PROCESADORES DE LENGUAJES

Grupo 64

Christian Paniagua Paniagua x150380

Javier Pérez Martín x150147

Pablo Heras Aranzana x150046

Índice

1. Introducción 5

2. Diseño Analizador Léxico 5

2.1. Tokens 5

2.2. Gramática 6

2.3. Autómata 7

2.4. Acciones semánticas 8

3. Diseño Analizador Sintáctico 9

3.1. Gramática 9

3.2. Demostración LL(1) 10

3.3. Analizador descendente recursivo. 11

4. Diseño Analizador Semántico 15

5. Tabla de símbolos 20

6. Gestor de errores 20

Anexo 22

1. Introducción

En esta memoria se detalla el diseño del compilador de Javasctipt-PL de la asignatura Procesadores de lenguajes 2018-2019. La implementación se ha realizado en el lenguaje Python en su versión 3. Para ejecutar el programa debe tenerse instalado Python 3 y ejecutar desde la carpeta src el comando 'python sintacticosemantico.py'. Posteriormente el programa pedirá al usuario introducir la dirección relaitva a src del fichero con el código fuente a compilar. Los ficheros de salida se almacenarán en la carpeta Salida después de la ejecución.

2. Diseño Analizador Léxico

El analizador léxico se ha implementado usando la herramienta PLY para Python. El diseño es el siguiente:

2.1. Tokens

<cteent, número>

<CAD, "lexema">

<ctebool, 1> (true)

<ctebool, 0> (false)

<MAS, -> (+)

<MENOS, -> (-)

<MUL, -> (*)

<DIV, -> (/)

<MOD, -> (%)

<AND, -> (&&)

<LLAVA, -> ({)

<LLAVC, -> (})

<PARA, -> (( )

<PARC, -> ( ))

<FIN, - > (;)

<SIG, - > (,)

<OR, -> (||)

<NOT, -> (!)

<ASIGR, -> (-=)

<ASIG, - > (=)

<OPMAY, - > (>)

<OPMEN, - > (<)

<OPIG, - > (==)

<OPDISTINTO,- > (!=)

<DOSPUNTOS, - > (:)

<PR, 1 > (int)

<PR, 2 > (bool)

<PR, 3 > (string)

<PR, 4 > (if)

<PR, 5 > (default)

<PR, 6 > (break)

<PR, 7 > (return)

<PR, 8 > (function)

<PR, 9 > (var)

<PR, 10 > (switch)

<PR, 11 > (case)

<PR, 12 > (print)

<PR, 13 > (prompt)

<Id, pos_ts> (siendo "pos_ts" la posición que ocupa el id dentro de la tabla de símbolos)

2.2. Gramática

S --> del S | d E | 'C | l I | / N | = D | : | + | - D1 | & J | '|' K | * | [|] | ; | , | ! D2 | < D3 | D4 >
C --> \ C' | oc1 C | '

C'--> 't' C | 'n' C | ' C | '' C | \ C

D --> = | λ

D1 --> = | λ

D2 --> = | λ

D3 --> = | λ

D4 --> = | λ

E --> d E | λ

I --> l I | d I | _ I | λ

J --> &

K --> '|'

N --> * N' | λ

N' --> * N'' | oc2 N'

N''--> * N'' | / S | oc3 N'

Definiciones:

del={esp, tab, eol}

d = {0..9}

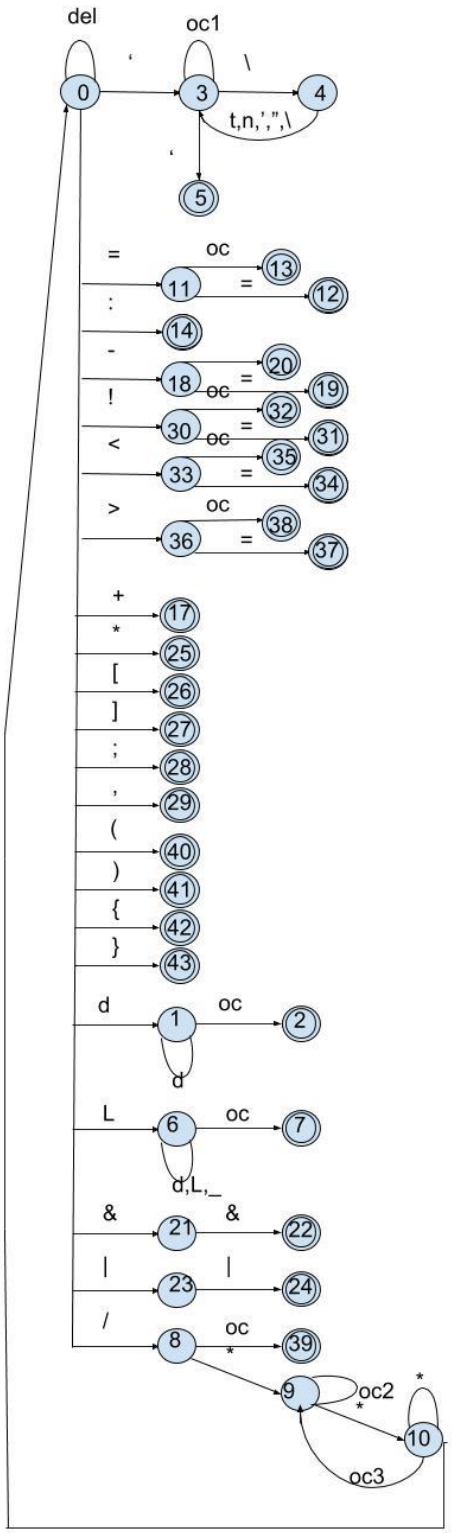
l = {a..z, A..Z}

oc1 = {cualquier carácter imprimible} - {\}

oc2 = {cualquier carácter imprimible} - {*}

oc3 = {cualquier carácter imprimible} - {/,*}

2.3. Autómata


del
oc1
0
3
4
5
t,n,',",\
=
oc
11
13
12
:
14
-
18
20
19
!
30
32
31
<
33
35
34
>
36
38
37
+
17
*
25
[
26
]
27
;
28
,
29
(
40
)
41
{
42
}
43
d
1
2
L
6
7
d,L,_
&
21
22
|
23
24
/
8
39
9
oc2
10
oc3

2.4. Acciones semánticas

L=Leer_carácter()
0:1 -->L, valor = valor_ascii(d)
1:1 -->L, valor = valor*10 + valor_ascii(d)
1:2 -->if(valor > 32767) then **ERROR**
else **G.Token(<ENT, valor>)**
0:3 -->L,lexema = carácter_ascii(‘)
3:3 -->L,lexema = lexema ⊕ carácter ascii(oc1)
3:4 -->L,lexema = lexema ⊕ carácter ascii(\ )
4:3 --> L,lexema = lexema ⊕ {
carácter ascii(t)
carácter ascii(n)
carácter ascii(‘)
carácter ascii(“)
carácter ascii(\ )
}
3:5-->L, lexema = lexema ⊕ carácter ascii(‘)
0:6-->L, lexema = carácter_asciii(l)
6:6-->L, lexema = lexema ⊕ {
carácter ascii(l)
carácter ascii(d)
carácter ascii(-)
}
6:7--> If (pos = TS(lexema)
then G.token(< Id , pos>)
Else if (pos = T.palreservada(lexema))
then **G.token(<PR , pos>)**
else pos = Insertar Ts(lexema),
Gtoken(< Id, pos>)
0:8-->L
0:39-->**G.token(<DIV, ~ >)**
8:9-->L
9:9-->L
9:10-->L
10:10-->L
10:0-->L
0:11-->L
11:12-->L, **G.token(<OP_IG, pos>)**
11:13-->**G.token(<Asig, ~ >)**
0:14-->L, **G.token(<Dos_puntos, ~ >)**
0:17-->L, **G.token(<Mas, ~ >)**
0:18-->L
18:19-->L, **G.token(<Asig_R, ~ >)**
18:20--> **G.token(<Menos, ~ >)**
0:21-->L
21:22-->L, **G.token(<AND, ~ >)**
0:23-->L
23:24-->L, **G.token(<OR, ~ >)**
0:25-->L, **G.token(<MUL, ~ >)**
0:26-->L, **G.token(<CORA, ~ >)**
0:27-->L, **G.token(<CORC, ~ >)**
0:28-->L, **G.token(<FIN, ~ >)**
0:29-->L,
0:30-->L
30:31-->L, **G.token(<OP_DISTINTO, ~ >)**
30:32-->L, **G.token(<NOT, ~ >)**
0:33-->L
33:34-->L, **G.token(<OP_MEN_IG, ~ >)**
33:35-->L, **G.token(<OP_MEN, ~ >)**
0:36-->L
36:38-->L, **G.token(<OP_MAY_IG, ~ >)**
36:37-->L, **G.token(<OP_MAY, ~ >)**
0:40-->L, **G.token(<PARA, ~ >)**
0:41-->L, **G.token(<PARC, ~ >)**
0:42-->L, **G.token(<LLAVA, ~ >)**
0:43-->L, **G.token(<LLAVC, ~ >)**

3. Diseño Analizador Sintáctico

3.1. Gramática

Terminales = { var id ; if ( ) while } { switch case break default int bool string return print
prompt function , || && == != < > -= ! = + - * / % CAD : cteent ctebool
}

NoTerminales = { P B T S _S F H L Q A K X C E _E R _R U _U V _V W _W Z _Z G _G D }
Axioma = P

Producciones = {

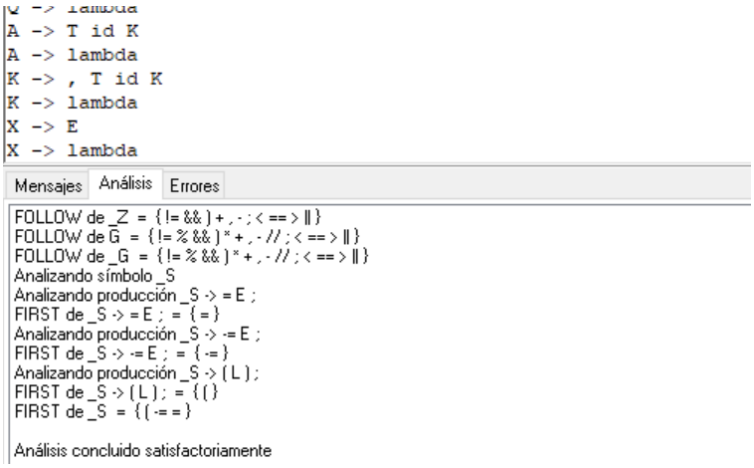
P -> B P
P -> F P
P -> lambda
B -> var T id ;
B -> if (E) S
B -> while (E) { C }
B -> switch (E) { D }
B -> S
T -> int
T -> bool
T -> string
S -> id _S
S -> return X ;
S -> print (E) ;
S -> prompt (id) ;
S -> break ;
_S -> = E ;
_S -> -= E ;
_S -> (L) ;
F -> function H id (A) { C }
H -> T
H -> lambda
L -> E Q
L -> lambda
Q -> , E Q
Q -> lambda
A -> T id K
A -> lambda
K -> , T id K
K -> lambda
X -> E
X -> lambda
C -> B C
C -> lambda
E -> R _E
_E -> || R _E
_E -> lambda
R -> U _R
_R -> && U _R
_R -> lambda
U -> V _U
_U -> == V _U
_U -> != V _U
_U -> lambda
V -> W _V
_V -> > W _V
_V -> < W _V
_V -> lambda
W -> Z _W
_W -> + Z _W
_W -> - Z _W
_W -> lambda
Z -> G _Z
_Z -> * G _Z
_Z -> / G _Z
_Z -> % G _Z
_Z -> lambda
G -> ! G
G -> id _G
G -> (E)
G -> cteent
G -> CAD
G -> ctebool
_G -> (L)
_G -> lambda
D -> case cteent : C D
D -> default : C
}

3.2. Demostración LL(1)

Puesto que nuestro método de análisis sintáctico es el "Descendente Recursivo" nuestra gramática debe ser una gramática LL(1). Este tipo de gramáticas son no ambiguas, no recursivas por la izquierda y factorizadas por la izquierda. A parte de estas condiciones necesarias, todas las producciones de la gramática con mismo antecedente deben cumplir (dos a dos) la condición LL(1):

- Si A → α_1, A → α_2 ∈ G => First(α_1) ∩ First(α_2) = ∅ Es decir, que no tengan nada en común.
- Si A→ α_1, A → α_2 ∈ G y λ ∈ First(α_1) => Follow(A) ∩ First(α_2) = ∅
  Pertenece a una de las reglas.

Tras eliminar las recursividades por la izquierda y factorizar nuestra gramática empleamos la herramienta SDGLL1 proporcionada en la web de la asignatura para comprobar que nuestra gramática es LL(1). Tras analizar la gramática en la herramienta obtuvimos los First y Follows de las distintas reglas, así como la confirmación de que la gramática es LL(1).



3.3. Analizador descendente recursivo.

A continuación, mostramos el código de los procedimientos que realizarán el análisis sintáctico:

```
print("Des ");

Proc P;
	if(st ϵ First(B P)={ break id if print prompt return switch var while })
then
	{
		print("1");
		B();
		P();
	}
	else if(st ϵ First(F P)={ function }) then
	{
		print("2");
		F();
		P();
	}
	else if(st ϵ Follow(P)={ $ })
	{
		print("3");
	}
	else ERROR;
END;

Proc B;
	if(st ϵ First(var T id ;)={ var }) then
	{
		print("4");
		EqT(var);
		T();
		EqT(id);
		EqT(;);
	}
	else if(st ϵ First(if ( E ) S)={ if }) then
	{
		print("5");
		EqT(if);
		EqT(();
		E();
		EqT());
		S();
	}
	else if(st ϵ First(while ( E ) { C })={ while }) then
	{
		print("6");
		EqT(while);
		EqT(();
		E();
		EqT());
		EqT({);
		C();
		EqT(});
	}
	else if(st ϵ First(switch ( E ) { D })={ switch }) then
	{
		print("7");
		EqT(switch);
		EqT(();
		E();
		EqT());
		EqT({);
		D();
		EqT(});
	}
	else if(st ϵ First(S)={ break id print prompt return }) then
	{
		print("8");
		S();
	}
	else ERROR;
END;
```

```
Proc T;
	if(st ϵ First(int)={ int }) then
	{
		print("9");
		EqT(int);
	}
	else if(st ϵ First(bool)={ bool }) then
	{
		print("10");
		EqT(bool);
	}
	else if(st ϵ First(string)={ string }) then
	{
		print("11");
		EqT(string);
	}
	else ERROR;
END;


Proc S;
	if(st ϵ First(id S')={ id }) then
	{
		print("12");
		EqT(id);
		S'();
	}
	else if(st ϵ First(return X ;)={ return }) then
	{
		print("13");
		EqT(return);
		X();
		EqT(;);
	}
	else if(st ϵ First(print ( E ) ;)={ print }) then
	{
		print("14");
		EqT(print);
		EqT(();
		E();
		EqT());
		EqT(;);
	}
	else if(st ϵ First(prompt ( id ) ;)={ prompt }) then
	{
		print("15");
		EqT(prompt);
		EqT(();
		EqT(id);
		EqT());
		EqT(;);
	}
	else if(st ϵ First( break ;)={ break }) then
	{
		print("16");
		EqT(break);
		EqT(;);
	}
	else ERROR;
END;
```

```
Proc S';
	if(st ϵ First(= E ;)={ = }) then
	{
		print("17");
		EqT(=);
		E();
		EqT(;);
	}
	else if(st ϵ First(-= E ;)={ -= }) then
	{
		print("18");
		EqT(-=);
		E();
		EqT(;);
	}
	else if(st ϵ First(( L ) ;)={ ( }) then
	{
		print("19");
		EqT(();
		L();
		EqT());
		EqT(;);
	}
	else ERROR;
END;

Proc F;
	if(st ϵ First(function H id ( A ) { C })={ function }) then
	{
		print("20");
		EqT(function);
		H();
		EqT(id);
		EqT(();
		A();
		EqT());
		EqT({);
		C();
		EqT(});
	}
	else ERROR;
END;

Proc H;
	if(st ϵ First(T)={ int bool string }) then
	{
		print("21");
		T();
	}
	else if(st ϵ Follow(H)={ id }) then
	{
		print("22");
	}
	else ERROR;
END;

Proc L;
	if(st ϵ First(E Q)={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("23");
		E();
		Q();
	}
	else if(st ϵ Follow(L)={ ) }) then
	{
		print("24");
	}
	else ERROR;
END;

Proc Q;
	if(st ϵ First(, E Q)={ , }) then
	{
		print("25");
		EqT(,);
		E();
		Q();
	}
	else if(st ϵ Follow(Q)={ ) }) then
	{
		print("26");
	}

	else ERROR;
END;

Proc A;
	if(st ϵ First(T id K)={ int bool string }) then
	{
		print("27");
		T();
		EqT(id);
		K();
	}
	else if(st ϵ Follow(A)={ ) ; }) then
	{
		print("28");
	}
	else ERROR;
END;

Proc K;
	if(st ϵ First(, T id K)={ , }) then
	{
		print("29");
		EqT(,);
		T();
		EqT(id);
		K();
	}
	else if(st ϵ Follow(K)={ ) }) then
	{
		print("30");
	}
	else ERROR;
END;

Proc X;
	if(st ϵ First(E)={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("31");
		E();
	}
	else if(st ϵ Follow(X)={ ; }) then
	{
		print("32");
	}
	else ERROR;
END;

Proc C;
	if(st ϵ First(B C)={ break id if print prompt return switch var while })
then
	{
		print("33");
		B();
		C();
	}
	else if(st ϵ Follow(C)={ case } default }) then
	{
		print("34");
	}
	else ERROR;
END;

Proc E;
	if(st ϵ First(R E')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("35");
		R();
		E'();
	}
	else ERROR;
END;

Proc E';
	if(st ϵ First(| | R E')={ | | }) then
	{
		print("36");
		EqT(| |);
		R();
		E'();
	}
	else if(st ϵ Follow(E')={ ) , ; }) then
	{
		print("37");
	}
	else ERROR;
END;
```

```
Proc R;
	if(st ϵ First(U R')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("38");
		U();
		R'();
	}
	else ERROR;
END;

Proc R';
	if(st ϵ First(&& U R')={ && }) then
	{
		print("39");
		EqT(&&);
		U();
		R'();
	}
	else if(st ϵ Follow(R')={ ) , ; || }) then
	{
		print("40");
	}
	else ERROR;
END;

Proc U;
	if(st ϵ First(V U')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("41");
		V();
		U'();
	}
	else ERROR;
END;

Proc U';
	if(st ϵ First(== V U')={ == }) then
	{
		print("42");
		EqT(==);
		V();
		U'();
	}
	else if(st ϵ First(!= V U')={ != }) then
	{
		print("43");
		EqT(!=);
		V();
		U'();
	}
	else if(st ϵ Follow(U')={ && ) , ; || }) then
	{
		print("44");
	}
	else ERROR;
END;

Proc V;
	if(st ϵ First(W V')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("45");
		W();
		V'();
	}
	else ERROR;
END;

Proc V';
	if(st ϵ First(> W V')={ > }) then
	{
		print("46");
		EqT(>);
		W();
		V'();
	}
	else if(st ϵ First(< W V')={ < }) then
	{
		print("47");
		EqT(<);
		W();
		V'();
	}
	else if(st ϵ Follow(V')={ != && ) , ; == || }) then
	{
		print("48");
	}

	else ERROR;
END;
```

```
Proc W;
	if(st ϵ First(Z W')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("49");
		Z();
		W'();
	}
	else ERROR;
END;

Proc W';
	if(st ϵ First(+ Z W')={ + }) then
	{
		print("50");
		EqT(+);
		Z();
		W'();
	}
	else if(st ϵ First(- Z W')={ - }) then
	{
		print("51");
		EqT(-);
		Z();
		W'();
	}
	else if(st ϵ Follow(W')={ != && ) , ; < == > || }) then
	{
		print("52");
	}
	else ERROR;
END;

Proc Z;
	if(st ϵ First(G Z')={ ! ( cte_bool CAD cte_ent id }) then
	{
		print("53");
		G();
		Z'();
	}
	else ERROR;
END;

Proc Z';

	if(st ϵ First(* G Z')={ * }) then
	{
		print("54");
		EqT(*);
		G();
		Z'();
	}
	else if(st ϵ First(/ G Z')={ / }) then
	{
		print("55");
		EqT(/);
		G();
		Z'();
	}
	else if(st ϵ First(% G Z')={ % }) then
	{
		print("56");
		EqT(%);
		G();
		Z'();
	}
	else if(st ϵ Follow(Z')={ != && ) + , - ; < == > || }) then
	{
		print("57");
	}
	else ERROR;
END;
```

```
Proc G;
		if(st ϵ First(! G)={ ! }) then
		{
			print("58");
			EqT(!);
			G();
		}
		else if(st ϵ First(id G')={ id }) then
		{
			print("59");
			EqT(id);
			G'();
		}
		else if(st ϵ First(( E ))={ ( }) then
		{
			print("60");
			EqT((();
			E();
			EqT());
		}
		else if(st ϵ First(cte_ent)={ cte_ent }) then
		{
			print("61");
			EqT(cte_ent);
		}
		else if(st ϵ First(CAD)={ CAD }) then
		{
			print("62");
			EqT(CAD);
		}
		else if(st ϵ First(cte_bool)={ cte_bool }) then
		{
			print("63");
			EqT(cte_bool);
		}
		else ERROR;
END;

Proc G';
		if(st ϵ First(( L ))={ ( }) then
		{
			print("64");
			EqT((();
			L();
			EqT());
		}
		else if(st ϵ Follow(G')={ != % && ) * + , - / ; < == > | | }) then
		{
			print("65");
		}
		else ERROR;
END;

Proc D;
		if(st ϵ First(case cte_ent : C D)={ case }) then
		{
			print("66");
			EqT(case);
			EqT(cte_ent);
			EqT(:);
			C();
			D();
		}
		else if(st ϵ ) then
		{
			print("67");
			EqT(default);
			EqT(:);
			C();
		}
		else ERROR;
END;
```

4. Diseño Analizador Semántico

A continuación, la traducción dirigida por la sintaxis en formato de esquema de traducción para el análisis semántico:

P -> {B.tipoSwitch=false} B P

P -> F P **{}**

P -> lambda **{}**

B -> var {declaración=true} T id {insertaTipoTS(id.lex, T.Tipo); insertaDespTS(id.pos, T.desp);} {declaración=false}; {B.tipoRet='tipo_vacio' }

B -> if (E) { if(E.tipo!=bool) then ERROR #1 else S.switch = B.switch } S

{ B.tipoRet = S.TipoRet; }

B -> while (E) { if(E.tipo!=bool) then ERROR #2 elseC.switch = B.switch } { C }
{B.tipoRet=C.tipoRet}

B -> switch (E) { if(E.tipo!=int) ERROR #3 } { D } {B.tipoRet=D.tipoRet}

B -> {S.switch = B.switch} S {B.tipoRet=S.tipoRet}

T -> int **{T.Tipo = int}**

T -> bool **{T.Tipo = bool}**

T -> string **{T.Tipo = string}**

S -> id _S { if(buscaTipo(id.pos) ¡= _S.tipo->t ERROR #4 S.tipoRet=tipo_vacio;
else if(buscaTipo(id.pos) == _S.tipo then ERROR #5; S.tipoRet=tipo_vacio;
else if(buscaTipo(id.pos) == tipo_vacio then insertaTipoTS(id.pos, int);
if(_S.tipo!=id.tipo) then ERROR #6 ;S.tipoRet=tipo_vacio; }
else Error}

S --> return X {if TSactual != then ERROR #7 TSL then S.tipoRet = X.tipo} ;

S -> print (E) {S.Tipo= if(E.tipo IN {int,bool,string} S.tipoRet=tipo_vacio; else ERROR #8 } ;

(E puede ser de cualquiera de los 3 tipos)

S -> prompt (id) {S.Tipo= if(buscaTipo(id.pos) IN {int, string} then S.tipoRet=tipo_vacio; else ERROR #9)} ;

(id tiene que ser entero o cadena)

S -> break ; {if(S.switch = false then ERROR #10 (“Break fuera de switch”)

else S.tipoRet=tipo_vacio;}

_S -> = E ; {_S.tipo = E.tipo}

_S -> -= E ; {_S.tipo = if(E.tipo==int) then int else ERROR #11 }

_S -> (L) ; {_S.tipo=L.tipo}

F -> function {declaración=true} H id {TL=creaTSL(); desp=0;} (A) {declaración=false} {insertaTipoTS(id.pos, A.Tipo->H.Tipo); declaración=false} { C {if(C.tipoRet != H.tipo) then ERROR #12; destruirTSL()} }

H -> T {H.Tipo = T.Tipo}

H -> lambda {H.Tipo = tipo_vacio}

L -> E Q {L.Tipo = E.tipo x Q.Tipo}

L -> lambda {L.tipo = tipo_vacio}

Q -> , E Q {Q.Tipo = E.tipo x Q_1.Tipo}

Q -> lambda {Q.Tipo = tipo_vacio}

A -> T id {insertatipoTS(id.pos,T.tipo); insertaDespTS(id.pos,desp); desp=desp++} K {A.Tipo = T.tipo x K.tipo}

A -> lambda {A.Tipo = tipo_vacio}

K -> , T id {insertatipoTS(id.pos,T.tipo); insertaDespTS(id.pos,desp); desp=desp++} K_1 {K.Tipo = T.tipo x K_1.tipo}

K -> lambda {K.Tipo = tipo_vacio}

X -> E {X.Tipo = E.Tipo}

X -> lambda {X.Tipo = tipo_vacio}

C -> {B.switch = C.switch; C_1.switch = B.switch} B C_1

{ C.TipoRet = if(B.tipoRet == C_1. tipoRet) then B.tipoRet

Else if(B.tipoRet == tipo_vacio) then C_1.tipoRet

Else if(C_1.tipoRet == tipo_vacio) then B.tipoRet

Else Error #13 }

C -> lambda {C.TipoRet = tipo_vacio}

E -> R _E **{E.Tipo = if(_E.Tipo== tipo_vacio) then R.Tipo else**

if(_E.Tipo== bool AND R.Tipo == bool) then bool else

ERROR #14 }

_E -> || R _E **{_E.Tipo = if((R.Tipo==bool) AND ($_{E_1}$.Tipo==bool OR tipo_vacio))then bool else ERROR #14}**

_E -> lambda **{E.Tipo = tipo_vacio }**

E puede ser: int, string, bool, Error

_E puede ser: bool, vacío, Error

R -> U _R **{R.Tipo = if(_R.Tipo== tipo_vacio) then U.Tipo else**

if(_R.Tipo== bool AND U.Tipo == bool) then bool else ERROR #14}

_R -> && U _R **{_R.Tipo = if((U.Tipo==bool) AND ($_{R_1}$.Tipo==bool OR tipo_vacio))then bool else ERROR #14}**

_R -> lambda **{R.Tipo = tipo_vacio }**

R puede ser: int, string, bool, Error

_R puede ser: bool, vacío, Error

U -> V _U **{U.Tipo = if(_U.Tipo== tipo_vacio) then V.Tipo else**

if(_U.Tipo== bool AND V.Tipo == int) then bool else

ERROR #14}

_U -> == V _U{_U.Tipo=if((V.Tipo==int) AND ($_{U_1}$.Tipo==tipo_vacio))then bool else ERROR #14}

_U -> != V _U {_U.Tipo=if((V.Tipo==int) AND ($_{U_1}$.Tipo==tipo_vacio))then bool else ERROR #14}

_U -> lambda **{_U.Tipo = tipo_vacio }**

U puede ser: int, string, bool, Error

_U puede ser: bool, vacío, Error

V -> W _V **{V.Tipo = if(_V.Tipo== tipo_vacio) then W.Tipo else**

if(_V.Tipo== bool AND W.Tipo == int) then bool else

ERROR #14 }

_V -> > W _V {_V.Tipo=if((W.Tipo==int) AND (_V$_1$.Tipo==tipo_vacio))then bool else

ERROR #14}

_V -> < W _V {_V.Tipo=if((W.Tipo==int) AND (_V$_1$.Tipo==tipo_vacio))then bool else

ERROR #14}

_V -> lambda {_V.Tipo = tipo_vacio }

V puede ser: int, string, bool, Error

_V puede ser: bool, vacío, Error

W -> Z _W {W.Tipo = if(_W.Tipo== tipo_vacio) then Z.Tipo else

if(_W.Tipo== int AND Z.Tipo == int) then int else

ERROR #14}

_W -> + Z _W {_W.Tipo = if((Z.Tipo==int) AND (_W$_1$.Tipo==int OR tipo_vacio))then int else

ERROR #14}

_W -> - Z _W {_W.Tipo = if((Z.Tipo==int) AND (_W$_1$.Tipo==int OR tipo_vacio))then int else

ERROR #14}

_W -> lambda {_W.Tipo = tipo_vacio }

W puede ser: int, string, bool, Error

_W puede ser: int, vacío, Error

Z -> G _Z {Z.Tipo = if(_Z.Tipo== tipo_vacio) then G.Tipo else

if(_Z.Tipo== int AND G.Tipo == int) then int else

ERROR #14 }

_Z -> * G _Z {_Z.Tipo = if((G.Tipo==int) AND (_Z_1.Tipo==int OR tipo_vacio)) then int else ERROR #14}

_Z -> / G _Z {_Z.Tipo = if((G.Tipo==int) AND (_Z_1.Tipo==int OR tipo_vacio)) then int else ERROR #14}

_Z -> % G _Z_1 {_Z.Tipo = if((G.Tipo==int) AND (_Z_1.Tipo==int OR tipo_vacio)) then int else ERROR #14}

_Z -> lambda {_Z.Tipo = tipo_vacio }

Z puede ser: int, string, bool, Error

_Z puede ser: int, tipo_vacio, Error

G-> !G_1 {G.Tipo = if(G_1.Tipo==bool) then bool else ERROR #15}

G -> id _G {G.Tipo = if(_G.Tipo== tipo_vacio) then buscaTipoTS(id.pos) else If (buscaTipoTS(id.pos)==_G.Tipo->t then t else ERROR #14}

G -> (E) {G.Tipo = E.Tipo}

G -> cteent {G.Tipo = int}

G -> CAD {G.Tipo = string}

G -> ctebool {G.Tipo = bool}

_G -> (L) {_G.Tipo = L.Tipo}

_G -> lambda {_G.Tipo = tipo_vacio }

G puede ser: int, string, bool, Error

_G puede ser: tipo_vacio, arg1 x arg2 x … x argN

D -> case cteent : {C.switch = true} C D_1

{D.tipoRet=if(C.tipoRet == D1.tipoRet) then C.tipoRet
Else if(C.tipoRet == tipo_vacio) then D1.tipoRet
Else if(D1.tipoRet == tipo_vacio) then C.tipoRet
Else Error}

D -> default : {C.switch = true } C {D.tipoRet = C.tipoRet}

}

5. Tabla de símbolos

La tabla de símbolos consta de dos tablas, una general y otra local. La tabla general almacena los identificadores globales tanto de variables como de funciones. Los atributos que almacena de una variable son: tipo ('int','bool','string') y el desplazamiento que refleja el tamaño reservado para el valor almacenado en dicha variable dependiendo del tipo de la variable. Los atributos que almacena para las funciones son: parámetros de entrada (de los cuales almacena el tipo y el modo de paso de parámetros que va a ser siempre 1 que implica paso por valor), etiqueta de la función, tipo de retorno y numero de parámetros. Cada tabla local se crea al declarar una función. Almacena los argumentos de entrada, el tipo de retorno y las variables locales declaradas dentro de la función. Una vez terminada la declaración de la función ésta se destruye y vuelca su información en el fichero de tabla de símbolos.

La tabla incluye un flag de declaración que modifica el comportamiento de la inserción de identificadores en la tabla de símbolos. Si el flag está activado se inserta el valor en la tabla de símbolos actual si no se ha hecho previamente. Si está desactivado pero se encuentra un identificador no declarado previamente, éste se almacena en la tabla global con tipo 'int'.

6. Gestor de errores

En este apartado se muestran los distintos mensajes de error que puede mostrar el gestor de errores. Nuestro compilador, en el momento en que detecta un fallo, detiene el análisis, muestra por pantalla el mensaje de error correspondiente y vuelca la información en los ficheros de salida.

Errores Léxicos:

Error #1: La cte entera supera el tamaño máximo permitido.

Error #2: carácter no permitido.

Errores Sintácticos:

Error #1: sintaxis incorrecta.

Error #2: El tipo de retorno de una función debe ser un tipo básico o vacío.

Error #3: Error en llamada de función.

Error #4: Parámetros formales de declaración de función incorrectos.

Error #5: Expresión mal construida.

Error #6: sentencia case/default incorrecta.

Error #7: token recibido distinto al esperado.

Errores Semánticos:

Error #1: La condición del 'if' debe ser de tipo booleano.

Error #2: La condición del 'while' debe ser de tipo booleano.

Error #3: La condición del 'switch' debe ser de tipo entero.

Error #4: Llamada a función con parámetro incorrectos.

Error #5: La variable no es de tipo función y por lo tanto no puede hacerse una llamada.

Error #6: Tipo erróneo en asignación.

Error #7: Sentencia return fuera de función.

Error #8: print() debe llamarse con un argumento de tipo entero, booleano o string.

Error #9: prompt() debe llamarse con una variable de tipo entero o string como argumento.

Error #10: break fuera de sentencia switch.

Error #11: la sentencia de asignación con resta solo admite operandos enteros

Error #12: el tipo de retorno de la función declarado: '+hTipo+' no coincide con el tipo devuelto.

Error #13: sentencias de retorno en función inconsistentes.

Error #14: expresión incorrecta.

Anexo

Pruebas Correctas:

Prueba 1:

```
var string texto;/* comentarioooooooooooooooooooooooooo
oooooooooooooooooooooooooo


*/
//pepe
 function string pideTexto ()
{
        print ( 'Introduce un texto' );
        prompt (texto);
  return 'cad';
}
texto = pideTexto();
function imprime (string msg, int p, int p1)
{
        print (msg);
}
 pideTexto();
var string textoAux;
textoAux = textoAux;
//imprime (textoAux);
```

Tokens

<PR,9>
<PR,3>
<ID,10>
<FIN,>
<PR,8>
<PR,3>
<ID,11>
<PARA,>
<PARC,>
<LLAVA,>
<PR,12>
<PARA,>
<CAD,"Introduce un texto">
<PARC,>
<FIN,>
<PR,13>
<PARA,>
<ID,10>
<PARC,>
<FIN,>
<PR,7>
<CAD,"cad">
<FIN,>
<LLAVC,>
<ID,10>
<ASIG,>
<ID,11>
<PARA,>
<PARC,>
<FIN,>
<PR,8>
<ID,12>
<PARA,>
<PR,3>
<ID,20>
<SIG,>
<PR,1>
<ID,21>
<SIG,>
<PR,1>
<ID,22>
<PARC,>
<LLAVA,>
<PR,12>
<PARA,>
<ID,20>
<PARC,>
<FIN,>
<LLAVC,>
<ID,11>
<PARA,>
<PARC,>
<FIN,>
<PR,9>
<PR,3>
<ID,13>
<FIN,>
<ID,13>
<ASIG,>
<ID,13>
<FIN,>

Árbol


Árbol resultado de:
Gramática: C:\Users\PABLO\Desktop\Arbol1.txt
Parse: C:\Users\PABLO\Proyectos\PDLG64\src\Salida\parse.txt
P (1)
B (4)
var
T (11)
string
id
;
P (2)
F (20)
function
H (21)
T (11)
string
id
(
A (28)
lambda
)
{
C (33)
B (8)
S (14)
print
(
E (35)
R (38)
U (41)
V (45)
W (49)
Z (53)
G (62)
CAD
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (37)
lambda
)
;
C (33)
B (8)
S (15)
prompt
(
id
)
;
C (33)
B (8)
S (13)
return
X (31)
E (35)
R (38)
U (41)
V (45)
W (49)
Z (53)
G (62)
CAD
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (37)
lambda
;
C (34)
lambda
}
P (1)
B (8)
S (12)
id
_S (17)
=
E (35)
R (38)
U (41)
V (45)
W (49)
Z (53)
G (59)
id
_G (64)
(
L (24)
lambda
)
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (37)
lambda
;
P (2)
F (20)
function
H (22)
lambda
id
(
A (27)
T (11)
string
id
K (29)
,
T (9)
int
id
K (29)
,
T (9)
int
id
K (30)
lambda
)
{
C (33)
B (8)
S (14)
print
(
E (35)
R (38)
U (41)
V (45)
W (49)
Z (53)
G (59)
id
_G (65)
lambda
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (37)
lambda
)
;
C (34)
lambda
}
P (1)
B (8)
S (12)
id
_S (19)
(
L (24)
lambda
)
;
P (1)
B (4)
var
T (11)
string

Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'texto'
ATRIBUTOS:
+despl: 0
+tipo: 'string'
---
* LEXEMA: 'pideTexto'
ATRIBUTOS:
+numParam: 0
+TipoRetorno: 'string'
+EtiqFuncion: 'ETIpideTexto2'
---
* LEXEMA: 'imprime'
ATRIBUTOS:
+numParam: 3
+TipoRetorno: 'tipo_vacio'
+EtiqFuncion: 'ETIimprime3'
+TipoParam1: 'string'
+ModoParam1: 1
+TipoParam2: 'int'
+ModoParam2: 1
+TipoParam3: 'int'
+ModoParam3: 1
---
* LEXEMA: 'textoAux'
ATRIBUTOS:
+despl: 130
+tipo: 'string'

TSL de la funcion 'pideTexto' #2:

TSL de la funcion 'imprime' #3:
* LEXEMA: 'msg' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'string'
---
* LEXEMA: 'p' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 128
+tipo: 'int'
---
* LEXEMA: 'p1' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 130
+tipo: 'int'

Prueba 2:

```
var int a; var bool b;
b = false && 2 == 2%2 || true && a > 34 + 3*a;
var bool c;
c = a  >  a;
if (c) a  -=  1;
if (2  <  a) a  -=  4;
 a = a  +  4;
print (a);
print (b);
```

Tokens

<PR,9>
<PR,1>
<ID,10>
<FIN,>
<PR,9>
<PR,2>
<ID,11>
<FIN,>
<ID,11>
<ASIG,>
<ctebool,0>
<AND,>
<cteent,2>
<OPIG,>
<cteent,2>
<MOD,>
<cteent,2>
<OR,>
<ctebool,1>
<AND,>
<ID,10>
<OPMAY,>
<cteent,34>
<MAS,>
<cteent,3>
<MUL,>
<ID,10>
<FIN,>
<PR,9>
<PR,2>
<ID,12>
<FIN,>
<ID,12>
<ASIG,>
<ID,10>
<OPMAY,>
<ID,10>
<FIN,>
<PR,4>
<PARA,>
<ID,12>
<PARC,>
<ID,10>
<ASIGR,>
<cteent,1>
<FIN,>
<PR,4>
<PARA,>
<cteent,2>
<OPMEN,>
<ID,10>
<PARC,>
<ID,10>
<ASIGR,>
<cteent,4>
<FIN,>
<ID,10>
<ASIG,>
<ID,10>
<MAS,>
<cteent,4>
<FIN,>
<PR,12>
<PARA,>
<ID,10>
<PARC,>
<FIN,>
<PR,12>
<PARA,>
<ID,11>
<PARC,>
<FIN,>

Árbol


Árbol resultado de:
Gramática: C:\Users\PABLO\Desktop\Arbol1.txt
Parse: C:\Users\PABLO\Proyectos\PDLG64\src\Salida\parse.txt

Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'a'
ATRIBUTOS:
+despl: 0
+tipo: 'int'
---
* LEXEMA: 'b'
ATRIBUTOS:
+despl: 2
+tipo: 'bool'
---
* LEXEMA: 'c'
ATRIBUTOS:
+despl: 6
+tipo: 'bool'

Prueba 3:

```
var int a;
var int b;
var int c;
print ( 'Introduce el primer operando' );
prompt (a);
print ( 'Introduce el segundo operando' );
prompt (b);
 function int divide (int num1, int num2)
{
        return num1/num2;
}
c = divide (a, b);
 print (c);
```

Tokens

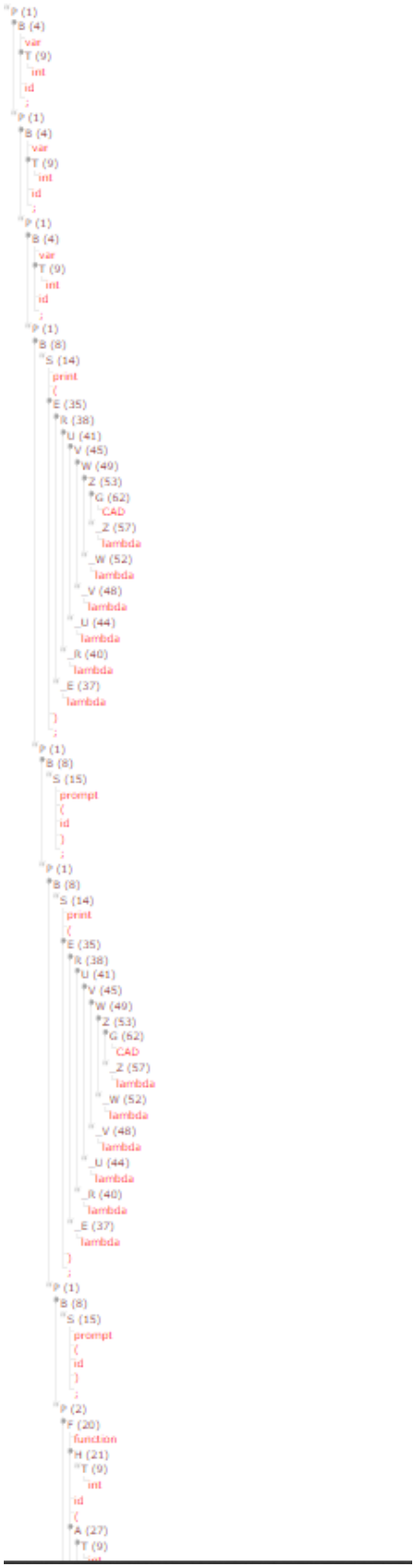
<PR,9>
<PR,1>
<ID,10>
<FIN,>
<PR,9>
<PR,1>
<ID,11>
<FIN,>
<PR,9>
<PR,1>
<ID,12>
<FIN,>
<PR,12>
<PARA,>
<CAD,"Introduce el primer operando">
<PARC,>
<FIN,>
<PR,13>
<PARA,>
<ID,10>
<PARC,>
<FIN,>
<PR,12>
<PARA,>
<CAD,"Introduce el segundo operando">
<PARC,>
<FIN,>
<PR,13>
<PARA,>
<ID,11>
<PARC,>
<FIN,>
<PR,8>
<PR,1>
<ID,13>
<PARA,>
<PR,1>
<ID,20>
<SIG,>
<PR,1>
<ID,21>
<PARC,>
<LLAVA,>
<PR,7>
<ID,20>
<DIV,>
<ID,21>
<FIN,>
<LLAVC,>
<ID,12>
<ASIG,>
<ID,13>
<PARA,>
<ID,10>
<SIG,>
<ID,11>
<PARC,>
<FIN,>
<PR,12>
<PARA,>
<ID,12>
<PARC,>
<FIN,>

Árbol

Árbol resultado de:

Gramática: C:\Users\PABLO\Desktop\Arbol1.txt

Parse: C:\Users\PABLO\Proyectos\PDLG64\src\Salida\parse.txt



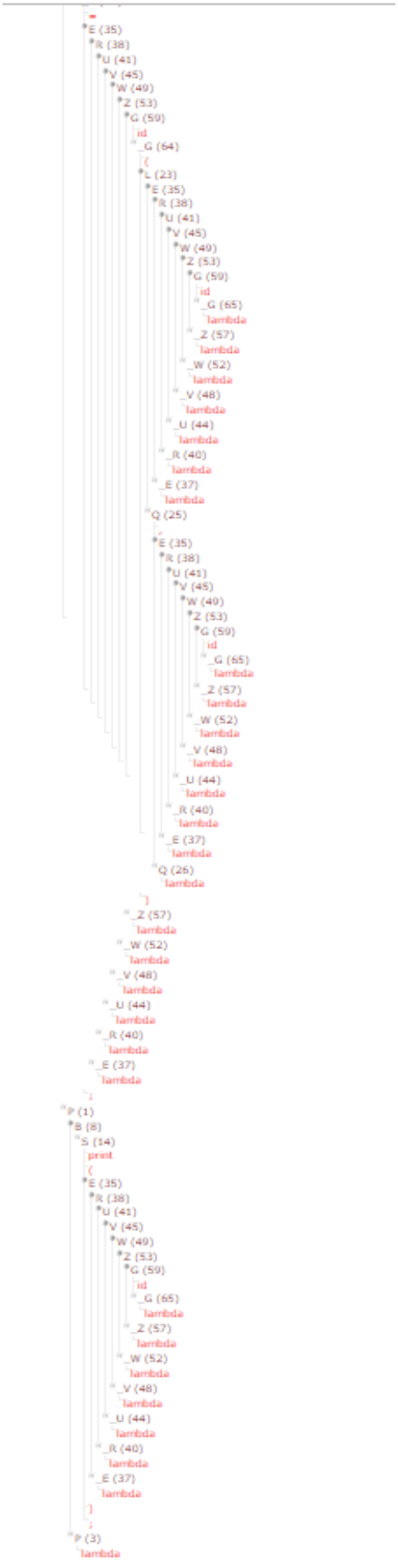


Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'a'
        ATRIBUTOS:
        +despl: 0
        +tipo: 'int'
        ----
* LEXEMA: 'b'
        ATRIBUTOS:
        +despl: 2
        +tipo: 'int'
        ----
* LEXEMA: 'c'
        ATRIBUTOS:
        +despl: 4
        +tipo: 'int'
        ----
* LEXEMA: 'divide'
        ATRIBUTOS:
        +numParam: 2
        +TipoRetorno: 'int'
        +EtiqFuncion: 'ETIdivide2'
        +TipoParam1: 'int'
        +ModoParam1: 1
        +TipoParam2: 'int'
        +ModoParam2: 1

TSL de la funcion 'divide' #2:
* LEXEMA: 'num1' (parametro de entrada de la funcion)
        ATRIBUTOS:
        +despl: 0
        +tipo: 'int'
        ----
* LEXEMA: 'num2' (parametro de entrada de la funcion)
        ATRIBUTOS:
        +despl: 2
        +tipo: 'int'

Prueba 4:

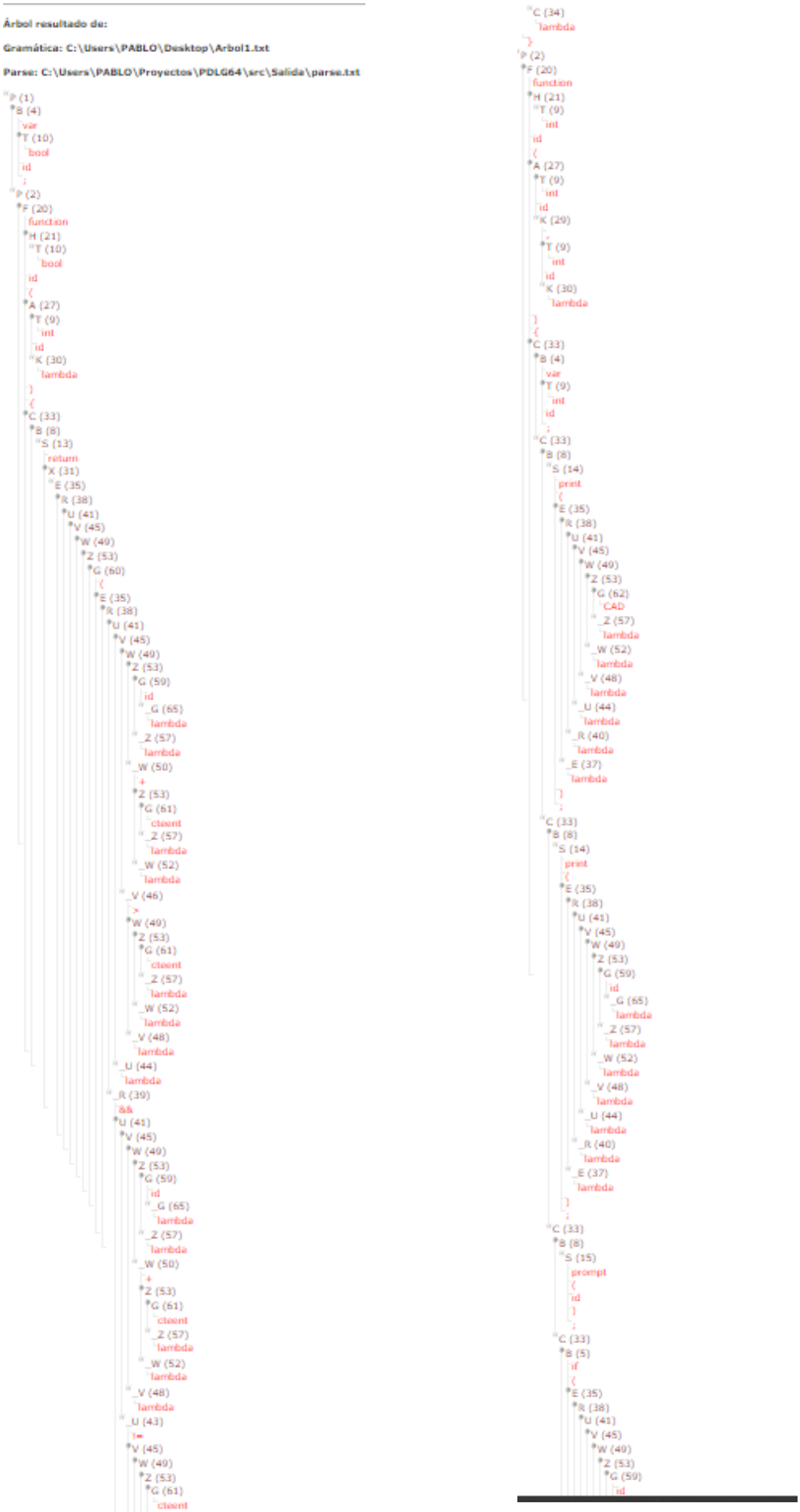
```
var bool boolean;
 function bool bisiesto (int a)
{
        return (a + 4 > 0 && a + 100 != 0 && a % 400 != 0);
}
 function int dias (int m, int a)
{
        var int dd;
        print ( 'di cuantos dias tiene el mes ' );
        print (m);
        prompt(dd);
        if (bisiesto(a)) dd = dd % 1;
        return dd;
}
 function bool esFechaCorrecta (int d, int m, int a)
{
        return m > 1 || m > 12 || d == 1 && d < dias (m, a);
}
 function demo ()
{

        if (esFechaCorrecta(25, 10, 2018)) print ( 'OK' );
  if (true) print ( 'OK' );
}
 var string A_A_A_ ;
demo();
```

Tokens

<PR,9>
<PR,2>
<ID,10>
<FIN,>
<PR,8>
<PR,2>
<ID,11>
<PARA,>
<PR,1>
<ID,20>
<PARC,>
<LLAVA,>
<PR,7>
<PARA,>
<ID,20>
<MAS,>
<cteent,4>
<OPMAY,>
<cteent,0>
<AND,>
<ID,20>
<MAS,>
<cteent,100>
<OPDISTINTO,>
<cteent,0>
<AND,>
<ID,20>
<MOD,>
<cteent,400>
<OPDISTINTO,>
<cteent,0>
<PARC,>
<FIN,>
<LLAVC,>
<PR,8>
<PR,1>
<ID,12>
<PARA,>
<PR,1>
<ID,20>
<SIG,>
<PR,1>
<ID,21>
<PARC,>
<LLAVA,>
<PR,9>
<PR,1>
<ID,22>
<FIN,>
<PR,12>
<PARA,>
<CAD,"di
cuantos dias
tiene el mes ">
<PARC,>
<FIN,>
<PR,12>
<PARA,>
<ID,20>
<PARC,>
<FIN,>
<PR,13>
<PARA,>
<ID,22>
<PARC,>
<FIN,>
<LLAVC,>
<PR,4>
<PARA,>
<ID,11>
<PARA,>
<ID,21>
<PARC,>
<PARC,>
<ID,22>
<ASIG,>
<ID,22>
<MOD,>
<cteent,1>
<FIN,>
<PR,7>
<ID,22>
<FIN,>
<LLAVC,>
<PR,8>
<PR,2>
<ID,13>
<PARA,>
<PR,1>
<ID,20>
<SIG,>
<PR,1>
<ID,21>
<SIG,>
<PR,1>
<ID,22>
<PARC,>
<LLAVA,>
<PR,7>
<ID,21>
<OPMAY,>
<cteent,1>
<OR,>
<ID,21>
<OPMAY,>
<cteent,12>
<OR,>
<ID,20>
<OPIG,>
<cteent,1>
<AND,>
<ID,20>
<OPMEN,>
<ID,12>
<PARA,>
<ID,21>
<SIG,>
<ID,22>
<PARC,>
<FIN,>
<LLAVC,>
<PR,8>
<ID,14>
<PARA,>
<PARC,>
<LLAVA,>
<PR,4>
<PARA,>
<ID,13>
<PARA,>
<cteent,25>
<SIG,>
<cteent,10>
<SIG,>
<cteent,2018>
<PARC,>
<PARC,>
<PR,12>
<PARA,>
<CAD,"OK">
<PARC,>
<FIN,>
<PR,4>
<PARA,>
<ctebool,1>
<PARC,>
<PR,12>
<PARA,>
<CAD,"OK">
<PARC,>
<FIN,>
<LLAVC,>
<PR,9>
<PR,3>
<ID,15>
<FIN,>
<ID,14>
<PARA,>
<PARC,>
<FIN,>

Árbol


Árbol resultado de:
Gramática: C:\Users\PABLO\Desktop\Arbol1.txt
Parse: C:\Users\PABLO\Proyectos\PDLG64\src\Salida\parse.txt


}
{
C (33)
B (8)
S (13)
return
X (31)
E (35)
R (38)
U (41)
V (45)
W (49)
Z (53)
G (59)
id
_G (65)
lambda
_Z (57)
lambda
_W (52)
lambda
_V (46)
>
W (49)
Z (53)
G (61)
count
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (36)
||
R (38)
U (41)
V (45)
W (49)
Z (53)
G (59)
id
_G (65)
lambda
_Z (57)
lambda
_W (52)
lambda
_V (46)
>
W (49)
Z (53)
G (61)
count
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (36)
||
R (38)
U (41)
V (45)
W (49)
Z (53)
G (59)
id
_G (65)
lambda
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (42)
==
V (45)
W (49)
Z (53)
G (61)
count
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (39)
&&
U (41)
V (45)
W (49)
Z (53)
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (37)
lambda
}
S (14)
print
(
E (35)
R (38)
U (41)
V (45)
W (49)
Z (53)
G (62)
CAD
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (37)
lambda
)
;
C (33)
B (5)
if
(
E (35)
R (38)
U (41)
V (45)
W (49)
Z (53)
G (63)
ctebool
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (37)
lambda
)
S (14)
print
(
E (35)
R (38)
U (41)
V (45)
W (49)
Z (53)
G (62)
CAD
_Z (57)
lambda
_W (52)
lambda
_V (48)
lambda
_U (44)
lambda
_R (40)
lambda
_E (37)
lambda
)
;
C (34)
lambda
}
P (1)
B (4)
var
T (11)
string
id
;
P (1)
B (8)
S (12)
id
_S (19)
(
L (24)
lambda
)
;
P (3)
lambda

Tabla de Símbolos

```
TABLA PRINCIPAL #1:
* LEXEMA: 'boolean'
        ATRIBUTOS:
        +despl: 0
        +tipo: 'bool'
        ------------------------------
* LEXEMA: 'bisiesto'
        ATRIBUTOS:
        +numParam: 1
        +TipoRetorno: 'bool'
        +EtiqFuncion: 'ETIbisiesto2'
        +TipoParam1: 'int'
        +ModoParam1: 1
        ------------------------------
* LEXEMA: 'dias'
        ATRIBUTOS:
        +numParam: 2
        +TipoRetorno: 'int'
        +EtiqFuncion: 'ETIdias3'
        +TipoParam1: 'int'
        +ModoParam1: 1
        +TipoParam2: 'int'
        +ModoParam2: 1
        ------------------------------
* LEXEMA: 'esFechaCorrecta'
        ATRIBUTOS:
        +numParam: 3
        +TipoRetorno: 'bool'
        +EtiqFuncion:
'ETIesFechaCorrecta4'
        +TipoParam1: 'int'
        +ModoParam1: 1
        +TipoParam2: 'int'
        +ModoParam2: 1
        +TipoParam3: 'int'
        +ModoParam3: 1
        ------------------------------
* LEXEMA: 'demo'
        ATRIBUTOS:
        +numParam: 0
        +TipoRetorno: 'tipo_vacio'
        +EtiqFuncion: 'ETIdemo5'
        ------------------------------
* LEXEMA: 'A_A_A_'
        ATRIBUTOS:
        +despl: 6
        +tipo: 'string'
        ------------------------------

TSL de la funcion 'bisiesto' #2:
* LEXEMA: 'a' (parametro de
entrada de la funcion)
        ATRIBUTOS:
        +despl: 0
        +tipo: 'int'
        ------------------------------

TSL de la funcion 'dias' #3:
* LEXEMA: 'm' (parametro de
entrada de la funcion)
        ATRIBUTOS:
        +despl: 0
        +tipo: 'int'
        ------------------------------
* LEXEMA: 'a' (parametro de
entrada de la funcion)
        ATRIBUTOS:
        +despl: 2
        +tipo: 'int'
        ------------------------------
* LEXEMA: 'dd'
        ATRIBUTOS:
```

+despl: 4
+tipo: 'int'

TSL de la funcion 'esFechaCorrecta' #4:
* LEXEMA: 'd' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'
---
* LEXEMA: 'm' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 2
+tipo: 'int'
---
* LEXEMA: 'a' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 4
+tipo: 'int'

TSL de la funcion 'demo' #5:

Prueba 5:

```
var bool boolean;
boolean = true;
var string pepe;
function bool bisiesto (int a)
{
        return (a % 4 > 0 && a - 122 != 0 || a * 400 < 0);
}
 function int dias (int m, int a)
{
        var int dd;
        print ( 'di cuantos dias tiene el mes ' );
        print (m);
        prompt(pepe);
        if (bisiesto(a)) dd = dd / 1;
        return dd;
}
 function bool esFechaCorrecta (int d, int m, int a)
{
        return !(d > dias (m, a));
}
 function demo ()
{

        if (esFechaCorrecta(25, 10, 2018)) print ( 'OK' );
}
 var int  aaa111 ;
demo();
```

Tokens

<PR,9>
<PR,2>
<ID,10>
<FIN,>
<ID,10>
<ASIG,>
<ctebool,1>
<FIN,>
<PR,9>
<PR,3>
<ID,11>
<FIN,>
<PR,8>
<PR,2>
<ID,12>
<PARA,>
<PR,1>
<ID,20>
<PARC,>
<LLAVA,>
<PR,7>
<PARA,>
<ID,20>
<MOD,>
<cteent,4>
<OPMAY,>
<cteent,0>
<AND,>
<ID,20>
<MENOS,>
<cteent,122>
<OPDISTINTO,>
<cteent,0>
<OR,>
<ID,20>
<MUL,>
<cteent,400>
<OPMEN,>
<cteent,0>
<PARC,>
<FIN,>
<LLAVC,>

<PR,8>
<PR,1>
<ID,13>
<PARA,>
<PR,1>
<ID,20>
<SIG,>
<PR,1>
<ID,21>
<PARC,>
<LLAVA,>
<PR,9>
<PR,1>
<ID,22>
<FIN,>
<PR,12>
<PARA,>
<CAD,"di
cuantos dias
tiene el mes ">
<PARC,>
<FIN,>
<PR,12>
<PARA,>
<ID,20>
<PARC,>
<FIN,>
<PR,13>
<PARA,>
<ID,11>
<PARC,>
<FIN,>
<PR,4>
<PARA,>
<ID,12>
<PARA,>
<ID,21>
<PARC,>
<PARC,>
<ID,22>
<ASIG,>
<ID,22>

<DIV,>
<cteent,1>
<FIN,>
<PR,7>
<ID,22>
<FIN,>
<LLAVC,>
<PR,8>
<PR,2>
<ID,14>
<PARA,>
<PR,1>
<ID,20>
<SIG,>
<PR,1>
<ID,21>
<SIG,>
<PR,1>
<ID,22>
<PARC,>
<LLAVA,>
<PR,7>
<NOT,>
<PARA,>
<ID,20>
<OPMAY,>
<ID,13>
<PARA,>
<ID,21>
<SIG,>
<ID,22>
<PARC,>
<PARC,>
<FIN,>
<LLAVC,>
<PR,8>
<ID,15>
<PARA,>
<PARC,>
<LLAVA,>
<PR,4>
<PARA,>

<ID,14>
<PARA,>
<cteent,25>
<SIG,>
<cteent,10>
<SIG,>
<cteent,2018>
<PARC,>
<PARC,>
<PR,12>
<PARA,>
<CAD,"OK">
<PARC,>
<FIN,>
<LLAVC,>
<PR,9>
<PR,1>
<ID,16>
<FIN,>
<ID,15>
<PARA,>
<PARC,>
<FIN,>

Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'boolean'
ATRIBUTOS:
+despl: 0
+tipo: 'bool'
---
* LEXEMA: 'pepe'
ATRIBUTOS:
+despl: 4
+tipo: 'string'
---
* LEXEMA: 'bisiesto'
ATRIBUTOS:
+numParam: 1
+TipoRetorno: 'bool'
+EtiqFuncion: 'ETIbisiesto2'
+TipoParam1: 'int'
+ModoParam1: 1
---
* LEXEMA: 'dias'
ATRIBUTOS:
+numParam: 2
+TipoRetorno: 'int'
+EtiqFuncion: 'ETIdias3'
+TipoParam1: 'int'
+ModoParam1: 1
+TipoParam2: 'int'
+ModoParam2: 1
---
* LEXEMA: 'esFechaCorrecta'
ATRIBUTOS:
+numParam: 3
+TipoRetorno: 'bool'
+EtiqFuncion: 'ETIesFechaCorrecta4'
+TipoParam1: 'int'
+ModoParam1: 1
+TipoParam2: 'int'
+ModoParam2: 1
+TipoParam3: 'int'
+ModoParam3: 1
---
* LEXEMA: 'demo'
ATRIBUTOS:
+numParam: 0
+TipoRetorno: 'tipo_vacio'
+EtiqFuncion: 'ETIdemo5'
---
* LEXEMA: 'aaa111'
ATRIBUTOS:
+despl: 134
+tipo: 'int'

TSL de la funcion 'bisiesto' #2:
* LEXEMA: 'a' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'

TSL de la funcion 'dias' #3:
* LEXEMA: 'm' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'
---
* LEXEMA: 'a' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 2
+tipo: 'int'
---
* LEXEMA: 'dd'
ATRIBUTOS:
+despl: 4
+tipo: 'int'

TSL de la funcion 'esFechaCorrecta' #4:
* LEXEMA: 'd' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'
---
* LEXEMA: 'm' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 2
+tipo: 'int'
---
* LEXEMA: 'a' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 4
+tipo: 'int'

TSL de la funcion 'demo' #5:

Pruebas Incorrectas:

Se muestra la lista de tokens y la tabla de símbolos hasta que se encuentra un error.

Prueba 6:

```
var bool b;
function bool bisiesto (int a)
{
	return (1 % 4 > 0 && 1 - 1022 != 0 || 1 * 400 < 0);
}
c=2+2+2+2+2+2+2+2+2+2+2+2+2+2+2;
function int dias (int m)
{
	switch (m)
	{
		case 1: case 3: case 5: case 7: case 8: case 10: case 12:
		return 3;
			break;
		case 4: case 6: while(true){a=2 ;} case 9: case 11:
			break; return 2;
		case 2: if (bisiesto (a)) a=1;
		default: a=2;
	}
	return 2;
	b = bisiesto(x);


}
return b;
```

Mensaje error

Error de compilacion:
Exception('ERROR Semántico en linea 24: sentencia return fuera de funcion')

Tokens

<PR,9>
<PR,2>
<ID,10>
<FIN,>
<PR,8>
<PR,2>
<ID,11>
<PARA,>
<PR,1>
<ID,20>
<PARC,>
<LLAVA,>
<PR,7>
<PARA,>
<cteent,1>
<MOD,>
<cteent,4>
<OPMAY,>
<cteent,0>
<AND,>
<cteent,1>
<MENOS,>
<cteent,1022>
<OPDISTINTO,>
<cteent,0>
<OR,>
<cteent,1>
<MUL,>
<cteent,400>
<OPMEN,>
<cteent,0>
<PARC,>
<FIN,>
<LLAVC,>
<ID,12>
<ASIG,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<FIN,>
<PR,8>
<PR,1>
<ID,13>
<PARA,>
<PR,1>
<ID,20>
<PARC,>
<LLAVA,>
<PR,10>
<PARA,>
<ID,20>
<PARC,>
<LLAVA,>
<PR,11>
<cteent,1>
<DOSPUNTOS,>
<PR,11>
<cteent,3>
<DOSPUNTOS,>
<PR,11>
<cteent,5>
<DOSPUNTOS,>
<PR,11>
<cteent,7>
<DOSPUNTOS,>
<PR,11>
<cteent,8>
<DOSPUNTOS,>
<PR,11>
<cteent,10>
<DOSPUNTOS,>
<PR,11>
<cteent,12>
<DOSPUNTOS,>
<PR,7>
<cteent,3>
<FIN,>
<PR,6>
<FIN,>
<PR,11>
<cteent,4>
<DOSPUNTOS,>
<PR,11>
<cteent,6>
<DOSPUNTOS,>
<PR,14>
<PARA,>
<ctebool,1>
<PARC,>
<LLAVA,>
<ID,14>
<ASIG,>
<cteent,2>
<FIN,>
<LLAVC,>
<PR,11>
<cteent,9>
<DOSPUNTOS,>
<PR,11>
<cteent,11>
<DOSPUNTOS,>
<PR,6>
<FIN,>
<PR,7>
<cteent,2>
<FIN,>
<PR,11>
<cteent,2>
<DOSPUNTOS,>
<PR,4>
<PARA,>
<ID,11>
<PARA,>
<ID,14>
<PARC,>
<PARC,>
<ID,14>
<ASIG,>
<cteent,1>
<FIN,>
<PR,5>
<DOSPUNTOS,>
<ID,14>
<ASIG,>
<cteent,2>
<FIN,>
<LLAVC,>
<PR,7>
<cteent,2>
<FIN,>
<ID,10>
<ASIG,>
<ID,11>
<PARA,>
<ID,15>
<PARC,>
<FIN,>
<LLAVC,>
<PR,7>
<ID,10>

Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'b'
ATRIBUTOS:
+despl: 0
+tipo: 'bool'
---
* LEXEMA: 'bisiesto'
ATRIBUTOS:
+numParam: 1
+TipoRetorno: 'bool'
+EtiqFuncion: 'ETIbisiesto2'
+TipoParam1: 'int'
+ModoParam1: 1
---
* LEXEMA: 'c'
ATRIBUTOS:
+despl: 4
+tipo: 'int'
---
* LEXEMA: 'dias'
ATRIBUTOS:
+numParam: 1
+TipoRetorno: 'int'
+EtiqFuncion: 'ETIdias3'
+TipoParam1: 'int'
+ModoParam1: 1
---
* LEXEMA: 'a'
ATRIBUTOS:
+despl: 6
+tipo: 'int'
---
* LEXEMA: 'x'
ATRIBUTOS:
+despl: 8
+tipo: 'int'

TSL de la funcion 'bisiesto' #2:
* LEXEMA: 'a' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'

TSL de la funcion 'dias' #3:
* LEXEMA: 'm' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'

Prueba 7:

```
var bool b;
c=2;
function bool bisiesto (bool a)
{
        var bool c;
        c=true;
        return (1 % 4 > 0 && 1 - 1022 != 0 || 1 * 400 < 0);
}

c=1;

b=bisiesto(bisiesto(bisiesto(true)));
c=2+2+2+2+2+2+2+2+2+2+2+2+2+2+2+2;

c=999999999999999999999999;
```

Mensaje error

Error de compilacion:
Exception('ERROR Lexico en linea 16: Entero supera el tamaño máximo')

Tokens

<PR,9>
<PR,2>
<ID,10>
<FIN,>
<ID,11>
<ASIG,>
<cteent,2>
<FIN,>
<PR,8>
<PR,2>
<ID,12>
<PARA,>
<PR,2>
<ID,20>
<PARC,>
<LLAVA,>
<PR,9>
<PR,2>
<ID,21>
<FIN,>
<ID,21>
<ASIG,>
<ctebool,1>
<FIN,>
<PR,7>
<PARA,>
<cteent,1>
<MOD,>
<cteent,4>
<OPMAY,>
<cteent,0>
<AND,>
<cteent,1>
<MENOS,>
<cteent,1022>
<OPDISTINTO,>
<cteent,0>
<OR,>
<cteent,1>
<MUL,>
<cteent,400>
<OPMEN,>
<cteent,0>
<PARC,>
<FIN,>
<LLAVC,>
<ID,11>
<ASIG,>
<cteent,1>
<FIN,>
<ID,10>
<ASIG,>
<ID,12>
<PARA,>
<ID,12>
<PARA,>
<ID,12>
<PARA,>
<ctebool,1>
<PARC,>
<PARC,>
<PARC,>
<FIN,>
<ID,11>
<ASIG,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<MAS,>
<cteent,2>
<FIN,>
<ID,11>
<ASIG,>

Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'b'
        ATRIBUTOS:
        +despl: 0
        +tipo: 'bool'
        ----
* LEXEMA: 'c'
        ATRIBUTOS:
        +despl: 4
        +tipo: 'int'
        ----
* LEXEMA: 'bisiesto'
        ATRIBUTOS:
        +numParam: 1
        +TipoRetorno: 'bool'
        +EtiqFuncion: 'ETIbisiesto2'
        +TipoParam1: 'bool'
        +ModoParam1: 1

TSL de la funcion 'bisiesto' #2:
* LEXEMA: 'a' (parametro de entrada de la funcion)
        ATRIBUTOS:
        +despl: 0
        +tipo: 'bool'
        ----
* LEXEMA: 'c'
        ATRIBUTOS:
        +despl: 4
        +tipo: 'bool'

Prueba 8:

```
var int a; var bool b;
b = false && 2 == 2%2 || true && a > 34 + 3*a;
var bool c;
c = a  >  a;
if (c) a  -=  1;
a=2+(2+c);
if (2  <  a) a  -=  4;
 a = a  +  4;
print (a);
print (b);
```

Mensaje error

Error de compilacion:
Exception('ERROR semantico en linea 6: expresion incorrecta')

Tokens

<PR,9>
<PR,1>
<ID,10>
<FIN,>
<PR,9>
<PR,2>
<ID,11>
<FIN,>
<ID,11>
<ASIG,>
<ctebool,0>
<AND,>
<cteent,2>
<OPIG,>
<cteent,2>
<MOD,>
<cteent,2>
<OR,>
<ctebool,1>
<AND,>
<ID,10>
<OPMAY,>
<cteent,34>
<MAS,>
<cteent,3>
<MUL,>
<ID,10>
<FIN,>
<PR,9>
<PR,2>
<ID,12>
<FIN,>
<ID,12>
<ASIG,>
<ID,10>
<OPMAY,>
<ID,10>
<FIN,>
<PR,4>
<PARA,>
<ID,12>
<PARC,>
<ID,10>
<ASIGR,>
<cteent,1>
<FIN,>
<ID,10>
<ASIG,>
<cteent,2>
<MAS,>
<PARA,>
<cteent,2>
<MAS,>
<ID,12>
<PARC,>

Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'a'
        ATRIBUTOS:
        +despl: 0
        +tipo: 'int'
        ----
* LEXEMA: 'b'
        ATRIBUTOS:
        +despl: 2
        +tipo: 'bool'
        ----
* LEXEMA: 'c'
        ATRIBUTOS:
        +despl: 6
        +tipo: 'bool'

Prueba 9:

```
var bool boolean;
 function bool bisiesto (int a)
{
        return (a + 4 > 0 && a + 100 != 0 && a % 400 != 0);
}
 function int dias (int m, int a)
{
        var int dd;
        print ( 'di cuantos dias tiene el mes ' );
        print (m);
        prompt(dd);
        if (bisiesto(a,boolean)) dd = dd % 1;
        return dd;
}
 function bool esFechaCorrecta (int d, int m, int a)
{
        return m > 1 || m > 12 || d == 1 && d < dias (m, a);
}
 function demo ()
{

        if (esFechaCorrecta(25, 10, 2018)) print ( 'OK' );
        if (true) print ( 'OK' );
}
 var string A_A_A_ ;
demo();
```

Mensaje error

Error de compilacion:
Exception("ERROR Semántico en linea 12: llamada a función que pide ['int'] con parámetro incorrectos['int', 'bool']")

Tokens

<PR,9>
<PR,2>
<ID,10>
<FIN,>
<PR,8>
<PR,2>
<ID,11>
<PARA,>
<PR,1>
<ID,20>
<PARC,>
<LLAVA,>
<PR,7>
<PARA,>
<ID,20>
<MAS,>
<cteent,4>
<OPMAY,>
<cteent,0>
<AND,>
<ID,20>
<MAS,>
<cteent,100>
<OPDISTINTO,>
<cteent,0>
<AND,>
<ID,20>
<MOD,>
<cteent,400>
<OPDISTINTO,>
<cteent,0>
<PARC,>
<FIN,>
<LLAVC,>
<PR,8>
<PR,1>
<ID,12>
<PARA,>
<PR,1>
<ID,20>
<SIG,>
<PR,1>
<ID,21>
<PARC,>
<LLAVA,>
<PR,9>
<PR,1>
<ID,22>
<FIN,>
<PR,12>
<PARA,>
<CAD,"di cuantos dias tiene el mes ">
<PARC,>
<FIN,>
<PR,12>
<PARA,>
<ID,20>
<PARC,>
<FIN,>
<PR,13>
<PARA,>
<ID,22>
<PARC,>
<FIN,>
<PR,4>
<PARA,>
<ID,11>
<PARA,>
<ID,21>
<SIG,>
<ID,10>
<PARC,>
<PARC,>

Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'boolean'
ATRIBUTOS:
+despl: 0
+tipo: 'bool'
---
* LEXEMA: 'bisiesto'
ATRIBUTOS:
+numParam: 1
+TipoRetorno: 'bool'
+EtiqFuncion: 'ETIbisiesto2'
+TipoParam1: 'int'
+ModoParam1: 1
---
* LEXEMA: 'dias'
ATRIBUTOS:
+numParam: 2
+TipoRetorno: 'int'
+EtiqFuncion: 'ETIdias3'
+TipoParam1: 'int'
+ModoParam1: 1
+TipoParam2: 'int'
+ModoParam2: 1

TSL de la funcion 'bisiesto' #2:
* LEXEMA: 'a' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'

TSL de la funcion 'dias' #3:
* LEXEMA: 'm' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'
---
* LEXEMA: 'a' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 2
+tipo: 'int'
---
* LEXEMA: 'dd'
ATRIBUTOS:
+despl: 4
+tipo: 'int'

Prueba 10:

```
var int contador;
var bool esCierto;
a=2;
var string cadena;

function int divide (int num1 ,string num2, int num3)
{
 a = b; //variable no existente que se declarará como global y entera
 var int b; //declaracion de variable local
 b=a; //b coge el valor de la variable global
 var int a; //declaracion de variable local de mismo nombre de la global
que hace que esta ultima no sea ya accesible

 return a;
ab=a3;
a3+1;
divide(1,'a',3);
}
```

Mensaje error

Error de compilacion:
Exception('ERROR Sintactico en linea 15: sintaxis incorrecta')

Tokens

<PR,9>
<PR,1>
<ID,10>
<FIN,>
<PR,9>
<PR,2>
<ID,11>
<FIN,>
<ID,12>
<ASIG,>
<cteent,2>
<FIN,>
<PR,9>
<PR,3>
<ID,13>
<FIN,>
<PR,8>
<PR,1>
<ID,14>
<PARA,>
<PR,1>
<ID,20>
<SIG,>
<PR,3>
<ID,21>
<SIG,>
<PR,1>
<ID,22>
<PARC,>
<LLAVA,>
<ID,12>
<ASIG,>
<ID,15>
<FIN,>
<PR,9>
<PR,1>
<ID,23>
<FIN,>
<ID,23>
<ASIG,>
<ID,12>
<FIN,>
<PR,9>
<PR,1>
<ID,24>
<FIN,>
<PR,7>
<ID,24>
<FIN,>
<ID,16>
<ASIG,>
<ID,17>
<FIN,>
<ID,17>
<MAS,>

Tabla de Símbolos

TABLA PRINCIPAL #1:
* LEXEMA: 'contador'
ATRIBUTOS:
+despl: 0
+tipo: 'int'
---
* LEXEMA: 'esCierto'
ATRIBUTOS:
+despl: 2
+tipo: 'bool'
---
* LEXEMA: 'a'
ATRIBUTOS:
+despl: 6
+tipo: 'int'
---
* LEXEMA: 'cadena'
ATRIBUTOS:
+despl: 8
+tipo: 'string'
---
* LEXEMA: 'divide'
ATRIBUTOS:
+numParam: 3
+TipoRetorno: 'int'
+EtiqFuncion: 'ETIdivide2'
+TipoParam1: 'int'
+ModoParam1: 1
+TipoParam2: 'string'
+ModoParam2: 1
+TipoParam3: 'int'
+ModoParam3: 1
---
* LEXEMA: 'b'
ATRIBUTOS:
+despl: 136
+tipo: 'int'
---
* LEXEMA: 'ab'
ATRIBUTOS:
+despl: 138
+tipo: 'int'

---
* LEXEMA: 'a3'
ATRIBUTOS:
+despl: 140
+tipo: 'int'

TSL de la funcion 'divide' #2:
* LEXEMA: 'num1' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 0
+tipo: 'int'
---
* LEXEMA: 'num2' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 2
+tipo: 'string'
---
* LEXEMA: 'num3' (parametro de entrada de la funcion)
ATRIBUTOS:
+despl: 130
+tipo: 'int'
---
* LEXEMA: 'b'
ATRIBUTOS:
+despl: 132
+tipo: 'int'
---
* LEXEMA: 'a'
ATRIBUTOS:
+despl: 134
+tipo: 'int'
