

Procesador de Lenguaje JavaScript PDL

Grupo 46

Sofía Hernández Montero
18M046

Jaime González Delgado
18M048

Fernando Bellido Pazos
18M008



POLITÉCNICA

UNIVERSIDAD
POLITÉCNICA
DE MADRID

Universidad Politécnica de Madrid
Grado de Matemáticas e Informática
Procesadores de Lenguajes
2020-2021

Contenido

OBJETIVOS.....	3
OBJETIVOS COMUNES.....	3
OBJETIVOS ESPECÍFICOS.....	3
EL PROYECTO.....	3
ANALIZADOR LÉXICO.....	4
TOKENS.....	4
GRAMÁTICA DEL LENGUAJE.....	4
<i>Leyenda</i>	4
AUTÓMATA FINITO DETERMINISTA	5
ACCIONES SEMÁNTICAS	5
AUTÓMATA TABULAR	7
TABLA DE SÍMBOLOS.....	8
DISEÑO	8
ANALIZADOR SINTÁCTICO.....	9
GRAMÁTICA.....	9
<i>Aumentada</i>	9
<i>Inicio</i>	9
<i>Funciones</i>	9
<i>Sentencias:</i>	9
<i>Expresiones</i>	9
<i>Argumentos de la función</i>	9
<i>Return</i>	9
LR(1).....	10
<i>Justificación de LR(1)</i>	16
ANALIZADOR SEMÁNTICO	17
AUMENTADA.....	17
INICIO.....	17
FUNCIONES.....	17
SENTENCIAS:.....	18
<i>Simples</i>	18
<i>Compuestas</i>	19
EXPRESIONES.....	20
ARGUMENTOS DE LA FUNCIÓN	22
RETURN	22
TIPOS USADOS	22
OBSERVACIONES	22
ERRORES.....	23
EJEMPLOS DE ERRORES ESPECÍFICOS	23
CÓDIGO PARA VAST	24
MINI-DEMO VISUAL	25
WEBGRAFÍA.....	26

Objetivos

La Práctica consistirá en el diseño y construcción de un Analizador de una versión del lenguaje JavaScript llamado JavaScript-PDL.

Objetivos comunes

- La estructura general de un programa compuesto por funciones, declaraciones y sentencias.
- Definición de funciones.
- Tipos enteros, lógicos y cadenas.
- Variables y su declaración.
- Constantes enteras y cadenas de caracteres.
- Sentencias: asignación, condicional simple, llamada a funciones y retorno.
- Expresiones.
- Comentarios.
- Operaciones de entrada/salida por terminal:
 - input
 - alert
- Operadores:
 - Aritméticos: +, -
 - Relacionales: ==, !=
 - Lógicos: &&, ||

Objetivos específicos

- Sentencias: **Sentencia repetitiva (do-while)**
- Operadores especiales: **Asignación con resta (-=)**
- Técnicas de Análisis Sintáctico: **Ascendente**
- Comentarios: **Comentario de bloque (/* */)**
- Cadenas: **Con comillas dobles (" ")**

El proyecto

Cabía destacar, que todos los ficheros propios del proyecto se encuentran disponibles en GitHub: <https://github.com/fbellidopazos/ProcesadoresLenguajes>

Analizador Léxico

Tokens

<abrirCorchete, ->	<cadena, lexema>	<return, ->
<cerrarCorchete, ->	<restaAsignacion, ->	<input, ->
<abrirParentesis, ->	<opAritmetico, 2>	<alert, ->
<cerrarParentesis, ->	<cteEntera, digito>	<if, ->
<puntoYcoma, ->	<opAritmetico, 1>	<number, ->
<coma, ->	<opLogico, 2>	<boolean, ->
<opRelacional, 2>	<opLogico, 1>	<string, ->
<opRelacional, 1>	<do, ->	<let, ->
<asignacion, ->	<while, ->	<EOF, ->
<identificador, postTS>	<function, ->	

Operador Artimetrico	Operador Logico	Operador relacional
1: +	1: &&	1: ==
2: -	2:	2: !=

Gramática del Lenguaje

$$S \rightarrow delS \mid lA \mid "C \mid dE \mid -G \mid |i \mid \&J \mid = N \mid !Q \mid /U \mid + \mid c.e$$

$$A \rightarrow dA \mid lA \mid o.c \mid _A$$

$$C \rightarrow c_1C \mid "$$

$$E \rightarrow dE \mid o.c$$

$$G \rightarrow = \mid o.c$$

$$I \rightarrow |$$

$$J \rightarrow \&$$

$$N \rightarrow = \mid o.c$$

$$Q \rightarrow =$$

$$U \rightarrow * Y$$

$$Y \rightarrow c_2Y \mid * Z$$

$$Z \rightarrow c_3Y \mid * Z \mid /S$$

Leyenda

$c1 = cualquier\ carácter \mid \{ " \}$

$c2 = cualquier\ carácter \mid \{ * \}$

$c3 = cualquier\ carácter \mid \{ *, / \}$

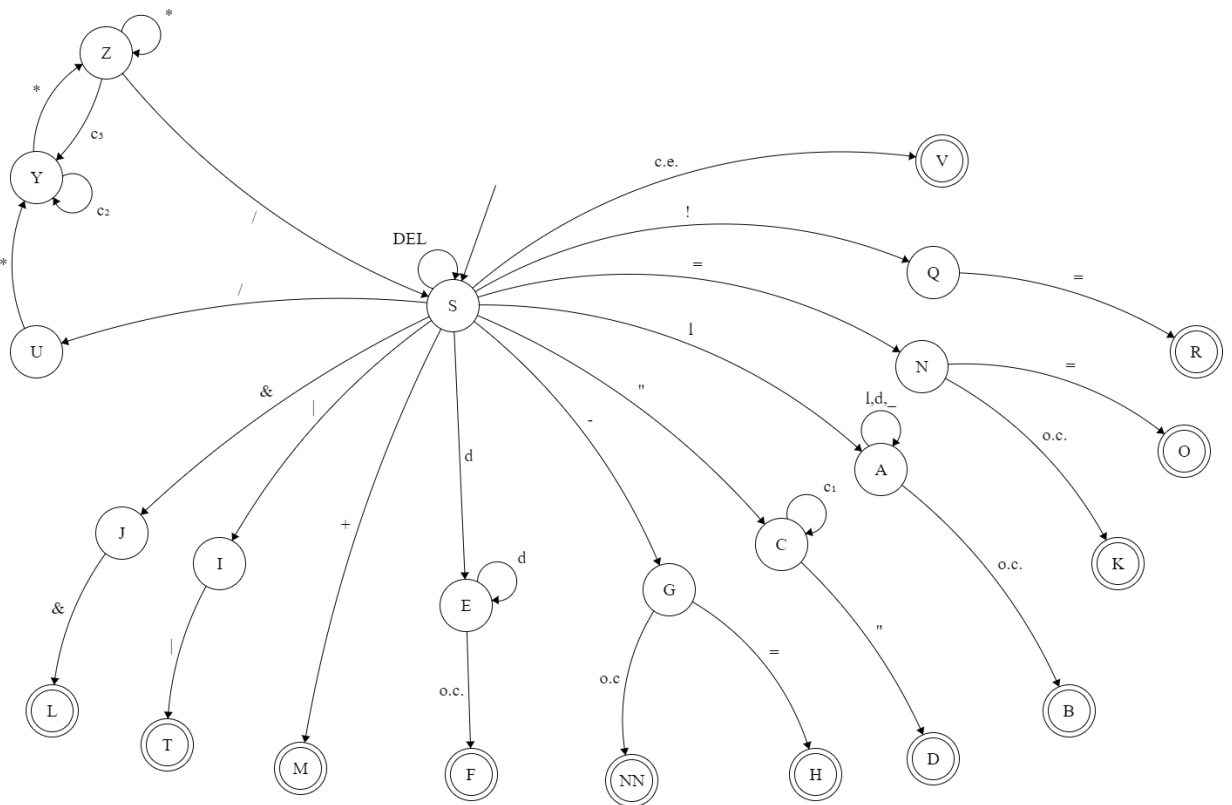
$c.e = caracteres\ especiales: \{ \}, (,), ;, ' \}$

$d = dígitos\ del\ 0\ al\ 9$

$l = letras\ de\ la\ a - z, A - Z$

$del = delimitadores: eol, tab, esp, etc.$

Autómata Finito Determinista



Acciones Semánticas

$S \rightarrow V$: Comprobar tipo carácter especial: Enviar token correspondiente; Leer;

$S \rightarrow Q$: leer;

$Q \rightarrow R$: Gen_Token(<opRelacional,2>)

$S \rightarrow N$: leer;

$N \rightarrow K$: Gen_Token(<asignación, ->)

$N \rightarrow O$: Gen_Token(<opRelacional,1>); leer;

$S \rightarrow A$: lexema=c;leer; //(Siendo c el carácter leído)

$A \rightarrow A$: lexema=lexema \oplus c; leer; //(Siendo c el carácter leído)

$A \rightarrow B$:

```
if(isReservada){
```

```
    Gen_Token(<lexema,->) // Tomamos el mismo nombre de la palabra
    puesto que es Case-Sensitive JS
```

```
}else{
```

```
    Gen_Token(<identificador,posTs>);
```

```
}
```

```
S→C: lexema= ""; leer();
C→C: lexema= lexema⊕ c1; leer();
C→D:
if(lexema.length >64){
    error
}else{
    Gen_Token(<Cadena,lexema>);
}

leer;
S→G: leer();
G→H: Gen_Token(<restaAsignacion,->); leer();
G→NN: Gen_Token(<opAritmetico,2>); leer();
S→E: digito=char2Int(d); leer();
E→E: digito=digito*10+ char2Int(d); leer();
E→F: if(digito > 32767){
    error
}else{
    Gen_Token(<Cte-entera,digito>);
}

S→M: Gen_Token(<opAritmetico,1>); leer();
S→I:leer();
I→T: Gen_Token(<opLogico,2>);leer();
S→J: leer();
J→L: Gen_Token(<opLogico,1>)
```

Autómata Tabular

	del	l	d	-	"	(or)	&	=	!	/	ce	c1	c2	c3	+	*	o.c	_
S	S	A	E	G	C	I	J	N	Q	U	V				M			
A		A	A														B	A
C					D							C						
E			E														F	
G								H									NN	
I						T												
J							L											
N								O									K	
Q								R										
U																Y		
Y													Y			Z		
Z										S				Y		Z		

Donde la posición (fila, columna) representa al estado que se llega desde el estado <fila> con carácter <columna>

Tabla de Símbolos

Diseño

Lexema	Tipo	Despl	numParam	TipoRetorno	EtiquFuncion

La tabla es no homogénea, ya que por cada parámetro de una función se añade una columna “TipoParam<numero>” indicando el tipo de parámetro que es el i-ésimo parámetro de la función

Analizador Sintáctico

Gramática

Aumentada

$$Z \rightarrow P$$

Inicio

$$P \rightarrow BP \mid FP \mid \lambda$$

Funciones

$$F \rightarrow I J G$$
$$I \rightarrow \text{function } H \text{ id}$$
$$J \rightarrow (A)$$
$$G \rightarrow \{ C \}$$
$$H \rightarrow T \mid \lambda$$
$$A \rightarrow T \text{ id } K \mid \lambda$$
$$K \rightarrow , T \text{ id } K \mid \lambda$$
$$C \rightarrow BC \mid \lambda$$

Sentencias:

Simples

$$S \rightarrow \text{id} = E ; \mid \text{id} (L) ; \mid \text{alert} (E) ; \mid \text{input} (\text{id}) ; \mid \text{return } X ; \mid \text{id} -= E ;$$

Compuestas

$$B \rightarrow \text{if} (E) S \mid \text{let } T \text{ id} ; \mid S \mid \text{do} \{ C \} \text{ while} (E) ;$$
$$T \rightarrow \text{number} \mid \text{boolean} \mid \text{string}$$

Expresiones

$$E \rightarrow E \mid \mid R \mid R$$
$$R \rightarrow R \&\& U \mid U$$
$$U \rightarrow U == V \mid U != V \mid V$$
$$V \rightarrow V + W \mid V - W \mid W$$
$$W \rightarrow \text{id} \mid (E) \mid \text{id} (L) \mid \text{entero} \mid \text{cadena}$$

Argumentos de la función

$$L \rightarrow E Q \mid \lambda$$
$$Q \rightarrow , E Q \mid \lambda$$

Return

$$X \rightarrow E \mid \lambda$$

LR(1)

A continuación, dejamos las tablas de First's y Follows , tabla del cálculo de la colección canónica, tabla Acción y tabla GoTo, en ese mismo orden.

FIRST / FOLLOW table		
Nonterminal	FIRST	FOLLOW
Z	{',if,let,identificador,alert,input,return,do,function	{}
P	{',if,let,identificador,alert,input,return,do,function	{}
F	{function}	{\$,if,let,identificador,alert,input,return,do,function}
I	{function}	{abrirParentesis}
J	{abrirParentesis}	{abrirCorchete}
G	{abrirCorchete}	{\$,if,let,identificador,alert,input,return,do,function}
H	{',number,boolean,string}	{identificador}
A	{',number,boolean,string}	{cerrarParentesis}
K	{coma,' '}	{cerrarParentesis}
C	{',if,let,identificador,alert,input,return,do}	{cerrarCorchete}
S	{identificador,alert,input,return}	{\$,if,let,identificador,alert,input,return,do,function,cerrarCorchete}
B	{if,let,identificador,alert,input,return,do}	{\$,if,let,identificador,alert,input,return,do,function,cerrarCorchete}
T	{number,boolean,string}	{identificador}
E	{identificador,abrirParentesis,cteEntera,cadena}	{puntoYcoma,cerrarParentesis,opLogico2,coma}
R	{identificador,abrirParentesis,cteEntera,cadena}	{puntoYcoma,cerrarParentesis,opLogico2,opLogico1,coma}
U	{identificador,abrirParentesis,cteEntera,cadena}	{puntoYcoma,cerrarParentesis,opLogico2,opLogico1,opRelacional1,opRelacional2,coma}
V	{identificador,abrirParentesis,cteEntera,cadena}	{puntoYcoma,cerrarParentesis,opLogico2,opLogico1,opRelacional1,opRelacional2,opAritmetico1,opAritmetico2,coma}
W	{identificador,abrirParentesis,cteEntera,cadena}	{puntoYcoma,cerrarParentesis,opLogico2,opLogico1,opRelacional1,opRelacional2,opAritmetico1,opAritmetico2,coma}
L	{',identificador,abrirParentesis,cteEntera,cadena}	{cerrarParentesis}
Q	{coma,' '}	{cerrarParentesis}
X	{',identificador,abrirParentesis,cteEntera,cadena}	{puntoYcoma}

State	function	identificador	abrirParentesis	cerrarParentesis	abrirCorchete	cerrarCorchete	coma	asignacion	puntoComa	ACTION										while	number	boolean	string	opLogico1	opLogico2	opRelacional1	opRelacional2	opAritmetico1	opAritmetico2	cteEntera	cadena	\$
										alert	input	return	restaAsignacion	if	let	do																
0	a13	a9								a10	a11	a12		a4	a5	a7										r1						
2	a13	a9								a10	a11	a12		a4	a5	a7										acc						
3	a13	a9								a10	a11	a12		a4	a5	a7										r1						
4			a16																							r1						
5																	a18	a19	a20													
6	F23	F24			a21	F24				F24	F24	F24		F24	F24	F24										F24						
7																																
8			a23																													
9			a25						a24				a26																			
10			a27																													
11			a28																													
12	a35	a36								F23														a37	a38							
13	x1																a18	a19	a20													
14																										r1						
15																										r1						
16	a35	a36																						a37	a38							
17	a42																															
18	F23																															
19	F23																															
20	F23																															
21	a9				a46	F15				a10	a11	a12		a4	a5	a7																
22																																
23				F11																												
24	a35	a36																						a37	a38							
25	a35	a36																						a37	a38							
26	a35	a36																						a37	a38							
27	a35	a36																						a37	a38							
28	a54																															
29										a55																						
30										F24																						
31				F15						F23									a56													
32				F15						F32									F31	F32	a58	a59										
33				F15						F15									F31	F31	F31	F31	a60	a61								
34				F15						F18									F38	F38	F38	F38	F38	F38								
35			a62							F15									F38	F38	F38	F38	F38	F38								
36	a35	a36																	F38	F38	F38	F38	F38	F38	a37	a38						
37				F41						F32									F41	F41	F41	F41	F41	F41								
38				F41						F41									F41	F41	F41	F41	F41	F41								
39	a64																															
40	F4																															
41				a65																												
42										a66									a66													
43						a67																										
44		a9				F15				a10	a11	a12		a4	a5	a7																
45	F4									F4				F4	F4	F4										F4						
46		a9								a10	a11	a12		a4	a5	a7																
47						a70																										
48		a71																														
49										a72																						
50				a73															a66													
51										a75																						
52																																
53				a77						a76																						
54				a78																												
55	F20	F20				F20				F20	F20	F20		F20	F20	F20										F20						
56		a35	a36																					a37	a38							
57		a35	a36																					a37	a38							
58		a35	a36																					a37	a38							
59		a35	a36																					a37	a38							
60		a35	a36																					a37	a38							
61		a35	a36																					a37	a38							
62		a35	a36																					a37	a38							
63				F41		a86																		a37	a38							
64																			a56													
65		a9		F4																												
66	F23	F23				F23				F23	F23	F23		F23	F23	F23										F23						
67																																
68						F11																										
69						a89																										
70																																
71				F15						a91																						
72	F16	F16				F16				a92																F16						
73																																
74				F41																												
75		a35	a36																					a37	a38							
76	F11	F11				F11				F11	F11	F11		F11	F11	F11									F11							
77										a94																						
78										a95																						
79				F15						F20									F20		a97											
80				F15						F31									F31	F31	a98	a99										
81				F31						F31	F31	F31		F31	F31	F31			F31	F31	F31	F31	a60	a61								
82				F31						F34	F34	F34		F34	F34	F34			F34	F34	F34	F34	a60	a61								
83				F31						F36	F36	F36		F36	F36	F36			F36	F36	F36	F36	a61	a61								
84				F31						F37	F37	F37		F37	F37	F37			F37	F37	F37	F37	a61	a61								
85			a96																													
86				F41						F40									F41	F41	F41	F41	F41	F41								
87	F22	F22				F22				F22	F22	F22		F22	F22	F22										F22						
88			a97																													
89	F4	F4								F4	F4	F4		F4	F4	F4										F4						
90				F15																												
91																																
92	F17	F17				F17				F17	F17	F17		F17	F17	F17										F17						
93																			a56													
94	F18	F18				F18				F18	F18	F18		F18	F18	F18										F18						
95	F18	F18				F18				F18	F18	F18		F18	F18	F18										F18						
96																																

[illegible]

[illegible]

Justificación de LR(1)

Debido a que en la tabla acción solo existe una opción por celda, ya sea desplazar(S), reducir(R) o aceptar (ACC), podemos asegurar que nuestra gramática es LR(1).

Además, se puede justificar, a partir del análisis de la colección canónica

Calculo de tablas

Debido a la inmensa cantidad de iteraciones del algoritmo para el desarrollo de tablas, hemos optado por usar una herramienta online:

<http://jsmachines.sourceforge.net/machines/slr.html>

La gramática para usarla con el programa:

```
Z -> P

P -> B P
P -> F P
P -> ''

F -> I J G
I -> function H identificador
J -> abrirParentesis A cerrarParentesis
G -> abrirCorchete C cerrarCorchete
H -> T
H -> ''
A -> T identificador K
A -> ''
K -> coma T identificador K
K -> ''
C -> B C
C -> ''

S -> identificador asignacion E puntoYcoma
S -> identificador abrirParentesis L cerrarParentesis puntoYcoma
S -> alert abrirParentesis E cerrarParentesis puntoYcoma
S -> input abrirParentesis identificador cerrarParentesis puntoYcoma
S -> return X puntoYcoma
S -> identificador restaAsignacion E puntoYcoma

B -> if abrirParentesis E cerrarParentesis S
B -> let T identificador puntoYcoma
B -> S
B -> do abrirCorchete C cerrarCorchete while abrirParentesis E
cerrarParentesis puntoYcoma

T -> number
T -> boolean
T -> string

E -> E opLogico2 R
E -> R
R -> R opLogico1 U
R -> U
U -> U opRelacional1 V
U -> U opRelacional2 V
U -> V
V -> V opAritmetico1 W
V -> V opAritmetico2 W
V -> W
W -> identificador
```



```

W -> abrirParentesis E cerrarParentesis
W -> identificador abrirParentesis L cerrarParentesis
W -> cteEntera
W -> cadena

L -> E Q
L -> ' '
Q -> coma E Q
Q -> ' '
X -> E
X -> ' '

```

Únicamente se ha usado para obtener las tablas. Para la extracción de estas tablas, se ha procedido a exportarlas a un excel, a partir del cual, se ha desarrollado un Script en Python que genere el código necesario para importarlo a Java

Analizador Semántico

Aumentada

Z → P

```

CrearTS() = TSGL
Despl = 0

```

Inicio

P → BP

```

If(B.hasReturn){
    Error(Return Fuera de funcion)
}

```

P → FP

```

P.atributos = P.atributos

```

P → λ

Funciones

F → I J G

```

If(G.hasReturn && I.tipo==G.returnType || !G.hasReturn && I.tipo =
Empty){
    insertarTipoTS(I.idFuncion, J.tipo( I.tipo))
}else{
    Error(Tipo de retorno y return no equiparan)
}
DestruirTS()
TSActual = TSG

```

I → function H id

```

TSActual = CrearTs
Despl = 0
InsertarEti(id.pos, genEti())
I.tipo = H.tipo
I.idFuncion = id.pos

```

J → (A)

```

J.tipo = A.tipo

```

G → {C}

```
G.atributos = C.atributos
```

H → T

```
1. H.tipo = T.tipo
```

H → λ

```
1. H.tipo = EMPTY
```

A → T id K

```
insertarTipoTSActual(id.pos, T.tipo)
A.tipo = T.tipo x K.tipo
```

A → λ

```
A.tipo = Empty
```

K → , T id K

```
insertarTipoTSActual(id.pos, T.tipo)
A.tipo = T.tipo x K.tipo
```

K → λ

```
K.tipo = Empty
```

C → BC

```
C.atributos = merge(B.atributos, C.atributos)
// la funcion merge, combina atributos dando prioridad a quien tenga
el return y el tipoError
```

C → λ

```
1. C.tipo = tipoOk
```

Sentencias:

Simples

S → id = E;

```
tipoId = BuscaTipoTs(id.pos)
if(tipoId == Empty){
    insertarTipoDesplTSGlobal (id.pos, Entero ,1)
    tipoId = Entero
}
If(E.tipo == tipoId){
    S.tipo = tipoOk
}else{
    S.tipo = tipoError
    Error(Asignacion incorrecta)
}
```

S → id (L);

```
If(isFuncion(id.pos)){
If(comprobarArg(id.pos, L.tipos)){
    S.tipo = tipoOK
}else{
    S.tipo = tipoError
}
```

```

        Error(Argumentos no correctos)
    }
}else{
    Error(Función No declarada)
}

```

S → alert (E);

```

If(E.tipo ∈ {String, Entero}) {
    S.tipo = tipoOk
}else{
    S.tipo = tipoError
    Error(Alert mal construido)
}

```

S → input (id);

```

tipoId = BuscaTipoTs(id.pos)
if(tipoId == Empty){
    insertarTipoDesplTSGlobal (id.pos, Entero ,1)
    tipoId = Entero
}

If(tipoId ∈ {String, Entero}){
    S.tipo = tipoOk
}else{
    S.tipo = tipoError
    Error(input mal construido)
}

```

S → return X;

```

S.returnType = X.tipo
S.hasReturn = True

```

S → id -= E;

```

tipoId = BuscaTipoTs(id.pos)
if(tipoId == Empty){
    insertarTipoDesplTSGlobal (id.pos, Entero ,1)
    tipoId = Entero
}
If(E.tipo == Entero && tipoId==Entero){
    S.tipo = tipoOk
}else{
    S.tipo = tipoError
    Error(Asignacion Con resta incorrecta)
}

```

Compuestas

B → if (E) S

```

If(E.tipo == Logico && S.tipo == tipoOk)
    B.tipo = tipoOk
Else{
    B.tipo = tipoError
    Error(Construccion del If incorrecta)
}

```

B → let T id;

```

tipoID = BuscaTipoTS(id.pos)
If(tipoID == Empty){
    insertarTipoTS(id.pos, T.tipo)
    B.tipo = tipoOk
}

```

```

}else{
    B.tipo = tipoError
    Error("Tipo del <id> ya declarado")
}

```

B → S

```

B.atributos = S.Atributos
// ".atributos" denota todos los atributos que tenga

```

B → do { C } while (E);

```

If(C.tipo == tipoOK && E.tipo == Logico){
    B.atributos = C.atributos
}else{
    B.tipo = tipoError
}
// ".atributos" denota todos los atributos que tenga

```

T → number

```

T.tipo = Entero
T.ancho = 1

```

T → boolean

```

T.tipo = Logico
T.ancho = 1

```

T → String

```

T.tipo = String
T.ancho = 64

```

Expresiones

E → E || R

```

If(E1.tipo == R.tipo == logico){
    E.tipo = logico
}else{
    E.tipo = tipoError
}

```

E → R

```

E.tipo = R.tipo

```

R → R && U

```

If(R1.tipo==U.tipo == Logico){
    R.tipo = logico
}else{
    R.tipo = tipoError
    Error(AND incorrecto)
}

```

R → U

```

R.tipo = U.tipo

```

U → U == V

```

If(U1.tipo = V.Tipo == Entero){
    U.tipo = logico
}else{
    U.tipo = tipoError
}

```

```

        Error(Igualdad incorrecta)
    }

```

$U \rightarrow U \neq V$

```

If(U1.tipo = V.Tipo = Entero){
    U.tipo = logico
}else{
    U.tipo = tipoError
    Error(NegacionIgualdad Incorrectp)
}

```

$U \rightarrow V$

```

U.tipo = V.tipo

```

$V \rightarrow V + W$

```

If(V.tipo == W.tipo == Entero){
    V.tipo = Entero
}else{
    V.tipo = tipoError
    Error(Suma incorrecta)
}

```

$V \rightarrow V - W$

```

If(V.tipo == W.tipo == Entero){
    V.tipo = Entero
}else{
    V.tipo = tipoError
    Error(Resta incorrecta)
}

```

$V \rightarrow W$

```

V.tipo = W.tipo

```

$W \rightarrow id$

```

tipoId = BuscaTipoTs(id.pos)
if(tipoId == Empty){
    W.tipo = Entero
    insertarTipoDesplTSGlobal(id.pos, entero,1)
}else{
    W.tipo = tipoId
}

```

$W \rightarrow (E)$

```

W.tipo = E.tipo

```

$W \rightarrow id(L)$

```

If(isFuncion(id.pos)){
    If(buscaTipoTS(id.pos) = L.tipos(t){
        W.tipo = t
    }else{
        W.tipo = tipoError
        Error(Argumentos no correctos)
    }
}else{
    Error(Funcion No declarada)
}

```

$W \rightarrow entero$

```
W.tipo = Entero
```

$W \rightarrow \text{cadena}$

```
W.tipo = String
```

Argumentos de la función

$L \rightarrow E Q$

```
L.tipo = E.tipo x Q.tipo
```

$L \rightarrow \lambda$

```
L.tipo = Empty
```

$Q \rightarrow , E Q$

```
Q.tipo = E.tipo x Q1.tipo
```

$Q \rightarrow \lambda$

```
Q.tipo = Empty
```

Return

$X \rightarrow E$

```
X.tipo = E.tipo
```

$X \rightarrow \lambda$

```
X.tipo = Empty
```

Tipos Usados

Los desplazamientos de cada variable, se ponen de manera automática en la tabla, de ese modo nos ahorramos una operación en las reglas semánticas

Tipo	Desplazamiento
NUMBER	1
STRING	64
BOOLEAN	1
FUNCTION	0
EMPTY	0
tipoOk	0
tipoError	0

Observaciones

Cabria destacar, que cuando se llama a la función InsertarTipo, se introduce directamente el ancho y se mueve el desplazamiento.

Errores

0	Error Lexico: Caracter No valido
1	Error Lexico: Ya esta en la tabla de simbolos
-1	Errores de implementación de compilador
2	Error Lexico-Semantico: Entero fuera de rango
3	Error Semantico: <msg>
4	Error Lexico-Semantico: Nombre de variable fuera de rango
5	Error Sintáctico: <msg>

Dependiendo del error cometido, se ha decidido mandar un mensaje – observación – del error más detallado, véase los ejemplos del anexo. Con la intención de hacer llegar al usuario cuanta más información de cara a evitar el error

Ejemplos de errores específicos

A continuación, mostramos algunos ejemplos de errores personalizados de cara a que el usuario pueda corregirlo

```
Error Semantico:
Alert tiene que tener como argumento una cadena o numero
@Usuario:
      (Contruccion Correcta) >> alert(<cadena|numero>);
      (Su construccion) >> alert(<'BOOLEAN'>);
```

```
Error Semantico:
Nombre de identificador: "b" ,ya declarado previamente
```

```
Error Sintactico en la linea:2
@Usuario: Se esperaba: ;
```

Código para VAST

Incluimos el código usado para Vast

```

Terminales = { function identificador cerrarParentesis
abrirParentesis  cerrarCorchete  abrirCorchete coma asignacion
puntoYcoma alert input return restaAsignacion if let do while number
boolean string opLogico2 opLogico1 opRelacional1 opRelacional2
opAritmetico1 opAritmetico2 cteEntera cadena }

NoTerminales = { Z P F I J G H A K C S B T E R U V W L Q X }

Axioma = Z
Producciones = {
    Z -> P

    P -> B P
    P -> F P
    P -> lambda

    F -> I J G
    I -> function H identificador
    J -> abrirParentesis A cerrarParentesis
    G -> abrirCorchete C cerrarCorchete
    H -> T
    H -> lambda
    A -> T identificador K
    A -> lambda
    K -> coma T identificador K
    K -> lambda
    C -> B C
    C -> lambda

    S -> identificador asignacion E puntoYcoma
    S -> identificador abrirParentesis L cerrarParentesis puntoYcoma
    S -> alert abrirParentesis E cerrarParentesis puntoYcoma
    S -> input abrirParentesis identificador cerrarParentesis
puntoYcoma
    S -> return X puntoYcoma
    S -> identificador restaAsignacion E puntoYcoma

    B -> if abrirParentesis E cerrarParentesis S
    B -> let T identificador puntoYcoma
    B -> S
    B -> do abrirCorchete C cerrarCorchete while abrirParentesis E
cerrarParentesis puntoYcoma

    T -> number
    T -> boolean
    T -> string

    E -> E opLogico2 R
    E -> R
    R -> R opLogico1 U
    R -> U
    U -> U opRelacional1 V
    U -> U opRelacional2 V
    U -> V
    V -> V opAritmetico1 W
    V -> V opAritmetico2 W
    V -> W

```



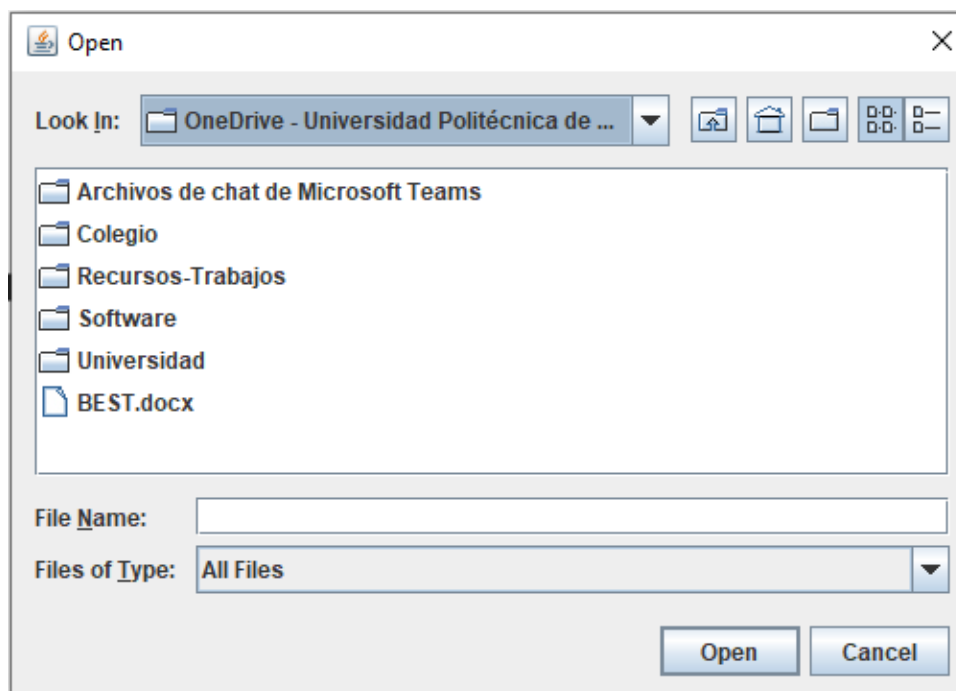
```

W -> identificador
W -> abrirParentesis E cerrarParentesis
W -> identificador abrirParentesis L cerrarParentesis
W -> cteEntera
W -> cadena

L -> E Q
L -> lambda
Q -> coma E Q
Q -> lambda
X -> E
X -> lambda
}

```

Mini-Demo Visual



TSGLOBAL #1

LEXEMA	Tipo	Despl	numParam	TipoRetorno	EtqFuncion	TipoParam1
cadena	'STRING'	0				
logico1	'BOOLEAN'	64				
logico2	'BOOLEAN'	65				
number2	'NUMBER'	66				
number1	'NUMBER'	67				

erroresDeAnálisis.txt
 logSemantico.txt
 Parse.txt
 tablaSimbolos.txt
 tokens.txt

El programa se puede encontrar(Junto al código) en
<https://github.com/fbellidopazos/ProcesadoresLenguajes>

Webgrafía

Agradecimientos especiales a las siguientes fuentes de información

- Clase Pair: <https://www.techiedelight.com/implement-pair-class-java/>
- Oracle "SimpleTableDemo.java":
<https://docs.oracle.com/javase/tutorial/uiswing/examples/components/SimpleTableDemoProject/src/components/SimpleTableDemo.java>
- Página web que nos genera las tablas:
<http://jsmachines.sourceforge.net/machines/slr.html>
- Información de JavaScriptPDL
<https://dlsiisv.fi.upm.es/procesadores/IntroJavaScript.html>
- Tests para la autocorrección
https://dlsiisv.fi.upm.es/draco/Registro_login/login.php

Anexo

Los arboles generados por el parse de aquellas pruebas correctas, se encontraran al final, su lectura es de izquierda a derecha y arriba abajo (por ocupar menos espacio), de todas maneras se adjunta el parse y el código usado para VAST.

Prueba 1

Código

```
1. /* Prueba Correcta */
2. /*
3. Sentencias Simples
4. */
5. let number n;
6. let boolean b;
7. let string s;
8.
9. if(b) n=1;
10. alert(s);
11. input(n);
12.
13. n-=2;
```

Tokens

```
<let, >
<number, >
<identificador, 0>
<puntoYcoma, >
<let, >
<boolean, >
<identificador, 1>
<puntoYcoma, >
<let, >
<string, >
<identificador, 2>
<puntoYcoma, >
<if, >
<abrirParentesis, >
<identificador, 3>
<cerrarParentesis, >
<identificador, 4>
<asignacion, >
<cteEntera, 1>
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<identificador, 5>
<cerrarParentesis, >
<puntoYcoma, >
<input, >
<abrirParentesis, >
<identificador, 6>
<cerrarParentesis, >
<puntoYcoma, >
<identificador, 7>
<restaAsignacion, >
<cteEntera, 2>
```

```
<puntoYcoma, >  
<EOF, >
```

Parse

Ascendente 27 24 28 24 29 24 40 39 36 33 31 43 39 36 33 31 17 23 40 39 36 33 31 19 25
20 25 43 39 36 33 31 22 25 4 2 2 2 2 2 2 1

Tabla de Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```
* LEXEMA : 'n'  
ATRIBUTOS:  
+ Tipo : 'NUMBER'  
+ Despl : 0
```

```
* LEXEMA : 'b'  
ATRIBUTOS:  
+ Tipo : 'BOOLEAN'  
+ Despl : 1
```

```
* LEXEMA : 's'  
ATRIBUTOS:  
+ Tipo : 'STRING'  
+ Despl : 2
```

Errores de Análisis

- Ninguno

Prueba 2

Código

```
1. /* Prueba Correcta */  
2.  
3. let number a;  
4. let number b;  
5. let boolean c;  
6. do{  
7.     if(c) a-=1;  
8. }while( (a+b) == 1 && (a-b) != 4 || c );
```

Tokens

```
<let, >  
<number, >  
<identificador, 0>  
<puntoYcoma, >  
<let, >  
<number, >  
<identificador, 1>  
<puntoYcoma, >  
<let, >  
<boolean, >  
<identificador, 2>  
<puntoYcoma, >  
<do, >  
<abrirCorchete, >  
<if, >  
<abrirParentesis, >
```

```

<identificador, 3>
<cerrarParentesis, >
<identificador, 4>
<restaAsignacion, >
<cteEntera, 1>
<puntoYcoma, >
<cerrarCorchete, >
<while, >
<abrirParentesis, >
<abrirParentesis, >
<identificador, 5>
<opAritmetico, 1>
<identificador, 6>
<cerrarParentesis, >
<opRelacional, 1>
<cteEntera, 1>
<opLogico, 1>
<abrirParentesis, >
<identificador, 7>
<opAritmetico, 2>
<identificador, 8>
<cerrarParentesis, >
<opRelacional, 2>
<cteEntera, 4>
<opLogico, 2>
<identificador, 9>
<cerrarParentesis, >
<puntoYcoma, >
<EOF, >

```

Parse

Ascendente 27 24 27 24 28 24 40 39 36 33 31 43 39 36 33 31 22 23 16 15 40 39 40 37 36
 33 31 41 39 36 43 39 34 33 40 39 40 38 36 33 31 41 39 36 43 39 35 32 31 40 39 36 33 30
 26 4 2 2 2 2 1

Tabla de Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```

* LEXEMA : 'a'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 0

```

```

* LEXEMA : 'b'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 1

```

```

* LEXEMA : 'c'
ATRIBUTOS:
+ Tipo : 'BOOLEAN'
+ Despl : 2

```

Errores de Análisis

- Ninguno

Prueba 3

Código

```
1. /* Prueba Correcta */
2.
3. function print (string s, string msg, number f)
4. {
5.     alert (s); alert (msg); alert (f);
6.
7. }
8.
9.
10. function number Factorial (number n)
11. {
12.     if (n != 0)     return 1;
13.     return n + Factorial (n - 1);
14. }
15.
16. print("Factorial","de 6:" ,Factorial(6));
```

Tokens

```
<function, >
<identificador, 0>
<abrirParentesis, >
<string, >
<identificador, 1>
<coma, >
<string, >
<identificador, 2>
<coma, >
<number, >
<identificador, 3>
<cerrarParentesis, >
<abrirCorchete, >
<alert, >
<abrirParentesis, >
<identificador, 4>
<cerrarParentesis, >
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<identificador, 5>
<cerrarParentesis, >
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<identificador, 6>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<function, >
<number, >
<identificador, 7>
<abrirParentesis, >
<number, >
<identificador, 8>
<cerrarParentesis, >
<abrirCorchete, >
<if, >
```

```

<abrirParentesis, >
<identificador, 9>
<opRelacional, 2>
<cteEntera, 0>
<cerrarParentesis, >
<return, >
<cteEntera, 1>
<puntoYcoma, >
<return, >
<identificador, 10>
<opAritmetico, 1>
<identificador, 11>
<abrirParentesis, >
<identificador, 12>
<opAritmetico, 2>
<cteEntera, 1>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<identificador, 13>
<abrirParentesis, >
<cadena, "Factorial">
<coma, >
<cadena, "de 6:">
<coma, >
<identificador, 14>
<abrirParentesis, >
<cteEntera, 6>
<cerrarParentesis, >
<cerrarParentesis, >
<puntoYcoma, >
<EOF, >

```

Parse

Ascendente 10 6 29 29 27 14 13 13 11 7 40 39 36 33 31 19 25 40 39 36 33 31 19 25 40 39
 36 33 31 19 25 16 15 15 15 8 5 27 9 6 27 14 11 7 40 39 36 43 39 35 33 31 43 39 36 33 31
 49 21 23 40 39 40 39 43 38 36 33 31 48 45 42 37 36 33 31 49 21 25 16 15 15 8 5 44 39 36
 33 31 44 39 36 33 31 43 39 36 33 31 48 45 42 39 36 33 31 48 47 47 45 18 25 4 2 3 3 1

Tabla de Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```

* LEXEMA : 'print'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 3
+ TipoRetorno : 'EMPTY'
+ EtiqFuncion : 'eti0'
+ TipoParam1 : 'STRING'
+ TipoParam2 : 'STRING'
+ TipoParam3 : 'NUMBER'

```

```

-----
* LEXEMA : 'Factorial'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 1
+ TipoRetorno : 'NUMBER'
+ EtiqFuncion : 'eti1'
+ TipoParam1 : 'NUMBER'
-----

```

CONTENIDO DE LA TABLA FUNCION `print` #2 :

```
* LEXEMA : 'f'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 0
```

```
-----
* LEXEMA : 'msg'
ATRIBUTOS:
+ Tipo : 'STRING'
+ Despl : 1
```

```
-----
* LEXEMA : 's'
ATRIBUTOS:
+ Tipo : 'STRING'
+ Despl : 65
-----
```

CONTENIDO DE LA TABLA FUNCION `Factorial` #3 :

```
* LEXEMA : 'n'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 0
-----
```

Errores de Análisis

- Ninguno

Prueba 4

Código

```
1. /* Programa Correcto - Ejemplo DRACO */
2.
3. let string texto;
4. function pideTexto ()
5. {
6.   alert ("Introduce un texto");
7.   input (texto);
8. }
9. function print (string msg)
10. {
11.     alert (msg);
12. }
13. pideTexto();
14. print (texto);
```

Tokens

```
<let, >
<string, >
<identificador, 0>
<puntoYcoma, >
<function, >
<identificador, 1>
<abrirParentesis, >
<cerrarParentesis, >
<abrirCorchete, >
<alert, >
<abrirParentesis, >
<cadena, "Introduce un texto">
```



```

<cerrarParentesis, >
<puntoYcoma, >
<input, >
<abrirParentesis, >
<identificador, 2>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<function, >
<identificador, 3>
<abrirParentesis, >
<string, >
<identificador, 4>
<cerrarParentesis, >
<abrirCorchete, >
<alert, >
<abrirParentesis, >
<identificador, 5>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<identificador, 6>
<abrirParentesis, >
<cerrarParentesis, >
<puntoYcoma, >
<identificador, 7>
<abrirParentesis, >
<identificador, 8>
<cerrarParentesis, >
<puntoYcoma, >
<EOF, >

```

Parse

Ascendente 29 24 10 6 12 7 44 39 36 33 31 19 25 20 25 16 15 15 8 5 10 6 29 14 11 7 40
 39 36 33 31 19 25 16 15 8 5 46 18 25 40 39 36 33 31 48 45 18 25 4 2 2 3 3 2 1

Tabla de Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```

* LEXEMA : 'texto'
ATRIBUTOS:
+ Tipo : 'STRING'
+ Despl : 0

```

```

* LEXEMA : 'pideTexto'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 0
+ TipoRetorno : 'EMPTY'
+ EtiqFuncion : 'eti1'

```

```

* LEXEMA : 'print'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 1
+ TipoRetorno : 'EMPTY'
+ EtiqFuncion : 'eti2'
+ TipoParam1 : 'STRING'

```

CONTENIDO DE LA TABLA FUNCION pideTexto #2 :

CONTENIDO DE LA TABLA FUNCION `print` #3 :

```
* LEXEMA : 'msg'
ATRIBUTOS:
+ Tipo : 'STRING'
+ Despl : 0
```

Errores de Análisis

- Ninguno

Prueba 5 – Prueba Completa

Código

```
1. /* prueba correcta - Ejemplo DRACO*/
2.
3. let number entero;
4. let string cadena;
5. let boolean logico;
6. let number abcdefghij_____ ;
7. function FuncionNumero ()
8. {
9.   let number i;
10.
11.   i = entero;
12.   entero=9-i;
13. }
14. function string FuncionRetorno (string hola, number x, boolean
    z)
15. {
16.   let number i;
17.
18.   input (i);
19.   alert (hola);
20.
21.   return hola;
22. }
23. function FuncionSentencia (number z_Z, boolean b)
24. {
25.   do
26.   {
27.     alert (88);
28.   } while (b);
29.   FuncionSentencia(z_Z,b);
30. }
31. function Funcion (number x, boolean b)
32. {
33.   FuncionNumero();
34.   alert
35.   (cadena);return;
36. }
37. let number iii;
38. let boolean bbb;
39. /* comentario
40. multi
41. línea ***/
42. input(iii);
43. alert (iii) ;
44.
45. iii -= iii;
```

```
46. bbb = bbb;  
47.  
48. Funcion  
49. ( iii,  
50.     bbb );
```

Tokens

```
<let, >  
<number, >  
<identificador, 0>  
<puntoYcoma, >  
<let, >  
<string, >  
<identificador, 1>  
<puntoYcoma, >  
<let, >  
<boolean, >  
<identificador, 2>  
<puntoYcoma, >  
<let, >  
<number, >  
<identificador, 3>  
<puntoYcoma, >  
<function, >  
<identificador, 4>  
<abrirParentesis, >  
<cerrarParentesis, >  
<abrirCorchete, >  
<let, >  
<number, >  
<identificador, 5>  
<puntoYcoma, >  
<identificador, 6>  
<asignacion, >  
<identificador, 7>  
<puntoYcoma, >  
<identificador, 8>  
<asignacion, >  
<cteEntera, 9>  
<opAritmetico, 2>  
<identificador, 9>  
<puntoYcoma, >  
<cerrarCorchete, >  
<function, >  
<string, >  
<identificador, 10>  
<abrirParentesis, >  
<string, >  
<identificador, 11>  
<coma, >  
<number, >  
<identificador, 12>  
<coma, >  
<boolean, >  
<identificador, 13>  
<cerrarParentesis, >  
<abrirCorchete, >  
<let, >  
<number, >  
<identificador, 14>
```

```
<puntoYcoma, >
<input, >
<abrirParentesis, >
<identificador, 15>
<cerrarParentesis, >
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<identificador, 16>
<cerrarParentesis, >
<puntoYcoma, >
<return, >
<identificador, 17>
<puntoYcoma, >
<cerrarCorchete, >
<function, >
<identificador, 18>
<abrirParentesis, >
<number, >
<identificador, 19>
<coma, >
<boolean, >
<identificador, 20>
<cerrarParentesis, >
<abrirCorchete, >
<do, >
<abrirCorchete, >
<alert, >
<abrirParentesis, >
<cteEntera, 88>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<while, >
<abrirParentesis, >
<identificador, 21>
<cerrarParentesis, >
<puntoYcoma, >
<identificador, 22>
<abrirParentesis, >
<identificador, 23>
<coma, >
<identificador, 24>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<function, >
<identificador, 25>
<abrirParentesis, >
<number, >
<identificador, 26>
<coma, >
<boolean, >
<identificador, 27>
<cerrarParentesis, >
<abrirCorchete, >
<identificador, 28>
<abrirParentesis, >
<cerrarParentesis, >
<puntoYcoma, >
<alert, >
```

```

<abrirParentesis, >
<identificador, 29>
<cerrarParentesis, >
<puntoYcoma, >
<return, >
<puntoYcoma, >
<cerrarCorchete, >
<let, >
<number, >
<identificador, 30>
<puntoYcoma, >
<let, >
<boolean, >
<identificador, 31>
<puntoYcoma, >
<input, >
<abrirParentesis, >
<identificador, 32>
<cerrarParentesis, >
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<identificador, 33>
<cerrarParentesis, >
<puntoYcoma, >
<identificador, 34>
<restaAsignacion, >
<identificador, 35>
<puntoYcoma, >
<identificador, 36>
<asignacion, >
<identificador, 37>
<puntoYcoma, >
<identificador, 38>
<abrirParentesis, >
<identificador, 39>
<coma, >
<identificador, 40>
<cerrarParentesis, >
<puntoYcoma, >
<EOF, >

```

Parse

Ascendente 27 24 29 24 28 24 27 24 10 6 12 7 27 24 40 39 36 33 31 17 25 43 39 40 38 36
 33 31 17 25 16 15 15 15 8 5 29 9 6 29 27 28 14 13 13 11 7 27 24 20 25 40 39 36 33 31 19
 25 40 39 36 33 31 49 21 25 16 15 15 15 15 8 5 10 6 27 28 14 13 11 7 43 39 36 33 31 19
 25 16 15 40 39 36 33 31 26 40 39 36 33 31 40 39 36 33 31 48 47 45 18 25 16 15 15 8 5 10
 6 27 28 14 13 11 7 46 18 25 40 39 36 33 31 19 25 50 21 25 16 15 15 15 8 5 27 24 28 24
 20 25 40 39 36 33 31 19 25 40 39 36 33 31 22 25 40 39 36 33 31 17 25 40 39 36 33 31 40
 39 36 33 31 48 47 45 18 25 4 2 2 2 2 2 2 2 3 3 3 2 2 2 1

Tabla de Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

* LEXEMA : 'entero'

ATRIBUTOS:

+ Tipo : 'NUMBER'

+ Despl : 0

 * LEXEMA : 'cadena'

ATRIBUTOS:
+ Tipo : 'STRING'
+ Despl : 1

* LEXEMA : 'logico'
ATRIBUTOS:
+ Tipo : 'BOOLEAN'
+ Despl : 65

* LEXEMA : 'abcdefghij_____'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 66

* LEXEMA : 'FuncionNumero'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 0
+ TipoRetorno : 'EMPTY'
+ EtiqFuncion : 'eti4'

* LEXEMA : 'FuncionRetorno'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 3
+ TipoRetorno : 'STRING'
+ EtiqFuncion : 'eti5'
+ TipoParam1 : 'STRING'
+ TipoParam2 : 'NUMBER'
+ TipoParam3 : 'BOOLEAN'

* LEXEMA : 'FuncionSentencia'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 2
+ TipoRetorno : 'EMPTY'
+ EtiqFuncion : 'eti6'
+ TipoParam1 : 'NUMBER'
+ TipoParam2 : 'BOOLEAN'

* LEXEMA : 'Funcion'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 2
+ TipoRetorno : 'EMPTY'
+ EtiqFuncion : 'eti7'
+ TipoParam1 : 'NUMBER'
+ TipoParam2 : 'BOOLEAN'

* LEXEMA : 'iii'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 67

* LEXEMA : 'bbb'
ATRIBUTOS:
+ Tipo : 'BOOLEAN'
+ Despl : 68

CONTENIDO DE LA TABLA FUNCION `FuncionNumero #2` :

```
* LEXEMA : 'i'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 0
```

CONTENIDO DE LA TABLA FUNCION `FuncionRetorno #3` :

```
* LEXEMA : 'z'
ATRIBUTOS:
+ Tipo : 'BOOLEAN'
+ Despl : 0
```

```
* LEXEMA : 'x'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 1
```

```
* LEXEMA : 'hola'
ATRIBUTOS:
+ Tipo : 'STRING'
+ Despl : 2
```

```
* LEXEMA : 'i'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 66
```

CONTENIDO DE LA TABLA FUNCION `FuncionSentencia #4` :

```
* LEXEMA : 'b'
ATRIBUTOS:
+ Tipo : 'BOOLEAN'
+ Despl : 0
```

```
* LEXEMA : 'z_z'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 1
```

CONTENIDO DE LA TABLA FUNCION `Funcion #5` :

```
* LEXEMA : 'b'
ATRIBUTOS:
+ Tipo : 'BOOLEAN'
+ Despl : 0
```

```
* LEXEMA : 'x'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 1
```

Errores de Análisis

- Ninguno

Prueba 6

El error esta en el operado *, nosotros no lo tenemos implementado

Código

```
1. /* Prueba Incorrecta - Error Lexico*/
2. function number SumaAlCuadrado (number a, number b)
3. {
4.     j= a + b;
5.     return j * j;
6.
7. }
```

Tokens

```
1. <function, >
2. <number, >
3. <identificador, 0>
4. <abrirParentesis, >
5. <number, >
6. <identificador, 1>
7. <coma, >
8. <number, >
9. <identificador, 2>
10. <cerrarParentesis, >
11. <abrirCorchete, >
12. <identificador, 3>
13. <asignacion, >
14. <identificador, 4>
15. <opAritmetico, 1>
16. <identificador, 5>
17. <puntoYcoma, >
18. <return, >
19. <identificador, 6>
20. <identificador, 7>
```

Parse

Ascendente 27 9 6 27 27 14 13 11 7 40 39 40 37 36 33 31 17 25

Tabla Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```
* LEXEMA : 'SumaAlCuadrado'
ATRIBUTOS:
+ Tipo : 'FUNCTION'
+ numParam : 2
+ TipoRetorno : 'NUMBER'
+ EtiqFuncion : 'eti0'
+ TipoParam1 : 'NUMBER'
+ TipoParam2 : 'NUMBER'
```

```
-----
* LEXEMA : 'j'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 0
-----
```

CONTENIDO DE LA TABLA FUNCION SumaAlCuadrado #2 :

```
* LEXEMA : 'b'
ATRIBUTOS:
+ Tipo : 'NUMBER'
```



```
+ Despl : 0
```

```
* LEXEMA : 'a'  
ATRIBUTOS:  
+ Tipo : 'NUMBER'  
+ Despl : 1
```

Errores de Análisis

Error Lexico:

Caracter No valido en la linea:5

Error Sintactico en la linea:5

@Usuario: Se esperaba: (,|,|,&&==,!=,+- o nada

Prueba 7

El error es la sentencia de declaración con asignación, no la hemos incluido

Código

```
1. /*Prueba incorrecta - Error Sintaxis*/  
2. let number var4 = 1+1;
```

Tokens

```
<let, >  
<number, >  
<identificador, 0>  
<asignacion, >
```

Parse

Ascendente 27

Tabla Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```
* LEXEMA : 'var4'
```

Errores de Análisis

Error Sintactico en la linea:2

@Usuario: Se esperaba: ;

Prueba 8

El error es un “return” fuera del cuerpo de función

Código

```
1. /* Prueba Incorrecta - Error Semantico*/  
2.  
3. let number b;  
4. return b;
```

Tokens

```
<let, >
<number, >
<identificador, 0>
<puntoYcoma, >
<return, >
<identificador, 1>
<puntoYcoma, >
<EOF, >
```

Parse

Ascendente 27 24 40 39 36 33 31 49 21 25 4 2 2 1

Tabla Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```
* LEXEMA : 'b'
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 0
```

Errores de Análisis

Error Semantico:

Return Fuera de funcion

Error Semantico:

Revisa los errores que tienes, algo no anda bien

Prueba 9

El error es una Re-Declaracion de variable

Código

```
/* Prueba Incorrecta */
let number b;
let boolean b;
```

Tokens

```
<let, >
<number, >
<identificador, 0>
<puntoYcoma, >
<let, >
<boolean, >
<identificador, 1>
<puntoYcoma, >
<EOF, >
```

Parse

Ascendente 27 24 28 24 4 2 2 1

Tabla Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```
* LEXEMA : 'b'
```

```
ATRIBUTOS:
+ Tipo : 'NUMBER'
+ Despl : 0
-----
```

Errores de Análisis

Error Semantico:

Nombre de identificador: "b" ,ya declarado previamente

Error Semantico:

Revisa los errores que tienes, algo no anda bien

Prueba 10

Error semántico en alert e input

Código

```
1. let boolean b;
2. alert(b);
3. input(b);
```

Tokens

```
<let, >
<boolean, >
<identificador, 0>
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<identificador, 1>
<cerrarParentesis, >
<puntoYcoma, >
<input, >
<abrirParentesis, >
<identificador, 2>
<cerrarParentesis, >
<puntoYcoma, >
<EOF, >
```

Parse

Ascendente 28 24 40 39 36 33 31 19 25 20 25 4 2 2 2 1

Tabla Símbolos

CONTENIDO DE LA TABLA TSMAIN #1 :

```
* LEXEMA : 'b'
ATRIBUTOS:
+ Tipo : 'BOOLEAN'
+ Despl : 0
-----
```

Errores de Análisis

Error Semantico:

Alert tiene que tener como argumento una cadena o numero@Usuario:

(Contruccion Correcta) >> alert(<cadena|numero>);

(Su construccion) >> alert(<'BOOLEAN'>);

Error Semantico:

input tiene que tener como argumento una cadena o un entero

@Usuario:

(Construccion Correcta) >> input(<cadena|numero>);

(Su construccion) >> input(<'BOOLEAN'>);

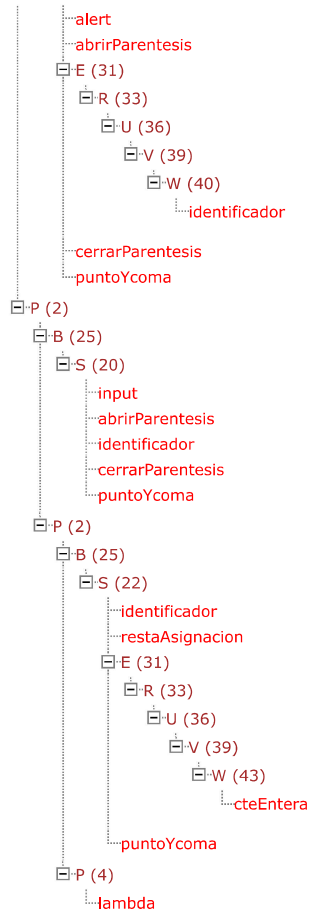
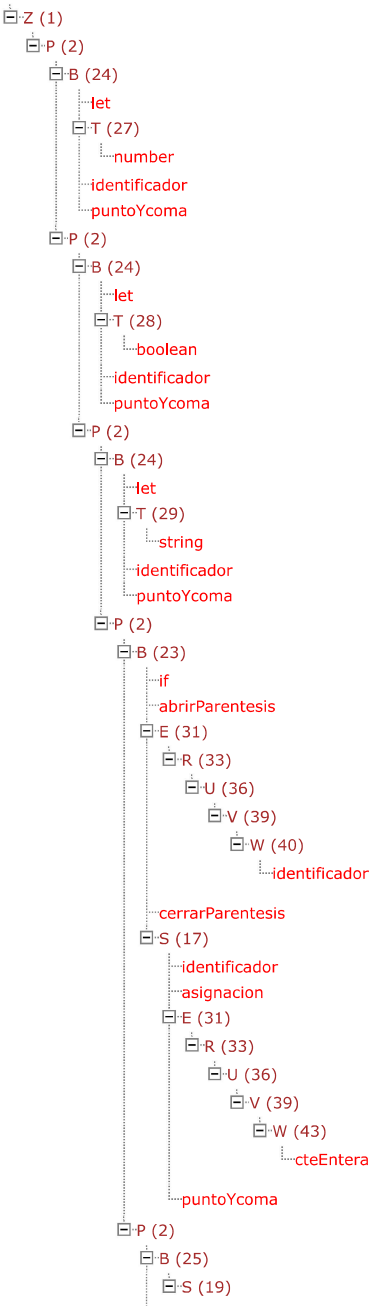
Error Semantico:

Revisa los errores que tienes, algo no anda bien

Árbol resultado de:

Gramática: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\2.Analizador Sintactico\GramaticaVast.txt

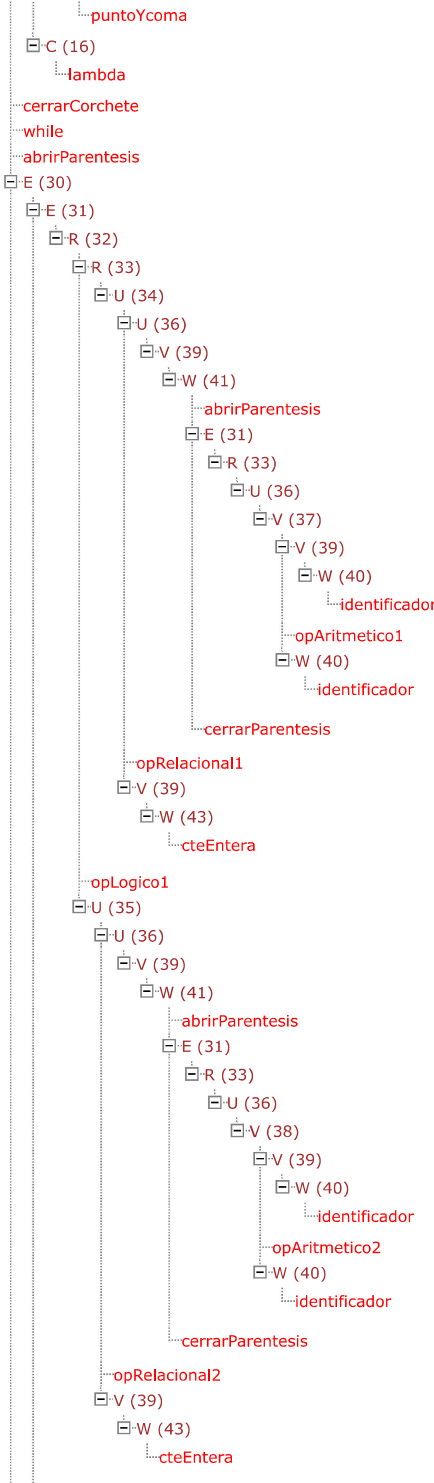
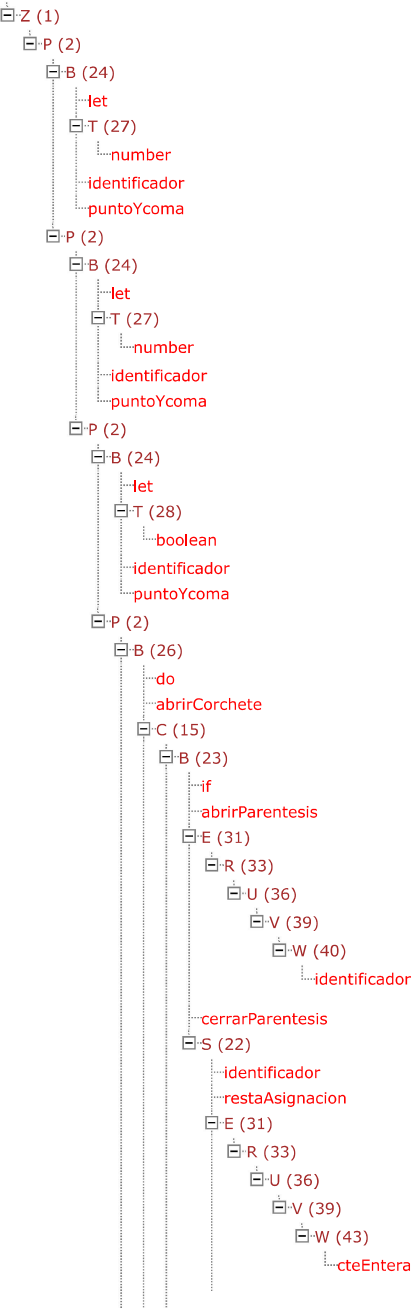
Parse: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\Anexo\allTests\Prueba 01\Parse0.txt

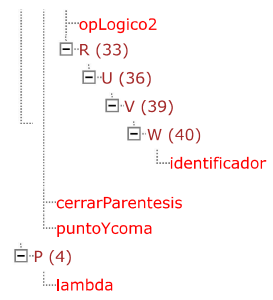


Árbol resultado de:

Gramática: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\2.Analizador Sintactico\GramaticaVast.txt

Parse: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\Anexo\allTests\Prueba 02\Parse1.txt

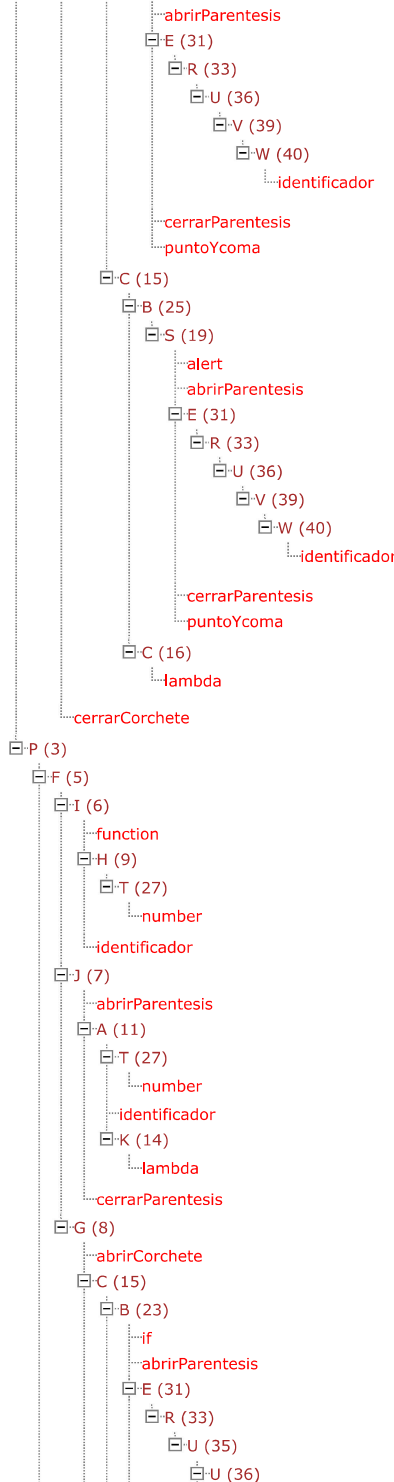
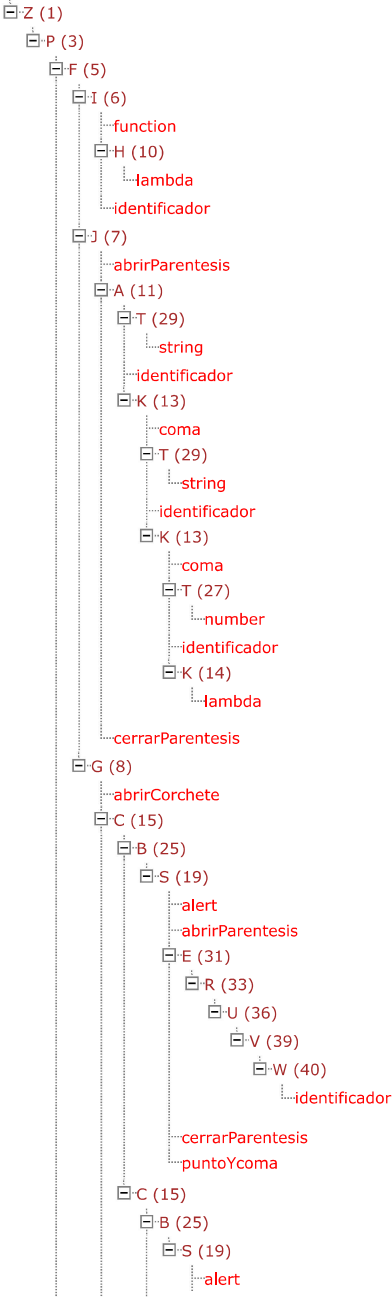




Árbol resultado de:

Gramática: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\2.Analizador Sintactico\GramaticaVast.txt

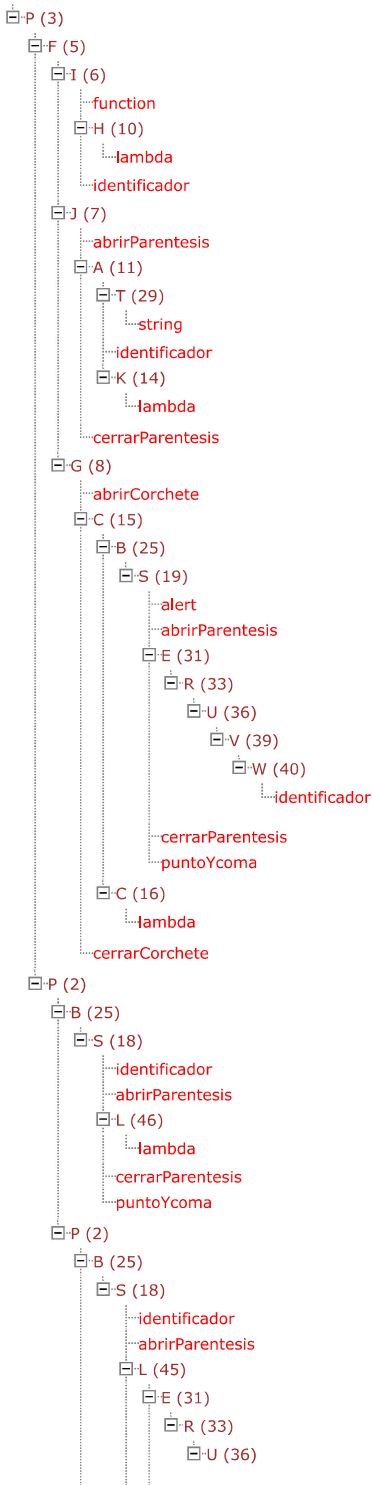
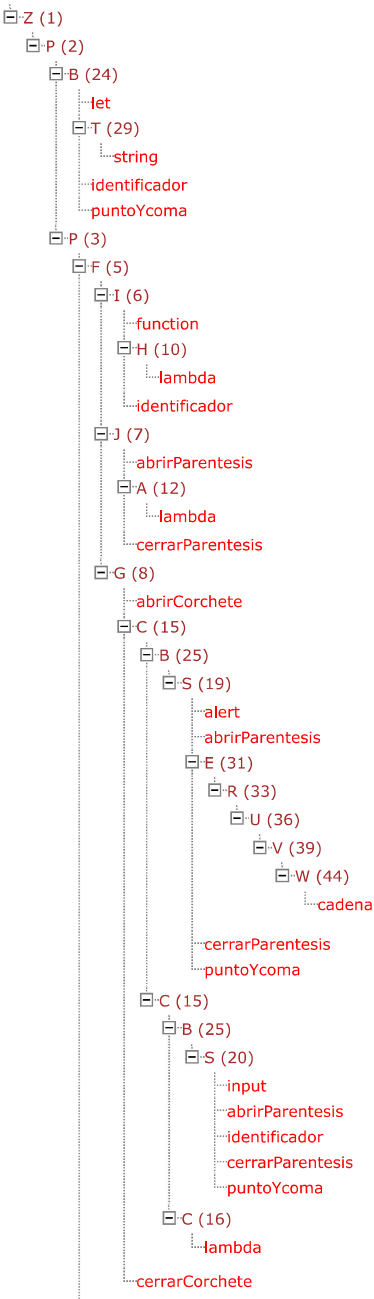
Parse: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\Anexo\allTests\Prueba 03\Parse2.txt

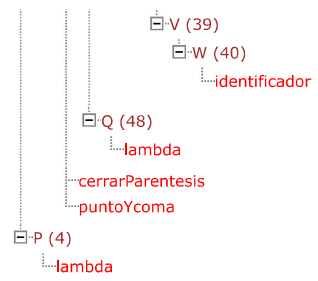


Árbol resultado de:

Gramática: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\2.Analizador Sintactico\GramaticaVast.txt

Parse: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\Anexo\allTests\Prueba 04\Parse3.txt





Árbol resultado de:

Gramática: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\2.Analizador Sintactico\GramaticaVast.txt

Parse: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\Anexo\allTests\Prueba 05\Parse4.txt

