
<TYPESTR, >
<ID, 0>
<SEPRPAR, >
<SEPLCURL, >
<IOOUT, >
<SEPLPAR, >
<SEPSTR, "Mensage introducido:">
<SEPRPAR, >
<SEPSCOM, >
<IOOUT, >
<SEPLPAR, >
<ID, >
<SEPRPAR, >
<SEPSCOM, >
<SEPRCURL, >
<DECLFUNC, >
<ID, 2>
<SEPLPAR, >
<SEPRPAR, >
<SEPLCURL, >
<IOOUT, >
<SEPLPAR, >
<SEPSTR, "Introduce un texto">
<SEPRPAR, >
<SEPSCOM, >
```

<IOIN, >
<SEPLPAR, >
<ID, >
<SEPRPAR, >
<SEPSCOM, >
<SEPRCURL, >
<ID, 2>
<SEPLPAR, >
<SEPRPAR, >
<SEPSCOM, >
<DECLVAR, >
<TYPESTR, >
<ID, 3>
<SEPSCOM, >
<ID, 3>
<OPASSIGN, >
<ID, 0>
<SEPSCOM, >
<ID, 1>
<SEPLPAR, >
<ID, 3>
<SEPRPAR, >
<SEPSCOM, >
<DELEOF, >

------TABLAS------

CONTENIDO DE LA TABLA # 1 :

* LEXEMA : 'texto'
---
* LEXEMA : 'imprime'
---
* LEXEMA : 'pideTexto'
---
* LEXEMA : 'textoAux'

CONTENIDO DE LA TABLA # 2 :

* LEXEMA : 'msg'

CONTENIDO DE LA TABLA # 3 :

PRUEBA2

```
var int number_trees;

function sayHowManyTrees() {
    var string trees;
    print("how many trees does elon want?");
    input (trees)
    return trees
}

function treelon(string elon, string musk)
{
   number_trees = sayHowManyTrees()
   print("Treelon Musk has planted")
   print(number_trees)
}
```

------TOKENS------

<DECLVAR, >
<TYPEINT, >
<ID, 0>
<SEPSCOM, >
<DECLFUNC, >
<ID, 1>
<SEPLPAR, >
<SEPRPAR, >
<SEPLCURL, >
<DECLVAR, >
<TYPESTR, >
<ID, >
<SEPSCOM, >
<IOOUT, >
<SEPLPAR, >
<SEPSTR, "how many trees does elon want?">
<SEPRPAR, >
<SEPSCOM, >
<IOIN, >
<SEPLPAR, >
<ID, >
<SEPRPAR, >
<RET, >
<ID, >
<SEPRCURL, >
<DECLFUNC, >
<ID, 2>
<SEPLPAR, >
<TYPESTR, >
<ID, 0>
<SEPCOM, >
<TYPESTR, >
<ID, 1>

<SEPRPAR, >
<SEPLCURL, >
<ID, >
<OPASSIGN, >
<ID, >
<SEPLPAR, >
<SEPRPAR, >
<IOOUT, >
<SEPLPAR, >
<SEPSTR, "Treelon Musk has planted">
<SEPRPAR, >
<IOOUT, >
<SEPLPAR, >
<ID, >
<SEPRPAR, >
<SEPRCURL, >
<DELEOF, >

------TABLAS------

CONTENIDO DE LA TABLA # 1 :

* LEXEMA : 'number_trees'
---
* LEXEMA : 'sayHowManyTrees'
---
* LEXEMA : 'treelon'

CONTENIDO DE LA TABLA # 3 :

* LEXEMA : 'elon'
---
* LEXEMA : 'musk'

PRUEBA3

```
var int num1;
var int num2;
function doPatata() {
   num 2 = num1;
   num2 /= num1;
}

function performPatata() {
   print("takes a number makes a patata");
   var int patata;
   doPatata();
   var int num3;
   patata = num3 / num2;
   patata = num2;
   if (patata < 0 || 0 < patata) {
   }
```

}

------TOKENS------

<DECLVAR, >
<TYPEINT, >
<ID, 0>
<SEPSCOM, >
<DECLVAR, >
<TYPEINT, >
<ID, 1>
<SEPSCOM, >
<DECLFUNC, >
<ID, 2>
<SEPLPAR, >
<SEPRPAR, >
<SEPLCURL, >
<ID, >
<VALINT, >
<OPASSIGN, >
<ID, >
<SEPSCOM, >
<ID, >
<OPDIVASSIGN, >
<ID, >
<SEPSCOM, >
<SEPRCURL, >
<DECLFUNC, >
<ID, 3>
<SEPLPAR, >
<SEPRPAR, >
<SEPLCURL, >
<IOOUT, >
<SEPLPAR, >
<SEPSTR, "takes a number makes a patata">
<SEPRPAR, >
<SEPSCOM, >
<DECLVAR, >
<TYPEINT, >
<ID, >
<SEPSCOM, >
<ID, >
<SEPLPAR, >
<SEPRPAR, >
<SEPSCOM, >
<DECLVAR, >
<TYPEINT, >
<ID, >
<SEPSCOM, >
<ID, >
<OPASSIGN, >
<ID, >

<OPDIV, >
<ID, >

------TABLAS------

CONTENIDO DE LA TABLA # 1 :

* LEXEMA : 'num1'
---
* LEXEMA : 'num2'
---
* LEXEMA : 'doPatata'
---
* LEXEMA : 'performPatata'

CONTENIDO DE LA TABLA # 2 :

CONTENIDO DE LA TABLA # 3 :

PRUEBA4(ERROR)

```
function potata(var string patata) {
   print($$$$$$$$$$)
   input(%)
   var #
   +++++
}
```

------ERRORES------

ERROR : linea 2
ERROR : linea 3
ERROR : linea 4

PRUEBA5(ERROR)

```
var string texto;
var string texto;
var string texto;
var string texto;
var string texto;
function texto(var string texto) {
   print("texto")
\
   texto("texto")
}

var int variable = 1 / 7
int =/ 7
var int string =  variable * 6
texto();
```

------ERRORES------

ERROR : linea 8
ERROR : linea 14

PRUEBA6(ERROR)

```
var string panzer = 12345;

doPanzer(panzer){}
var int ok = 3 * 2 / 1 + 1 - 4;
var int variable = panzer * panzer;
panzer *= 57;
57 /= panzer;
panzer2 -= hio
var +=  var
*//*
var string panzer;
var string var = "var";

function doPanzer(var string no) {
   print("Hola panzer");
}
```

------ERRORES------

ERROR : linea 4
ERROR : linea 5
ERROR : linea 6
ERROR : linea 8
ERROR : linea 9
ERROR : linea 10