Procesador de Lenguaje JavaScript PDL

Grupo 46

Sofía Hernández Montero
18M046

Jaime González Delgado
18M048

Fernando Bellido Pazos
18M008

UNIVERSIDAD
POLITÉCNICA
DE MADRID

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

Contenido

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

Cabria 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, posTS>	<function, ->	

Operador Artimetico	Operador Logico	Operador relacional
1: +	1: &&	1: ==
2: -	2: ||	2: !=

Gramática del Lenguaje

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

A → $dA \mid lA \mid o.c \mid \_A$

C → $c_1C \mid "$

E → $dE \mid o.c$

G→ $= \mid o.c$

I → |

J → &

N → $= \mid o.c$

Q → =

U → $*Y$

Y → $c_2Y \mid *Z$

Z → $c_3Y \mid *Z \mid /S$

Leyenda

c1 = cualquier carácter | {"}

c2= cualquier carácter | {}*

c3= cualquier carácter | {, /}*

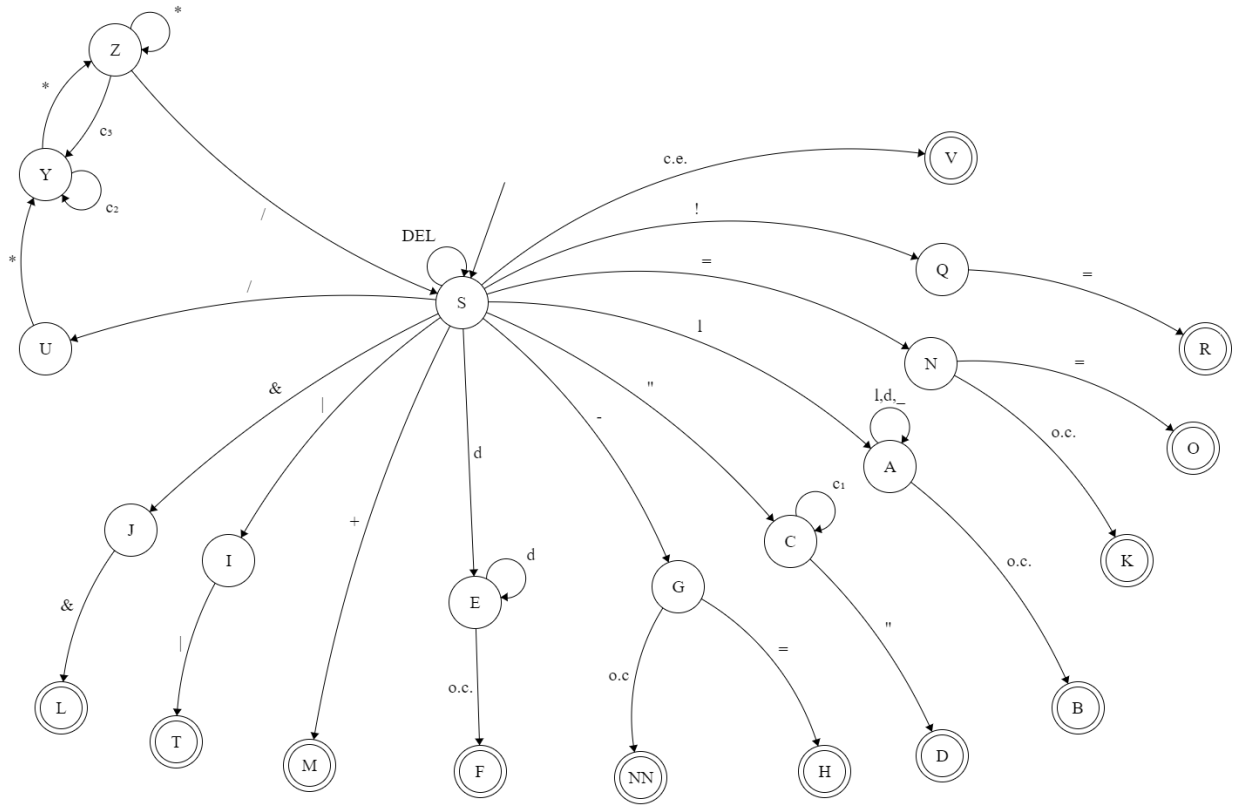
c.e = caracteres especiales: {,},(,), ; , ','

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→V: Comprobar tipo carácter especial: Enviar token correspondiente; Leer;

S→Q: leer;

Q→R: Gen_Token(<opRelacional,2>)

S→N: leer;

N→K: Gen_Token(<asignación, ->)

N→O: Gen_Token(<opRelacional,1>); leer;

S→A: lexema=c;leer; //(Siendo c el carácter leido)

A→A: lexema=lexema ⊕c; leer; //(Siendo c el carácter leído)

A→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⊕ c_1; 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	EtiqFuncion

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

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

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

$E \rightarrow E \mid\mid R \mid R$

$R \rightarrow R \&\& U \mid U$

$U \rightarrow U == V \mid U \text{ != } V \mid V$

$V \rightarrow V + W \mid V - W \mid W$

$W \rightarrow \text{id} \mid (\ E\ ) \mid \text{id}\ (\ L\ ) \mid \text{entero} \mid \text{cadena}$

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}

SLR closure table

Goto	Kernel	State	Closure
	{Z -> .P}	0	{Z -> .P; P -> .B P; P -> .F P; P -> .; B -> .if abrirParentesis E cerrarParentesis S; B -> .let T identificador puntoYcoma; B -> .S; B -> .do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma; F -> .I J G; 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; I -> .function H identificador}
goto(0, P)	{Z -> P.}	1	{Z -> P.}
goto(0, B)	{P -> B.P}	2	{P -> B.P; P -> .B P; P -> .F P; P -> .; B -> .if abrirParentesis E cerrarParentesis S; B -> .let T identificador puntoYcoma; B -> .S; B -> .do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma; F -> .I J G; 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; I -> .function H identificador}
goto(0, F)	{P -> F.P}	3	{P -> F.P; P -> .B P; P -> .F P; P -> .; B -> .if abrirParentesis E cerrarParentesis S; B -> .let T identificador puntoYcoma; B -> .S; B -> .do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma; F -> .I J G; 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; I -> .function H identificador}
goto(0, if)	{B -> if.abrirParentesis E cerrarParentesis S}	4	{B -> if.abrirParentesis E cerrarParentesis S}
goto(0, let)	{B -> let.T identificador puntoYcoma}	5	{B -> let.T identificador puntoYcoma; T -> .number; T -> .boolean; T -> .string}
goto(0, S)	{B -> S.}	6	{B -> S.}
goto(0, do)	{B -> do.abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}	7	{B -> do.abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}
goto(0, I)	{F -> I.J G}	8	{F -> I.J G; J -> .abrirParentesis A cerrarParentesis}
goto(0, identificador)	{S -> identificador.asignacion E puntoYcoma; S -> identificador.abrirParentesis L cerrarParentesis puntoYcoma; S -> identificador.restaAsignacion E puntoYcoma}	9	{S -> identificador.asignacion E puntoYcoma; S -> identificador.abrirParentesis L cerrarParentesis puntoYcoma; S -> identificador.restaAsignacion E puntoYcoma}
goto(0, alert)	{S -> alert.abrirParentesis E cerrarParentesis puntoYcoma}	10	{S -> alert.abrirParentesis E cerrarParentesis puntoYcoma}
goto(0, input)	{S -> input.abrirParentesis identificador cerrarParentesis puntoYcoma}	11	{S -> input.abrirParentesis identificador cerrarParentesis puntoYcoma}
goto(0, return)	{S -> return.X puntoYcoma}	12	{S -> return.X puntoYcoma; X -> .E; X -> .; 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}
goto(0, function)	{I -> function.H identificador}	13	{I -> function.H identificador; H -> .T; H -> .; T -> .number; T -> .boolean; T -> .string}
goto(2, P)	{P -> B P.}	14	{P -> B P.}
goto(2, B)	{P -> B.P}	2	
goto(2, F)	{P -> F.P}	3	
goto(2, if)	{B -> if.abrirParentesis E cerrarParentesis S}	4	
goto(2, let)	{B -> let.T identificador puntoYcoma}	5	
goto(2, S)	{B -> S.}	6	
goto(2, do)	{B -> do.abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}	7	
goto(2, I)	{F -> I.J G}	8	
goto(2, identificador)	{S -> identificador.asignacion E puntoYcoma; S -> identificador.abrirParentesis L cerrarParentesis puntoYcoma; S -> identificador.restaAsignacion E puntoYcoma}	9	
goto(2, alert)	{S -> alert.abrirParentesis E cerrarParentesis puntoYcoma}	10	
goto(2, input)	{S -> input.abrirParentesis identificador cerrarParentesis puntoYcoma}	11	
goto(2, return)	{S -> return.X puntoYcoma}	12	
goto(2, function)	{I -> function.H identificador}	13	
goto(3, P)	{P -> F P.}	15	{P -> F P.}
goto(3, B)	{P -> B.P}	2	
goto(3, F)	{P -> F.P}	3	
goto(3, if)	{B -> if.abrirParentesis E cerrarParentesis S}	4	
goto(3, let)	{B -> let.T identificador puntoYcoma}	5	
goto(3, S)	{B -> S.}	6	
goto(3, do)	{B -> do.abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}	7	
goto(3, I)	{F -> I.J G}	8	
goto(3, identificador)	{S -> identificador.asignacion E puntoYcoma; S -> identificador.abrirParentesis L cerrarParentesis puntoYcoma; S -> identificador.restaAsignacion E puntoYcoma}	9	
goto(3, alert)	{S -> alert.abrirParentesis E cerrarParentesis puntoYcoma}	10	
goto(3, input)	{S -> input.abrirParentesis identificador cerrarParentesis puntoYcoma}	11	
goto(3, return)	{S -> return.X puntoYcoma}	12	
goto(3, function)	{I -> function.H identificador}	13	
goto(4, abrirParentesis)	{B -> if abrirParentesis.E cerrarParentesis S}	16	{B -> if abrirParentesis.E cerrarParentesis S; 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}
goto(5, T)	{B -> let T.identificador puntoYcoma}	17	{B -> let T.identificador puntoYcoma}
goto(5, number)	{T -> number.}	18	{T -> number.}
goto(5, boolean)	{T -> boolean.}	19	{T -> boolean.}
goto(5, string)	{T -> string.}	20	{T -> string.}
goto(7, abrirCorchete)	{B -> do abrirCorchete.C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}	21	{B -> do abrirCorchete.C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma; C -> .B C; C -> .; B -> .if abrirParentesis E cerrarParentesis S; B -> .let T identificador puntoYcoma; B -> .S; B -> .do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma; 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}
goto(8, J)	{F -> I J.G}	22	{F -> I J.G; G -> .abrirCorchete C cerrarCorchete}
goto(8, abrirParentesis)	{J -> abrirParentesis.A cerrarParentesis}	23	{J -> abrirParentesis.A cerrarParentesis; A -> .T identificador K; A -> .; T -> .number; T -> .boolean; T -> .string}
goto(9, asignacion)	{S -> identificador asignacion.E puntoYcoma}	24	{S -> identificador asignacion.E puntoYcoma; 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}
goto(9, abrirParentesis)	{S -> identificador abrirParentesis.L cerrarParentesis puntoYcoma}	25	{S -> identificador abrirParentesis.L cerrarParentesis puntoYcoma; L -> .E Q; L -> .; 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}
goto(9, restaAsignacion)	{S -> identificador restaAsignacion.E puntoYcoma}	26	{S -> identificador restaAsignacion.E puntoYcoma; 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}
goto(10, abrirParentesis)	{S -> alert abrirParentesis.E cerrarParentesis puntoYcoma}	27	{S -> alert abrirParentesis.E cerrarParentesis puntoYcoma; 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}
goto(11, abrirParentesis)	{S -> input abrirParentesis.identificador cerrarParentesis puntoYcoma}	28	{S -> input abrirParentesis.identificador cerrarParentesis puntoYcoma}
goto(12, X)	{S -> return X.puntoYcoma}	29	{S -> return X.puntoYcoma}
goto(12, E)	{X -> E.; E -> E.opLogico2 R}	30	{X -> E.; E -> E.opLogico2 R}
goto(12, R)	{E -> R.; R -> R.opLogico1 U}	31	{E -> R.; R -> R.opLogico1 U}
goto(12, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}
goto(12, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}
goto(12, W)	{V -> W.}	34	{V -> W.}
goto(12, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}
goto(12, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	{W -> abrirParentesis.E cerrarParentesis; 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}
goto(12, cteEntera)	{W -> cteEntera.}	37	{W -> cteEntera.}
goto(12, cadena)	{W -> cadena.}	38	{W -> cadena.}
goto(13, H)	{I -> function H.identificador}	39	{I -> function H.identificador}
goto(13, T)	{H -> T.}	40	{H -> T.}
goto(13, number)	{T -> number.}	18	
goto(13, boolean)	{T -> boolean.}	19	
goto(13, string)	{T -> string.}	20	
goto(16, E)	{B -> if abrirParentesis E.cerrarParentesis S; E -> E.opLogico2 R}	41	{B -> if abrirParentesis E.cerrarParentesis S; E -> E.opLogico2 R}
goto(16, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(16, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(16, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(16, W)	{V -> W.}	34	
goto(16, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(16, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(16, cteEntera)	{W -> cteEntera.}	37	
goto(16, cadena)	{W -> cadena.}	38	
goto(17, identificador)	{B -> let T identificador.puntoYcoma}	42	{B -> let T identificador.puntoYcoma}
goto(21, C)	{B -> do abrirCorchete C.cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}	43	{B -> do abrirCorchete C.cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}
goto(21, B)	{C -> B.C}	44	{C -> B.C; C -> .B C; C -> .; B -> .if abrirParentesis E cerrarParentesis S; B -> .let T identificador puntoYcoma; B -> .S; B -> .do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma; 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}
goto(21, if)	{B -> if.abrirParentesis E cerrarParentesis S}	4	
goto(21, let)	{B -> let.T identificador puntoYcoma}	5	
goto(21, S)	{B -> S.}	6	
goto(21, do)	{B -> do.abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}	7	
goto(21, identificador)	{S -> identificador.asignacion E puntoYcoma; S -> identificador.abrirParentesis L cerrarParentesis puntoYcoma; S -> identificador.restaAsignacion E puntoYcoma}	9	
goto(21, alert)	{S -> alert.abrirParentesis E cerrarParentesis puntoYcoma}	10	
goto(21, input)	{S -> input.abrirParentesis identificador cerrarParentesis puntoYcoma}	11	
goto(21, return)	{S -> return.X puntoYcoma}	12	
goto(22, G)	{F -> I J G.}	45	{F -> I J G.}
goto(22, abrirCorchete)	{G -> abrirCorchete.C cerrarCorchete}	46	{G -> abrirCorchete.C cerrarCorchete; C -> .B C; C -> .; B -> .if abrirParentesis E cerrarParentesis S; B -> .let T identificador puntoYcoma; B -> .S; B -> .do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma; 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}
goto(23, A)	{J -> abrirParentesis A.cerrarParentesis}	47	{J -> abrirParentesis A.cerrarParentesis}
goto(23, T)	{A -> T.identificador K}	48	{A -> T.identificador K}
goto(23, number)	{T -> number.}	18	
goto(23, boolean)	{T -> boolean.}	19	
goto(23, string)	{T -> string.}	20	
goto(24, E)	{S -> identificador asignacion E.puntoYcoma; E -> E.opLogico2 R}	49	{S -> identificador asignacion E.puntoYcoma; E -> E.opLogico2 R}
goto(24, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(24, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(24, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(24, W)	{V -> W.}	34	
goto(24, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(24, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(24, cteEntera)	{W -> cteEntera.}	37	
goto(24, cadena)	{W -> cadena.}	38	
goto(25, L)	{S -> identificador abrirParentesis L.cerrarParentesis puntoYcoma}	50	{S -> identificador abrirParentesis L.cerrarParentesis puntoYcoma}
goto(25, E)	{L -> E.Q; E -> E.opLogico2 R}	51	{L -> E.Q; E -> E.opLogico2 R; Q -> .coma E Q; Q -> .}
goto(25, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(25, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(25, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(25, W)	{V -> W.}	34	
goto(25, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(25, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(25, cteEntera)	{W -> cteEntera.}	37	
goto(25, cadena)	{W -> cadena.}	38	
goto(26, E)	{S -> identificador restaAsignacion E.puntoYcoma; E -> E.opLogico2 R}	52	{S -> identificador restaAsignacion E.puntoYcoma; E -> E.opLogico2 R}
goto(26, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(26, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(26, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(26, W)	{V -> W.}	34	
goto(26, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(26, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(26, cteEntera)	{W -> cteEntera.}	37	
goto(26, cadena)	{W -> cadena.}	38	
goto(27, E)	{S -> alert abrirParentesis E.cerrarParentesis puntoYcoma; E -> E.opLogico2 R}	53	{S -> alert abrirParentesis E.cerrarParentesis puntoYcoma; E -> E.opLogico2 R}
goto(27, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(27, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(27, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(27, W)	{V -> W.}	34	
goto(27, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(27, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(27, cteEntera)	{W -> cteEntera.}	37	
goto(27, cadena)	{W -> cadena.}	38	
goto(28, identificador)	{S -> input abrirParentesis identificador.cerrarParentesis puntoYcoma}	54	{S -> input abrirParentesis identificador.cerrarParentesis puntoYcoma}
goto(29, puntoYcoma)	{S -> return X puntoYcoma.}	55	{S -> return X puntoYcoma.}
goto(30, opLogico2)	{E -> E opLogico2.R}	56	{E -> E opLogico2.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}
goto(31, opLogico1)	{R -> R opLogico1.U}	57	{R -> R opLogico1.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}
goto(32, opRelacional1)	{U -> U opRelacional1.V}	58	{U -> U opRelacional1.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}
goto(32, opRelacional2)	{U -> U opRelacional2.V}	59	{U -> U opRelacional2.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}
goto(33, opAritmetico1)	{V -> V opAritmetico1.W}	60	{V -> V opAritmetico1.W; W -> .identificador; W -> .abrirParentesis E cerrarParentesis; W -> .identificador abrirParentesis L cerrarParentesis; W -> .cteEntera; W -> .cadena}
goto(33, opAritmetico2)	{V -> V opAritmetico2.W}	61	{V -> V opAritmetico2.W; W -> .identificador; W -> .abrirParentesis E cerrarParentesis; W -> .identificador abrirParentesis L cerrarParentesis; W -> .cteEntera; W -> .cadena}
goto(35, abrirParentesis)	{W -> identificador abrirParentesis.L cerrarParentesis}	62	{W -> identificador abrirParentesis.L cerrarParentesis; L -> .E Q; L -> .; 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}
goto(36, E)	{W -> abrirParentesis E.cerrarParentesis; E -> E.opLogico2 R}	63	{W -> abrirParentesis E.cerrarParentesis; E -> E.opLogico2 R}
goto(36, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(36, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(36, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(36, W)	{V -> W.}	34	
goto(36, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(36, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(36, cteEntera)	{W -> cteEntera.}	37	
goto(36, cadena)	{W -> cadena.}	38	
goto(39, identificador)	{I -> function H identificador.}	64	{I -> function H identificador.}
goto(41, cerrarParentesis)	{B -> if abrirParentesis E cerrarParentesis.S}	65	{B -> if abrirParentesis E cerrarParentesis.S; 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}
goto(41, opLogico2)	{E -> E opLogico2.R}	56	
goto(42, puntoYcoma)	{B -> let T identificador puntoYcoma.}	66	{B -> let T identificador puntoYcoma.}
goto(43, cerrarCorchete)	{B -> do abrirCorchete C cerrarCorchete.while abrirParentesis E cerrarParentesis puntoYcoma}	67	{B -> do abrirCorchete C cerrarCorchete.while abrirParentesis E cerrarParentesis puntoYcoma}
goto(44, C)	{C -> B C.}	68	{C -> B C.}
goto(44, B)	{C -> B.C}	44	
goto(44, if)	{B -> if.abrirParentesis E cerrarParentesis S}	4	
goto(44, let)	{B -> let.T identificador puntoYcoma}	5	
goto(44, S)	{B -> S.}	6	
goto(44, do)	{B -> do.abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}	7	
goto(44, identificador)	{S -> identificador.asignacion E puntoYcoma; S -> identificador.abrirParentesis L cerrarParentesis puntoYcoma; S -> identificador.restaAsignacion E puntoYcoma}	9	
goto(44, alert)	{S -> alert.abrirParentesis E cerrarParentesis puntoYcoma}	10	
goto(44, input)	{S -> input.abrirParentesis identificador cerrarParentesis puntoYcoma}	11	
goto(44, return)	{S -> return.X puntoYcoma}	12	
goto(46, C)	{G -> abrirCorchete C.cerrarCorchete}	69	{G -> abrirCorchete C.cerrarCorchete}
goto(46, B)	{C -> B.C}	44	
goto(46, if)	{B -> if.abrirParentesis E cerrarParentesis S}	4	
goto(46, let)	{B -> let.T identificador puntoYcoma}	5	
goto(46, S)	{B -> S.}	6	
goto(46, do)	{B -> do.abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma}	7	
goto(46, identificador)	{S -> identificador.asignacion E puntoYcoma; S -> identificador.abrirParentesis L cerrarParentesis puntoYcoma; S -> identificador.restaAsignacion E puntoYcoma}	9	
goto(46, alert)	{S -> alert.abrirParentesis E cerrarParentesis puntoYcoma}	10	
goto(46, input)	{S -> input.abrirParentesis identificador cerrarParentesis puntoYcoma}	11	
goto(46, return)	{S -> return.X puntoYcoma}	12	
goto(47, cerrarParentesis)	{J -> abrirParentesis A cerrarParentesis.}	70	{J -> abrirParentesis A cerrarParentesis.}
goto(48, identificador)	{A -> T identificador.K}	71	{A -> T identificador.K; K -> .coma T identificador K; K -> .}
goto(49, puntoYcoma)	{S -> identificador asignacion E puntoYcoma.}	72	{S -> identificador asignacion E puntoYcoma.}
goto(49, opLogico2)	{E -> E opLogico2.R}	56	
goto(50, cerrarParentesis)	{S -> identificador abrirParentesis L cerrarParentesis.puntoYcoma}	73	{S -> identificador abrirParentesis L cerrarParentesis.puntoYcoma}
goto(51, Q)	{L -> E Q.}	74	{L -> E Q.}
goto(51, opLogico2)	{E -> E opLogico2.R}	56	
goto(51, coma)	{Q -> coma.E Q}	75	{Q -> coma.E Q; 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}
goto(52, puntoYcoma)	{S -> identificador restaAsignacion E puntoYcoma.}	76	{S -> identificador restaAsignacion E puntoYcoma.}
goto(52, opLogico2)	{E -> E opLogico2.R}	56	
goto(53, cerrarParentesis)	{S -> alert abrirParentesis E cerrarParentesis.puntoYcoma}	77	{S -> alert abrirParentesis E cerrarParentesis.puntoYcoma}
goto(53, opLogico2)	{E -> E opLogico2.R}	56	
goto(54, cerrarParentesis)	{S -> input abrirParentesis identificador cerrarParentesis.puntoYcoma}	78	{S -> input abrirParentesis identificador cerrarParentesis.puntoYcoma}
goto(56, R)	{E -> E opLogico2 R.; R -> R.opLogico1 U}	79	{E -> E opLogico2 R.; R -> R.opLogico1 U}
goto(56, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(56, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(56, W)	{V -> W.}	34	
goto(56, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(56, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(56, cteEntera)	{W -> cteEntera.}	37	
goto(56, cadena)	{W -> cadena.}	38	
goto(57, U)	{R -> R opLogico1 U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	80	{R -> R opLogico1 U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}
goto(57, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(57, W)	{V -> W.}	34	
goto(57, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(57, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(57, cteEntera)	{W -> cteEntera.}	37	
goto(57, cadena)	{W -> cadena.}	38	
goto(58, V)	{U -> U opRelacional1 V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	81	{U -> U opRelacional1 V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}
goto(58, W)	{V -> W.}	34	
goto(58, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(58, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(58, cteEntera)	{W -> cteEntera.}	37	
goto(58, cadena)	{W -> cadena.}	38	
goto(59, V)	{U -> U opRelacional2 V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	82	{U -> U opRelacional2 V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}
goto(59, W)	{V -> W.}	34	
goto(59, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(59, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(59, cteEntera)	{W -> cteEntera.}	37	
goto(59, cadena)	{W -> cadena.}	38	
goto(60, W)	{V -> V opAritmetico1 W.}	83	{V -> V opAritmetico1 W.}
goto(60, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(60, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(60, cteEntera)	{W -> cteEntera.}	37	
goto(60, cadena)	{W -> cadena.}	38	
goto(61, W)	{V -> V opAritmetico2 W.}	84	{V -> V opAritmetico2 W.}
goto(61, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(61, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(61, cteEntera)	{W -> cteEntera.}	37	
goto(61, cadena)	{W -> cadena.}	38	
goto(62, L)	{W -> identificador abrirParentesis L.cerrarParentesis}	85	{W -> identificador abrirParentesis L.cerrarParentesis}
goto(62, E)	{L -> E.Q; E -> E.opLogico2 R}	51	
goto(62, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(62, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(62, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(62, W)	{V -> W.}	34	
goto(62, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(62, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(62, cteEntera)	{W -> cteEntera.}	37	
goto(62, cadena)	{W -> cadena.}	38	
goto(63, cerrarParentesis)	{W -> abrirParentesis E cerrarParentesis.}	86	{W -> abrirParentesis E cerrarParentesis.}
goto(63, opLogico2)	{E -> E opLogico2.R}	56	
goto(65, S)	{B -> if abrirParentesis E cerrarParentesis S.}	87	{B -> if abrirParentesis E cerrarParentesis S.}
goto(65, identificador)	{S -> identificador.asignacion E puntoYcoma; S -> identificador.abrirParentesis L cerrarParentesis puntoYcoma; S -> identificador.restaAsignacion E puntoYcoma}	9	
goto(65, alert)	{S -> alert.abrirParentesis E cerrarParentesis puntoYcoma}	10	
goto(65, input)	{S -> input.abrirParentesis identificador cerrarParentesis puntoYcoma}	11	
goto(65, return)	{S -> return.X puntoYcoma}	12	
goto(67, while)	{B -> do abrirCorchete C cerrarCorchete while.abrirParentesis E cerrarParentesis puntoYcoma}	88	{B -> do abrirCorchete C cerrarCorchete while.abrirParentesis E cerrarParentesis puntoYcoma}
goto(69, cerrarCorchete)	{G -> abrirCorchete C cerrarCorchete.}	89	{G -> abrirCorchete C cerrarCorchete.}
goto(71, K)	{A -> T identificador K.}	90	{A -> T identificador K.}
goto(71, coma)	{K -> coma.T identificador K}	91	{K -> coma.T identificador K; T -> .number; T -> .boolean; T -> .string}
goto(73, puntoYcoma)	{S -> identificador abrirParentesis L cerrarParentesis puntoYcoma.}	92	{S -> identificador abrirParentesis L cerrarParentesis puntoYcoma.}
goto(75, E)	{Q -> coma E.Q; E -> E.opLogico2 R}	93	{Q -> coma E.Q; E -> E.opLogico2 R; Q -> .coma E Q; Q -> .}
goto(75, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(75, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(75, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(75, W)	{V -> W.}	34	
goto(75, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(75, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(75, cteEntera)	{W -> cteEntera.}	37	
goto(75, cadena)	{W -> cadena.}	38	
goto(77, puntoYcoma)	{S -> alert abrirParentesis E cerrarParentesis puntoYcoma.}	94	{S -> alert abrirParentesis E cerrarParentesis puntoYcoma.}
goto(78, puntoYcoma)	{S -> input abrirParentesis identificador cerrarParentesis puntoYcoma.}	95	{S -> input abrirParentesis identificador cerrarParentesis puntoYcoma.}
goto(79, opLogico1)	{R -> R opLogico1.U}	57	
goto(80, opRelacional1)	{U -> U opRelacional1.V}	58	
goto(80, opRelacional2)	{U -> U opRelacional2.V}	59	
goto(81, opAritmetico1)	{V -> V opAritmetico1.W}	60	
goto(81, opAritmetico2)	{V -> V opAritmetico2.W}	61	
goto(82, opAritmetico1)	{V -> V opAritmetico1.W}	60	
goto(82, opAritmetico2)	{V -> V opAritmetico2.W}	61	
goto(85, cerrarParentesis)	{W -> identificador abrirParentesis L cerrarParentesis.}	96	{W -> identificador abrirParentesis L cerrarParentesis.}
goto(88, abrirParentesis)	{B -> do abrirCorchete C cerrarCorchete while abrirParentesis.E cerrarParentesis puntoYcoma}	97	{B -> do abrirCorchete C cerrarCorchete while abrirParentesis.E cerrarParentesis puntoYcoma; 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}
goto(91, T)	{K -> coma T.identificador K}	98	{K -> coma T.identificador K}
goto(91, number)	{T -> number.}	18	
goto(91, boolean)	{T -> boolean.}	19	
goto(91, string)	{T -> string.}	20	
goto(93, Q)	{Q -> coma E Q.}	99	{Q -> coma E Q.}
goto(93, opLogico2)	{E -> E opLogico2.R}	56	
goto(93, coma)	{Q -> coma.E Q}	75	
goto(97, E)	{B -> do abrirCorchete C cerrarCorchete while abrirParentesis E.cerrarParentesis puntoYcoma; E -> E.opLogico2 R}	100	{B -> do abrirCorchete C cerrarCorchete while abrirParentesis E.cerrarParentesis puntoYcoma; E -> E.opLogico2 R}
goto(97, R)	{E -> R.; R -> R.opLogico1 U}	31	
goto(97, U)	{R -> U.; U -> U.opRelacional1 V; U -> U.opRelacional2 V}	32	
goto(97, V)	{U -> V.; V -> V.opAritmetico1 W; V -> V.opAritmetico2 W}	33	
goto(97, W)	{V -> W.}	34	
goto(97, identificador)	{W -> identificador.; W -> identificador.abrirParentesis L cerrarParentesis}	35	
goto(97, abrirParentesis)	{W -> abrirParentesis.E cerrarParentesis}	36	
goto(97, cteEntera)	{W -> cteEntera.}	37	
goto(97, cadena)	{W -> cadena.}	38	
goto(98, identificador)	{K -> coma T identificador.K}	101	{K -> coma T identificador.K; K -> .coma T identificador K; K -> .}
goto(100, cerrarParentesis)	{B -> do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis.puntoYcoma}	102	{B -> do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis.puntoYcoma}
goto(100, opLogico2)	{E -> E opLogico2.R}	56	
goto(101, K)	{K -> coma T identificador K.}	103	{K -> coma T identificador K.}
goto(101, coma)	{K -> coma.T identificador K}	91	
goto(102, puntoYcoma)	{B -> do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma.}	104	{B -> do abrirCorchete C cerrarCorchete while abrirParentesis E cerrarParentesis puntoYcoma.}

State	ACTION																													
	function	identificador	abrirParentesis	cerrarParentesis	abrirCorchete	cerrarCorchete	coma	asignacion	puntoYcoma	alert	input	return	restaAsignacion	if	let	do	while	number	boolean	string	opLogico2	opLogico1	opRelacional1	opRelacional2	opAritmetico1	opAritmetico2	cteEntera	cadena	$
0	s13	s9								s10	s11	s12		s4	s5	s7													r3
1																													acc
2	s13	s9								s10	s11	s12		s4	s5	s7													r3
3	s13	s9								s10	s11	s12		s4	s5	s7													r3
4			s16																										
5																		s18	s19	s20									
6	r24	r24				r24				r24	r24	r24		r24	r24	r24													r24
7					s21																								
8			s23																										
9			s25					s24					s26																
10			s27																										
11			s28																										
12		s35	s36						r49																		s37	s38	
13		r3																s18	s19	s20									
14																													r1
15																													r2
16		s35	s36																								s37	s38	
17		s42																											
18		r6																											
19		r7																											
20		r8																											
21		s9				r15				s10	s11	s12		s4	s5	s7													
22					s46																								
23				r11														s18	s19	s20									
24		s35	s36																								s37	s38	
25		s35	s36	r45																							s37	s38	
26		s35	s36																								s37	s38	
27		s35	s36																								s37	s38	
28		s54																											
29									s55																				
30									r45												r56								
31				r30			r30		r30												r30	r57							
32				r32			r32		r32												r33	r32	s58	s59					
33				r35			r35		r35												r35	r35	r35	r35	s60	s61			
34				r38			r38		r38												r38	r38	r38	r38	r38	r38			
35			s62	r39			r39		r39												r39	r39	r39	r39	r39	r39			
36		s35	s36																								s37	s38	
37				r42			r42		r42												r43	r42	r43	r42	r43	r42			
38				r43			r43		r43												r43	r43	r43	r43	r43	r43			
39		s64																											
40		r9																											
41				s65																	r56								
42									s66																				
43						s67																							
44		s9				r15				s10	s11	s12		s4	s5	s7													
45	r4	r4								r4	r4	r4		r4	r4	r4													r4
46		s9				r15				s10	s11	s12		s4	s5	s7													
47				s70																									
48		s71																											
49									s72												r56								
50				s73																									
51				r47			s75														r56								
52									s76												r56								
53				s77																	r56								
54				s78																									
55	r20	r20				r20				r20	r20	r20		r20	r20	r20													r20
56		s35	s36																								s37	s38	
57		s35	s36																								s37	s38	
58		s35	s36																								s37	s38	
59		s35	s36																								s37	s38	
60		s35	s36																								s37	s38	
61		s35	s36																								s37	s38	
62		s35	s36	r45																							s37	s38	
63			s86																		r56								
64			r5																										
65		s9								s10	s11	s12																	
66	r23	r23				r23				r23	r23	r23		r23	r23	r23													r23
67																	s88												
68						r14																							
69						s89																							
70					r6																								
71				r13			s91																						
72	r16	r16				r16				r16	r16	r16		r16	r16	r16													r16
73									s92																				
74				r44																									
75		s35	s36																								s37	s38	
76	r21	r21				r21				r21	r21	r21		r21	r21	r21													r21
77									s94																				
78									s95																				
79				r23			r23		r23												r33	r57							
80				r31			r31		r31												r31	r31	s58	s59					
81				r33			r33		r33												r33	r33	r33	r33	s60	s61			
82				r34			r34		r34												r34	r34	r34	r34	s60	s61			
83				r36			r36		r36												r36	r36	r36	r36	r36	r36			
84				r37			r37		r37												r37	r37	r37	r37	r37	r37			
85				s96																									
86				r40			r40		r40												r40	r40	r40	r40	r40	r40			
87	r22	r22				r22				r22	r22	r22		r22	r22	r22													r22
88			s97																										
89	r7	r7								r7	r7	r7		r7	r7	r7													r7
90				r10																									
91																		s18	s19	s20									
92	r17	r17				r17				r17	r17	r17		r17	r17	r17													r17
93				r47			s75														r56								
94	r18	r18				r18				r18	r18	r18		r18	r18	r18													r18
95	r19	r19				r19				r19	r19	r19		r19	r19	r19													r19
96				r41			r41		r41												r41	r41	r41	r41	r41	r41			
97		s35	s36																								s37	s38	
98		s101																											
99				r46																									
100				s102																	r56								
101				r13			s91																						
102									s104																				
103				r12																									
104	r25	r25				r25				r25	r25	r25		r25	r25	r25													r25

GOTO																			
Z	P	F	I	J	G	H	A	K	C	S	B	T	E	R	U	V	W	L	Q	X
	1	3	8							6	2									
	14	3	8							6	2									
	15	3	8							6	2									
												17								
				22																
													30	31	32	33	34			29
						39						40								
													41	31	32	33	34			
									43	6	44									
					45															
							47					48								
													49	31	32	33	34			
													51	31	32	33	34	50		
													52	31	32	33	34			
													53	31	32	33	34			
													63	31	32	33	34			
									68	6	44									
									69	6	44									
																			74	
														79	32	33	34			
															80	33	34			
																81	34			
																82	34			
																	83			
																	84			
													51	31	32	33	34	85		
										87										
								90												
													93	31	32	33	34			
												98								
																			99	
													100	31	32	33	34			
								103												

LR table

ACTION

State	function	identificador	abrirParentesis	cerrarParentesis	abrirCorchete	cerrarCorchete	coma	asignacion	puntoYcoma	alert	input	return	restaAsignacion	if	let	do	while	number	boolean	string	opLogico2	opLogico1	opRelacional1	opRelacional2	opAritmetico1	opAritmetico2	cteEntera	cadena	$
0	s13	s9								s10	s11	s12		s4	s5	s7													r_3
1																													acc
2	s13	s9								s10	s11	s12		s4	s5	s7													r_3
3	s13	s9								s10	s11	s12		s4	s5	s7													r_3
4			s16																										
5																		s18	s19	s20									
6	r_{24}	r_{24}				r_{24}				r_{24}	r_{24}	r_{24}		r_{24}	r_{24}	r_{24}													r_{24}
7					s21																								
8			s23																										
9			s25					s24					s26																
10			s27																										
11			s28																										
12		s35	s36						r_{49}																		s37	s38	
13		r_9																s18	s19	s20									
14																													r_1
15																													r_2
16		s35	s36																								s37	s38	
17		s42																											
18		r_{26}																											
19		r_{27}																											
20		r_{28}																											
21		s9				r_{15}				s10	s11	s12		s4	s5	s7													
22					s46																								
23				r_{11}														s18	s19	s20									
24		s35	s36																								s37	s38	
25		s35	s36	r_{45}																							s37	s38	
26		s35	s36																								s37	s38	
27		s35	s36																								s37	s38	
28		s54																											
29									s55																				
30									r_{48}												s56								
31				r_{30}			r_{30}		r_{30}												r_{30}	s57							
32				r_{32}			r_{32}		r_{32}												r_{32}	r_{32}	s58	s59					
33				r_{35}			r_{35}		r_{35}												r_{35}	r_{35}	r_{35}	r_{35}	s60	s61			
34				r_{38}			r_{38}		r_{38}												r_{38}	r_{38}	r_{38}	r_{38}	r_{38}	r_{38}			
35			s62	r_{39}			r_{39}		r_{39}												r_{39}	r_{39}	r_{39}	r_{39}	r_{39}	r_{39}			
36		s35	s36																								s37	s38	
37				r_{42}			r_{42}		r_{42}												r_{42}	r_{42}	r_{42}	r_{42}	r_{42}	r_{42}			
38				r_{43}			r_{43}		r_{43}												r_{43}	r_{43}	r_{43}	r_{43}	r_{43}	r_{43}			
39		s64																											
40		r_8																											
41				s65																	s56								
42									s66																				
43						s67																							
44		s9				r_{15}				s10	s11	s12		s4	s5	s7													
45	r_4	r_4								r_4	r_4	r_4		r_4	r_4	r_4													r_4
46		s9				r_{15}				s10	s11	s12		s4	s5	s7													
47				s70																									
48		s71																											
49									s72												s56								
50				s73																									
51				r_{47}			s75														s56								
52									s76												s56								
53				s77																	s56								
54				s78																									
55	r_{20}	r_{20}				r_{20}				r_{20}	r_{20}	r_{20}		r_{20}	r_{20}	r_{20}													r_{20}
56		s35	s36																								s37	s38	
57		s35	s36																								s37	s38	
58		s35	s36																								s37	s38	
59		s35	s36																								s37	s38	
60		s35	s36																								s37	s38	
61		s35	s36																								s37	s38	
62		s35	s36	r_{45}																							s37	s38	
63				s86																	s56								
64			r_5																										
65		s9								s10	s11	s12																	
66	r_{23}	r_{23}				r_{23}				r_{23}	r_{23}	r_{23}		r_{23}	r_{23}	r_{23}													r_{23}
67																	s88												
68						r_{14}																							
69						s89																							
70					r_6																								
71				r_{13}			s91																						
72	r_{16}	r_{16}				r_{16}				r_{16}	r_{16}	r_{16}		r_{16}	r_{16}	r_{16}													r_{16}
73									s92																				
74				r_{44}																									
75		s35	s36																								s37	s38	
76	r_{21}	r_{21}				r_{21}				r_{21}	r_{21}	r_{21}		r_{21}	r_{21}	r_{21}													r_{21}
77									s94																				
78									s95																				
79				r_{29}			r_{29}		r_{29}												r_{29}	s57							
80				r_{31}			r_{31}		r_{31}												r_{31}	r_{31}	s58	s59					
81				r_{33}			r_{33}		r_{33}												r_{33}	r_{33}	r_{33}	r_{33}	s60	s61			
82				r_{34}			r_{34}		r_{34}												r_{34}	r_{34}	r_{34}	r_{34}	s60	s61			
83				r_{36}			r_{36}		r_{36}												r_{36}	r_{36}	r_{36}	r_{36}	r_{36}	r_{36}			
84				r_{37}			r_{37}		r_{37}												r_{37}	r_{37}	r_{37}	r_{37}	r_{37}	r_{37}			
85				s96																									
86				r_{40}			r_{40}		r_{40}												r_{40}	r_{40}	r_{40}	r_{40}	r_{40}	r_{40}			
87	r_{22}	r_{22}				r_{22}				r_{22}	r_{22}	r_{22}		r_{22}	r_{22}	r_{22}													r_{22}
88			s97																										
89	r_7	r_7								r_7	r_7	r_7		r_7	r_7	r_7													r_7
90				r_{10}																									
91																		s18	s19	s20									
92	r_{17}	r_{17}				r_{17}				r_{17}	r_{17}	r_{17}		r_{17}	r_{17}	r_{17}													r_{17}
93				r_{47}			s75														s56								
94	r_{18}	r_{18}				r_{18}				r_{18}	r_{18}	r_{18}		r_{18}	r_{18}	r_{18}													r_{18}
95	r_{19}	r_{19}				r_{19}				r_{19}	r_{19}	r_{19}		r_{19}	r_{19}	r_{19}													r_{19}
96				r_{41}			r_{41}		r_{41}												r_{41}	r_{41}	r_{41}	r_{41}	r_{41}	r_{41}			
97		s35	s36																								s37	s38	
98		s101																											
99				r_{46}																									
100				s102																	s56								
101				r_{13}			s91																						
102									s104																				
103				r_{12}																									
104	r_{25}	r_{25}				r_{25}				r_{25}	r_{25}	r_{25}		r_{25}	r_{25}	r_{25}													r_{25}

GOTO

State	Z	P	F	I	J	G	H	A	K	C	S	B	T	E	R	U	V	W	L	Q	X
0		1	3	8							6	2									
1																					
2		14	3	8							6	2									
3		15	3	8							6	2									
4																					
5													17								
6																					
7																					
8					22																
9																					
10																					
11																					
12														30	31	32	33	34			29
13							39						40								
14																					
15																					
16														41	31	32	33	34			
17																					
18																					
19																					
20																					
21										43	6	44									
22						45															
23								47					48								
24														49	31	32	33	34			
25														51	31	32	33	34	50		
26														52	31	32	33	34			
27														53	31	32	33	34			
28																					
29																					
30																					
31																					
32																					
33																					
34																					
35																					
36														63	31	32	33	34			
37																					
38																					
39																					
40																					
41																					
42																					
43																					
44										68	6	44									
45																					
46										69	6	44									
47																					
48																					
49																					
50																					
51																				74	
52																					
53																					
54																					
55																					
56															79	32	33	34			
57																80	33	34			
58																	81	34			
59																	82	34			
60																		83			
61																		84			
62														51	31	32	33	34	85		
63																					
64																					
65											87										
66																					
67																					
68																					
69																					
70																					
71									90												
72																					
73																					
74																					
75														93	31	32	33	34			
76																					
77																					
78																					
79																					
80																					
81																					
82																					
83																					
84																					
85																					
86																					
87																					
88																					
89																					
90																					
91													98								
92																					
93																				99	
94																					
95																					
96																					
97														100	31	32	33	34			
98																					
99																					
100																					
101									103												
102																					
103																					
104																					

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

Análisis Semántico

Aumentada

Z → P

```
1. If(P.tipo=tipoError) error(Revisa los errores que tienes, algo
   no anda bien)
2. Z.tipo = P.tipo
```

Inicio

P → BP

```
1. P.tipo = (B.tipo == tipoOk && P.tipo==tipoOk)? tipoOk :
   tipoError
2. If(B.hasReturn)
3.   Error(Return fuera de funcion)
4.   P.tipo=tipoError
```

P→ FP

```
1. P.tipo = (F.tipo == tipoOk && P.tipo==tipoOk)? tipoOk :
   tipoError
```

P→ λ

```
1. P.tipo = tipoOk
```

Funciones

F → I J G

```
1. if(G.hasReturn && G.returnType == I.Tipo)
2.   F.tipo = tipoOk
3. Else{
4.   F.tipo = tipoError
5.   Error()
6. }
```

I → function H id

```
1. ZonaDecl=True
2. TSActual = crearTS()
3. insertarEti(id.pos,"eti"+contador)
4. insertarReturn(id.pos,H.tipo)
```

J → (A)

```
1. insertarDatosFuncionTS(idUltimaFuncion,A.numeros,A.tipos)
```

G → { C }

```
1. G.tipo = C.tipo
```

```
2. G.returnType = C.returnType
3. G.hasReturn = C.hasReturn
```

H → T

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

H → λ

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

A → T id K

```
1. A.numeros=1+k.numeros
2. A.tipos=[T] U K.tipos
3. insertarTSActual(id,T.tipo)
```

A → λ

```
1. A.numeros = 0
2. A.tipos=[]
```

K → , T id K

```
1. K.numeros=1+K1.numeros
2. K.tipos = [T.tipo] U K.tipos
3. insertarTSActual(id,T.tipo)
```

K → λ

```
1. K.numeros = 0
2. K.tipo = []
```

C → BC

```
1. C.tipo = (B.tipo = C1.tipo = tipoOk)? tipoOk : (tipoError &&
   Error() )
2.
3. if(B.hasReturn){
4.   c.returnType = B.returnType
5. }else if(C1.hasReturn){
6.   C.returnType = C1.returnType
7. }
```

C → λ

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

Sentencias:

Simples

S → id = E ;

```
1. If(comprobarTipoTS(id) ¡= VACIO)
2.   S.tipo = (getTipoTS(id)== E.tipo)? tipoOk : tipoError
3. Else{
4.   insertarTSGlobal(id,NUMBER)
5.   S.tipo = (NUMBER== E.tipo)? tipoOk : tipoError
6. }
```

S → id (L) ;

```
1. If(estaFuncionTS(id))
2.   S.tipo = comprobarArgumentosYNum(ArgyNum(id.pos),L.tipos,
   L.numeros)? tipoOk : tipoError
3. Else
4.   Error("funcion no declarada")
```

S → alert (E) ;

```
1. S.tipo = (E.tipo == NUMBER || E.tipo=String)? tipoOK : tipoError
```

S → input (id) ;

```
1. If(tieneTipo(id.pos))
2.          S.tipo = (getTipoTS(id) == NUMBER ||
   getTipoTS(id)=String)? tipoOK : tipoError
3. Else
4.   insertarTSGlobal(id,NUMBER)
5.   S.tipo = (getTipoTS(id) == NUMBER || getTipoTS(id)=String)?
   tipoOK : tipoError
```

S → return X ;

```
1. S.returnType = X.tipo
2. S.hasReturn=true
```

S → id -= E ;

```
1. if(tieneTipo(id.pos))
2.   S.tipo = (getTipoTS(id) = E.tipo = NUMBER)? tipoOk: tipoError
3. Else{
4.   insertarTSGlobal(id,NUMBER)
5.   S.tipo = (E.tipo = NUMBER)? tipoOk: tipoError
6. }
```

Compuestas

B → if (E) S

```
1. B.atrib = (E.tipo == BOOLEAN && S.tipo=tipoOk)? S.atrib : (
   error()&& tipoError)
```

B → let T id ;

```
1. If(!tieneTipo(id.pos))
2.   insertarTSActual(getTipoTS(id),T.tipo)
3.   B.tipo = tipoOk
4. Else
5.   B.tipo = tipoError
```

B → S

```
1. B.atributos = S.atributos
2. # .atributos denota todos los posibles atributos que tenga S
```

B → do { C } while (E);

```
1. B.atrib = (E.tipo == BOOLEAN && C.tipo=tipoOk)? C.atrib :
   (error() && B.tipo = tipoError)
```

T → number

```
1. T.tipo = NUMBER
```

T → boolean

```
1. T.tipo = BOOLEAN
```

T → String

```
1. T.tipo = STRING
```

Expresiones

E → E || R

```
1. E.tipo = (E1.tipo = R.tipo = BOOLEAN)? BOOLEAN: error() &&
   E.tipo = tipoError
```

E → R

```
1. E.tipo = R.tipo
```

R → R && U

```
1. R.tipo = (R1.tipo = U.tipo = BOOLEAN)? BOOLEAN: error() &&
   R.tipo = tipoError
```

R → U

```
1. R .tipo = U.tipo
```

U → U == V

```
1. U.tipo = (U1.tipo == V.tipo)? BOOLEAN: error() && U.tipo =
   tipoError
```

U → U != V

```
1. U.tipo = (U1.tipo == V.tipo)? BOOLEAN: error() && U.tipo =
   tipoError
```

U → V

```
1. U.tipo = V.tipo
```

V → V + W

```
1. V.tipo = (V1.tipo == NUMBER && W.tipo == NUMBER)? NUMBER :
   error() && V.tipo = tipoError
```

V → V - W

```
1. V.tipo = (V1.tipo == NUMBER && W.tipo == NUMBER)? NUMBER :
   error() && V.tipo = tipoError
```

V → W

```
1. V.tipo = W.tipo
```

W → id

```
1. if(!estaEnTS(id.valor)) {
2.          insertarTSGlobal(id,NUMBER)
3. }
4. W.tipo = getTipoTS(id)
```

W → (E)

```
1. W.tipo = E.tipo
```

W → id (L)

```
1. if(getReturnTypeTS(id) != Empty &&
   comprobarArgumentosYNum(id,L.tipos,L.numeros)){
2.   W.tipo = getReturnTypeTS(id)
3. }else{
4.   error()
5.   W.tipo = tipoError
6. }
```

W → entero

```
1. W.tipo = NUMBER
```

W → cadena

```
1. W.tipo = STRING
```

Argumentos de la función

L → E Q

```
1. L.numeros = Q.numeros+1
2. L.tipos= [E.tipo] U Q.tipos
```

L → λ

```
1. L.numeros = 0
2. L.tipos=[]
```

Q → , E Q

```
1. Q.numeros = Q1.numeros+1
2. Q.tipos = [E.tipo] U Q.tipos
```

Q → λ

```
1. Q.numeros = 0
2. Q.tipos=[]
```

Return

X → E

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

X → λ

```
1. 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

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: ;
```

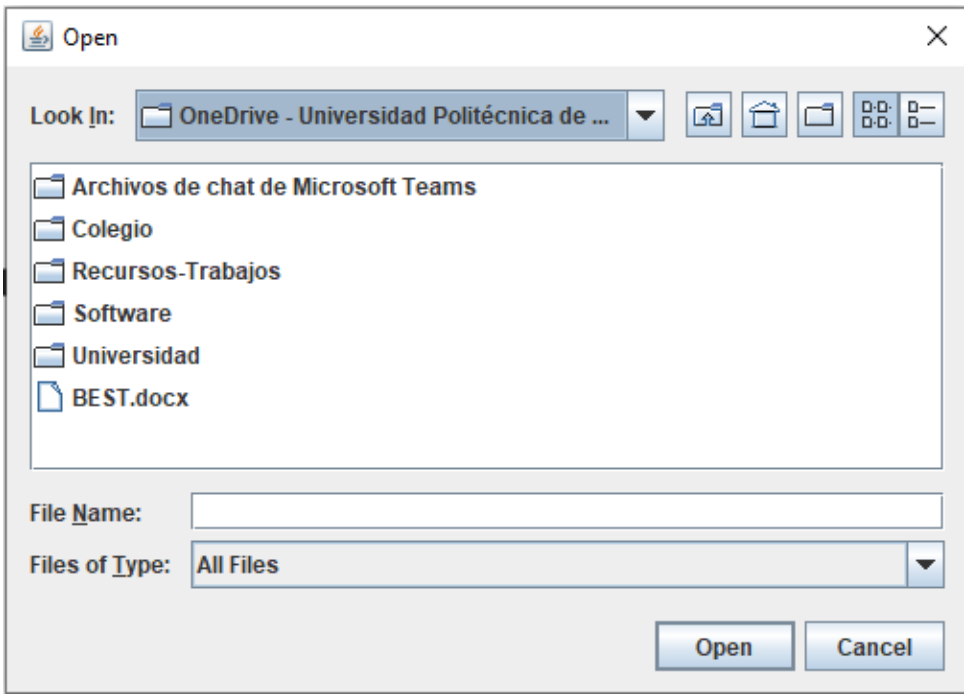
Código para VAST

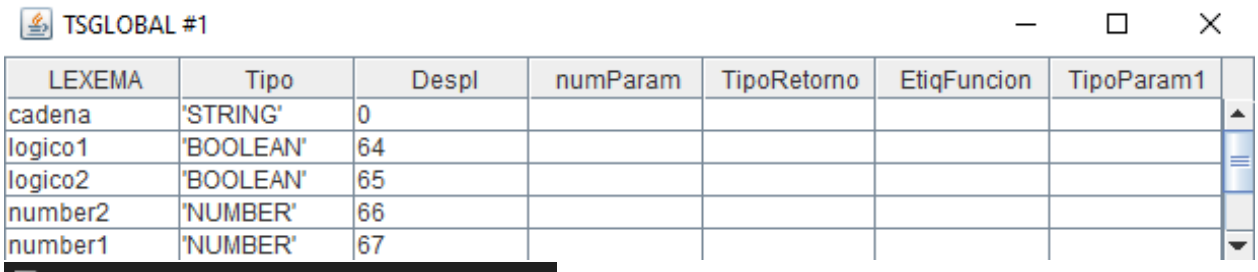
Incluimos el código usado para Vast

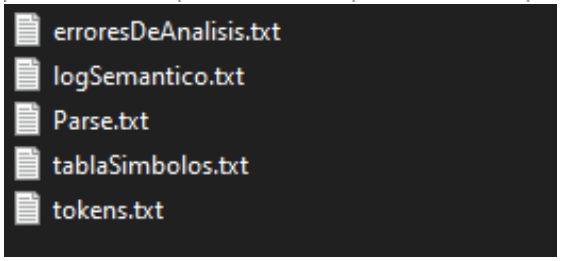
```
Terminales = { function identificador  cerrarParentesis
abrirParentesis   cerrarCorchete  abrirCorchete coma asignacion
puntoYcoma alert input return restaAsignacion if let do while number
boolean string opLogico2 opLogico1 opRelacional1 opRelacional2
opAritmetico1 opAritmetico2 cteEntera cadena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

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







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 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 3 2 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:

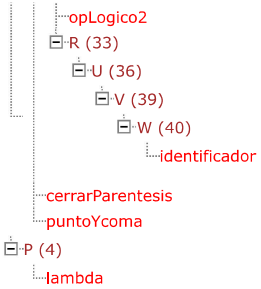
Gramática: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\2.Analizador Sintactico\GramaticaVast.txt

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


opLogico2
R (33)
U (36)
V (39)
W (40)
identificador
cerrarParentesis
puntoYcoma
P (4)
lambda

Árbol resultado de:

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

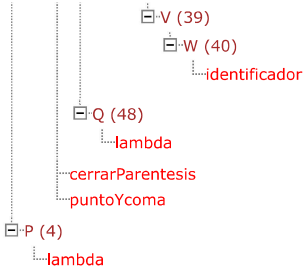
Parse: D:\OneDrive - Universidad Politécnica de Madrid\Universidad\3º\Procesadores de Lenguajes\Practica\ProcesadoresLenguajes\Documentacion\Anexo\allTests\Prueba 03\Parse2.txt


V (39)
W (40)
identificador
opRelacional2
V (39)
W (43)
cteEntera
cerrarParentesis
S (21)
return
X (49)
E (31)
R (33)
U (36)
V (39)
W (43)
cteEntera
puntoYcoma
C (15)
B (25)
S (21)
return
X (49)
E (31)
R (33)
U (36)
V (37)
V (39)
W (40)
identificador
opAritmetico1
W (42)
identificador
abrirParentesis
L (45)
E (31)
R (33)
U (36)
V (38)
V (39)
W (40)
identificador
opAritmetico2
W (43)
cteEntera
Q (48)
lambda
cerrarParentesis
puntoYcoma
C (16)
lambda
cerrarCorchete
P (2)
B (25)
S (18)
identificador
abrirParentesis
L (45)
E (31)
R (33)
U (36)
V (39)
W (44)
cadena
Q (47)
coma
E (31)
R (33)
U (36)
V (39)
W (44)
cadena
Q (47)
coma
E (31)
R (33)
U (36)
V (39)
W (42)
identificador
abrirParentesis
L (45)
E (31)
R (33)
U (36)
V (39)
W (43)
cteEntera
Q (48)
lambda
cerrarParentesis
Q (48)
lambda
cerrarParentesis
puntoYcoma
P (4)
lambda

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

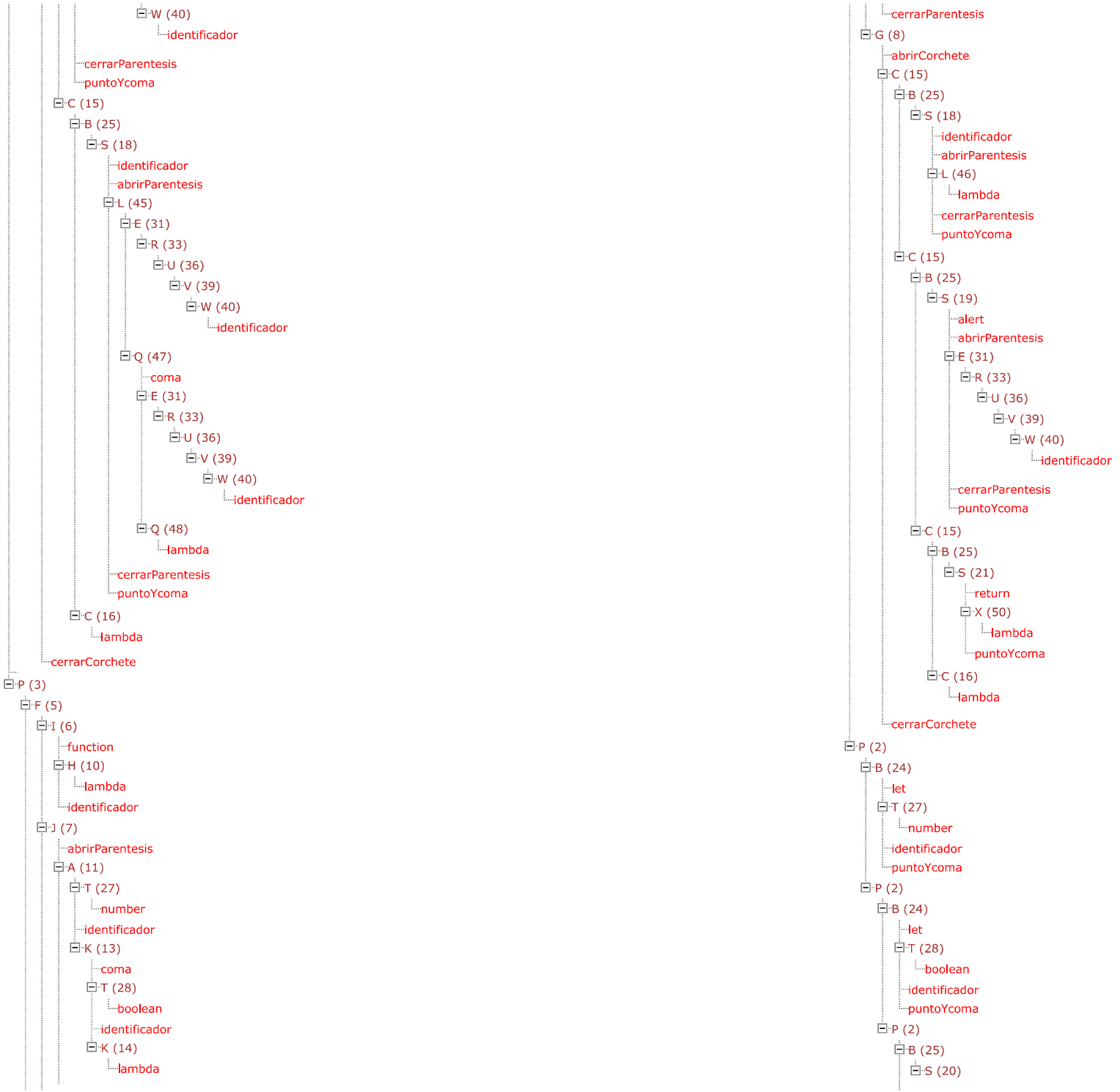

V (39)
W (40)
identificador
Q (48)
lambda
cerrarParentesis
puntoYcoma
P (4)
lambda

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


number
identificador
K (13)
coma
T (28)
boolean
identificador
K (14)
lambda
cerrarParentesis
G (8)
abrirCorchete
C (15)
B (24)
let
T (27)
number
identificador
puntoYcoma
C (15)
B (25)
S (20)
input
abrirParentesis
identificador
cerrarParentesis
puntoYcoma
C (15)
B (25)
S (19)
alert
abrirParentesis
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
cerrarParentesis
puntoYcoma
C (15)
B (25)
S (21)
return
X (49)
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
puntoYcoma
C (16)
lambda
cerrarCorchete
P (3)
F (5)
I (6)
function
H (10)
lambda
identificador
J (7)
abrirParentesis
A (11)
T (27)
number
identificador
K (13)
coma
T (28)
boolean
identificador
K (14)
lambda
cerrarParentesis
G (8)
abrirCorchete
C (15)
B (26)
do
abrirCorchete
C (15)
B (25)
S (19)
alert
abrirParentesis
E (31)
R (33)
U (36)
V (39)
W (43)
cteEntera
cerrarParentesis
puntoYcoma
C (16)
lambda
cerrarCorchete
while
abrirParentesis
E (31)
R (33)
U (36)
V (39)


W (40)
identificador
cerrarParentesis
puntoYcoma
C (15)
B (25)
S (18)
identificador
abrirParentesis
L (45)
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
Q (47)
coma
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
Q (48)
lambda
cerrarParentesis
puntoYcoma
C (16)
lambda
cerrarCorchete
P (3)
F (5)
I (6)
function
H (10)
lambda
identificador
J (7)
abrirParentesis
A (11)
T (27)
number
identificador
K (13)
coma
T (28)
boolean
identificador
K (14)
lambda
cerrarParentesis
G (8)
abrirCorchete
C (15)
B (25)
S (18)
identificador
abrirParentesis
L (46)
lambda
cerrarParentesis
puntoYcoma
C (15)
B (25)
S (19)
alert
abrirParentesis
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
cerrarParentesis
puntoYcoma
C (15)
B (25)
S (21)
return
X (50)
lambda
puntoYcoma
C (16)
lambda
cerrarCorchete
P (2)
B (24)
let
T (27)
number
identificador
puntoYcoma
P (2)
B (24)
let
T (28)
boolean
identificador
puntoYcoma
P (2)
B (25)
S (20)


input
abrirParentesis
identificador
cerrarParentesis
puntoYcoma
P (2)
B (25)
S (19)
alert
abrirParentesis
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
cerrarParentesis
puntoYcoma
P (2)
B (25)
S (22)
identificador
restaAsignacion
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
puntoYcoma
P (2)
B (25)
S (17)
identificador
asignacion
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
puntoYcoma
P (2)
B (25)
S (18)
identificador
abrirParentesis
L (45)
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
Q (47)
coma
E (31)
R (33)
U (36)
V (39)
W (40)
identificador
Q (48)
lambda
cerrarParentesis
puntoYcoma
P (4)
lambda