Memoria de la Práctica de Procesadores de Lenguajes: Analizador Léxico

Diego José Abengózar, Alejandro García, Ignacio Javier Encinas

23/10/2019

1 Diseño Analizador Léxico

-*Tokens*
-Gramática
-Autómata
-Acciones semánticas
-Errores

2 Tabla de Símbolos

3 Matriz de Transiciones

4 Anexo de Errores

Analizador Léxico

Tokens

$< ENT, num >$

$< CAD, lex >$

$< ID, lex >$

$< TipoVarLOG, - >$

$< ELSE, - >$

$< DECFunc, - >$

$< IF, - >$

$< input, - >$

$< tipoVarENT, - >$

$< Print, - >$

$< Return, - >$

$< tipoVarCAD, - >$

$< DEC, - >$

$< ASIG, - >$

$< ASIGOR, - >$

$< Coma, - >$

$< PuntoComa, - >$

$< ParentesisAbrir, - >$

$< ParentesisCerrar, - >$

$< CorcheteAbrir, - >$

$< CorcheteCerrar, - >$

$< SUMA, - >$

$< NOT, - >$

$< MENOR, - >$

Gramática

A→ $delA$ | ; | { | } | (|) | + | < | ! | = | , | $|B$ | lC | dD | $'E$ | $/F$

$B \rightarrow =$

$C \rightarrow lC|dC|\_C|\lambda$

D→ $dD|\lambda$

E→ $cE|*E|/E|'$

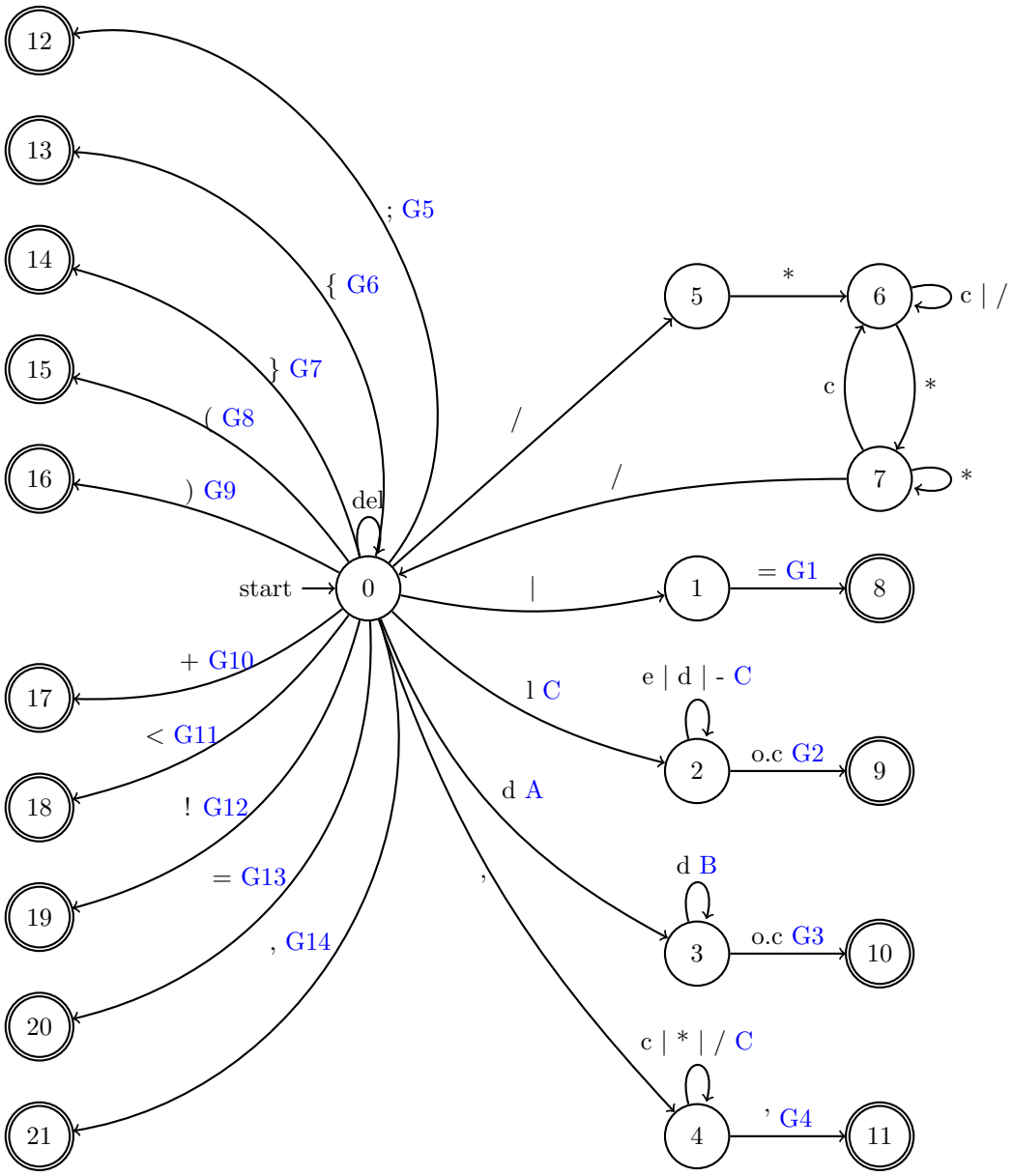
$F \rightarrow *G$

$G \rightarrow cG|/G|*H$

$H \rightarrow /A|cG|*H$

Donde c=T-{+, /}

Autómata Finito



Acciones Semánticas

Lee $\forall$ transición menos o.c

C: CONCAT

$G_1 : GEN\_TOKEN(ASIG\_OR,)$

$G_2 : if(lex = palRes)$
GEN_TOKEN(palRes, -)
else if ((p=BUSCA_TS(lex))=NULL)
p=AÑADE_TS(lex)
GEN_TOKEN(ID, p)

$G_3 : if((p = BUSCA\_TS(lex)) = NULL)$
p=AÑADE_TS(lex)
GEN_TOKEN(ID, p)

A : num = valor(d)

B: num = num$*$10 + valor(d)

$G_4 : if(num >= 2^{16})$
Error()
else GEN_TOKEN(ENT, num)

G_5 : GEN_TOKEN(CAD, lex)

G_6 : GEN_TOKEN(TipoVarLOG, -)

G_6 : GEN_TOKEN(ELSE, -)

G_7 : GEN_TOKEN(DECFunc, -)

G_8 : GEN_TOKEN(IF, -)

G_9 : GEN_TOKEN(input, -)

G_{10} : GEN_TOKEN(tipoVarENT, -)

G_{11} : GEN_TOKEN(Print, -)

G_{12} : GEN_TOKEN(Return, -)

G_{13} : GEN_TOKEN(tipoVarCAD, -)

G_{14} : GEN_TOKEN(DEC, -)

G_{15} : GEN_TOKEN(ASIG, -)

G_{16} : GEN_TOKEN(ASIGOR, -)

G_{17} : GEN_TOKEN(Coma, -)

G_{18} : GEN_TOKEN(PuntoComa, -)

G_{19} : GEN_TOKEN(ParentesisAbrir, -)

G_{20} : GEN_TOKEN(ParentesisCerrar, -)

G_{21} : GEN_TOKEN(CorcheteAbrir, -)

G_{22} : GEN_TOKEN(CorcheteCerrar, -)

G_{23} : GEN_TOKEN(SUMA, -)

G_{24} : GEN_TOKEN(NOT, -)

G_{25} : GEN_TOKEN(MENOR, -)

Errores

Cualquier transición que no sea contemplada por el autómata
Así como el empleo de números fuera de rango

Tabla de Símbolos

Contiene la información de los identificadores, de los cuales se guardan los campos: lexema, tipo y desplazamiento. Para las funciones se guardará el número de parámetros, su tipo, la forma de paso de parámetros y el tipo del valor de retorno. Se creará una tabla para su estudio local, la cual se destruirá al terminar de leer el cuerpo de la función.

Matriz de Transiciones

MT_AFD	|	**letra**	**digito**	’	/	_	**carácter**	*	**delimitador**	;
0	1 lee	2 C	3A	4 lee	5 lee	-1 error	-1 error	-1 error	0 lee	12 G5
1	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error
2	9 G2	2 C	2 C	9 G2	9 G2	2 C	9 G2	9 G2	9 G2	9 G2
3	10 G3	10 G3	3 B	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3
4	4 C	4 C	4 C	11 G4	4 C	4 C	4 C	4 C	4 C	4 C
5	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	6 lee	-1 error	-1 error
6	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	7 lee	6 lee	6 lee
7	6 lee	6 lee	6 lee	6 lee	0 lee	6 lee	6 lee	7 lee	6 lee	6 lee

MT_AFD	{	}	(	)	+	<	!	=	,
0	13 G6	14 G7	15 G8	16 G9	17 G10	18 G11	19 G12	20 G13	21 G14
1	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	8 G1	-1 error
2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2	9 G2
3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3	10 G3
4	4 C	4 C	4 C	4 C	4 C	4 C	4 C	4 C	4 C
5	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error	-1 error
6	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee
7	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee	6 lee

Anexo de Pruebas

Error 1: Número fuera de rango.
Fuente:
var int a;
var int b;
a = 33333;
b = a;
if ($a < b$) b = 1;
if ($b < a$) b = 8;
a = a + b;
print (a);
print (b);

Tokens
$< DEC, >< TipoVarENT, >< ID, 0 >< PuntoComa, >$
$< DEC, >< TipoVarENT, >< ID, 1 >< PuntoComa, >$
$< ID, 0 >< ASIG, >$

Error 2: Transición no prevista

Fuente:
var string texto; /*Comentario bueno*/
function imprime (string msg)
{
print (msg);
}
/ Comentario malo*/
function pideTexto ()
{.
print ('Introduce un texto');
input (texto);
}
pideTexto();
var string textoAux;
textoAux = texto;
imprime (textoAux);

Tokens:
$< DEC, >< TipoVarCAD, >< ID, 0 >< PuntoComa, >$
$< DECFunc, >< ID, 1 >< ParentesisAbrir, >< TipoVarCAD, >$
$< ID, 2 >< ParentesisCerrar, >< CorcheteAbrir, >< Print, >$
$< ParentesisAbrir, >< ID, 2 >< ParentesisCerrar, >< PuntoComa, >$
$< CorcheteCerrar, >$

Error 3: Transición no prevista

Fuente:

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

Tokens:

$< DEC, >< TipoVarCAD, >< ID, 0 >< PuntoComa, >$
$< DECFunc, >< ID, 1 >< ParentesisAbrir, >< ParentesisCerrar, >$
$< CorcheteAbrir, >< Print, >< ParentesisAbrir, >$
<CAD, "Introduce un texto);
input (texto);
}
function imprime (string msg)
{
print (" >
$< ID, 2 >$
$< ID, 3 >$

Pruebas pasadas con éxito:

1:

Fuente:

```
var int a;
var int b;
a = 3;
b = a;
var boolean c;
c = a < b;
if (c) b = 1;
c = b < a;
if (c) b = 4;
a = a + b;
print (a);
print (b);
```

Tokens:

$< DEC, - >< TipoVarENT, - >< ID, 0 >< PuntoComa, - >< DEC, - >$

$< TipoVarENT, - >< ID, 1 >< PuntoComa, - >< ID, 0 >$
$< ASIG, - >< ENT, 3 >< PuntoComa, - >< ID, 1 >$
$< ASIG, - >< ID, 0 >< PuntoComa, - >< DEC, - >$
$< TipoVarLOG, - >< ID, 2 >< PuntoComa, - >< ID, 2 >$
$< ASIG, - >< ID, 0 >< MENOR, - >< ID, 1 >$
$< PuntoComa, - >< IF, - >< ParentesisAbrir, - >< ID, 2 >$
$< ParentesisCerrar, - >< ID, 1 >< ASIG, - >< ENT, 1 >$
$< PuntoComa, - >< ID, 2 >< ASIG, - >< ID, 1 >$
$< MENOR, - >< ID, 0 >< PuntoComa, - >< IF, - >$
$< ParentesisAbrir, - >< ID, 2 >< ParentesisCerrar, - >< ID, 1 >$
$< ASIG, - >< ENT, 4 >< PuntoComa, - >< ID, 0 >$
$< ASIG, - >< ID, 0 >< SUMA, - >< ID, 1 >$
$< PuntoComa, - >< Print, - >< ParentesisAbrir, - >< ID, 0 >$
$< ParentesisCerrar, - >< PuntoComa, - >< Print, - >< ParentesisAbrir, - >$
$< ID, 1 >< ParentesisCerrar, - >< PuntoComa, - >$

2:

Fuente:

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

Tokens:

$< DEC, >< TipoVarENT, >< ID, 0 >< PuntoComa, >$
$< DEC, >< TipoVarENT, >< ID, 1 >< PuntoComa, >$
$< DEC, >< TipoVarENT, >< ID, 2 >< PuntoComa, >$
$< Print, >< ParentesisAbrir, >< CAD,$ "Introduce el primer operando" $>$
$< ParentesisCerrar, >$
$< PuntoComa, >< Input, >< ParentesisAbrir, >< ID, 0 >$
$< ParentesisCerrar, >< PuntoComa, >< Print, >< ParentesisAbrir, >$
$< CAD,$ "Introduce el segundo operando" $>< ParentesisCerrar, >< PuntoComa, >$
$< Input, >$
$< ParentesisAbrir, >< ID, 1 >< ParentesisCerrar, >< PuntoComa, >$
$< DECFunc, >< TipoVarENT, >< ID, 3 >< ParentesisAbrir, >< TipoVarENT, >$
$< ID, 4 >< Coma, >< TipoVarENT, >< ID, 5 >$
$< ParentesisCerrar, >< CorcheteAbrir, >< Return, >< ID, 4 >$

$< SUMA, >< ID, 5 >< PuntoComa, >< CorcheteCerrar, >$
$< ID, 2 >< ASIG, >< ID, 3 >< ParentesisAbrir, >$
$< ID, 0 >< Coma, >< ID, 1 >< ParentesisCerrar, >$
$< PuntoComa, >< Print, >< ParentesisAbrir, >< ID, 2 >$
$< ParentesisCerrar, >< PuntoComa, >$

3:

Fuente:

```
var int a;
var int b;
a = 3;
b = a;
if (a < b) b = 1;
a = a + b;
print (a);
print (b);
```

Tokens:

$< DEC, >< TipoVarENT, >< ID, 0 >< PuntoComa, >< DEC, >$
$< TipoVarENT, >< ID, 1 >< PuntoComa, >< ID, 0 >$
$< ASIG, >< ENT, 3 >< PuntoComa, >< ID, 1 >$
$< ASIG, >< ID, 0 >< PuntoComa, >< IF, >$
$< ParentesisAbrir, >< ID, 0 >< MENOR, >< ID, 1 >$
$< ParentesisCerrar, >< ID, 1 >< ASIG, >< ENT, 1 >$
$< PuntoComa, >< ID, 0 >< ASIG, >< ID, 0 >$
$< SUMA, >< ID, 1 >< PuntoComa, >< Print, >$
$< ParentesisAbrir, >< ID, 0 >< ParentesisCerrar, >< PuntoComa, >$
$< Print, >< ParentesisAbrir, >< ID, 1 >< ParentesisCerrar, >$
$< PuntoComa, >$