Procesador de Lenguaje JavaScript PL

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 (" ")**

Analizador Léxico

"Un analizador léxico o analizador lexicográfico es la primera fase de un compilador, consistente en un programa que recibe como entrada el código fuente de otro programa y produce una salida compuesta de tokens"

– Wikipedia

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 → $delS \mid lA \mid "C \mid \ dE \mid -G \mid \mid i \mid \&J \mid = N \mid !Q \mid /U \mid \ + \mid c.e \mid \_A$	J → &
A → $dA \mid lA \mid o.c \mid \_A$	N → = | $o.c$
C → $c_1 C \mid "$	Q → =
E → $dE \mid o.c$	U → $* Y$
G→ = | $o.c$	Y → $c_2 Y \mid * Z$
I → |	Z → $c_3 Y \mid * Z \mid /S$

Leyenda

c1 = cualquier carácter | {"}

c2= cualquier carácter | {}*

c3= cualquier carácter | {, /}*

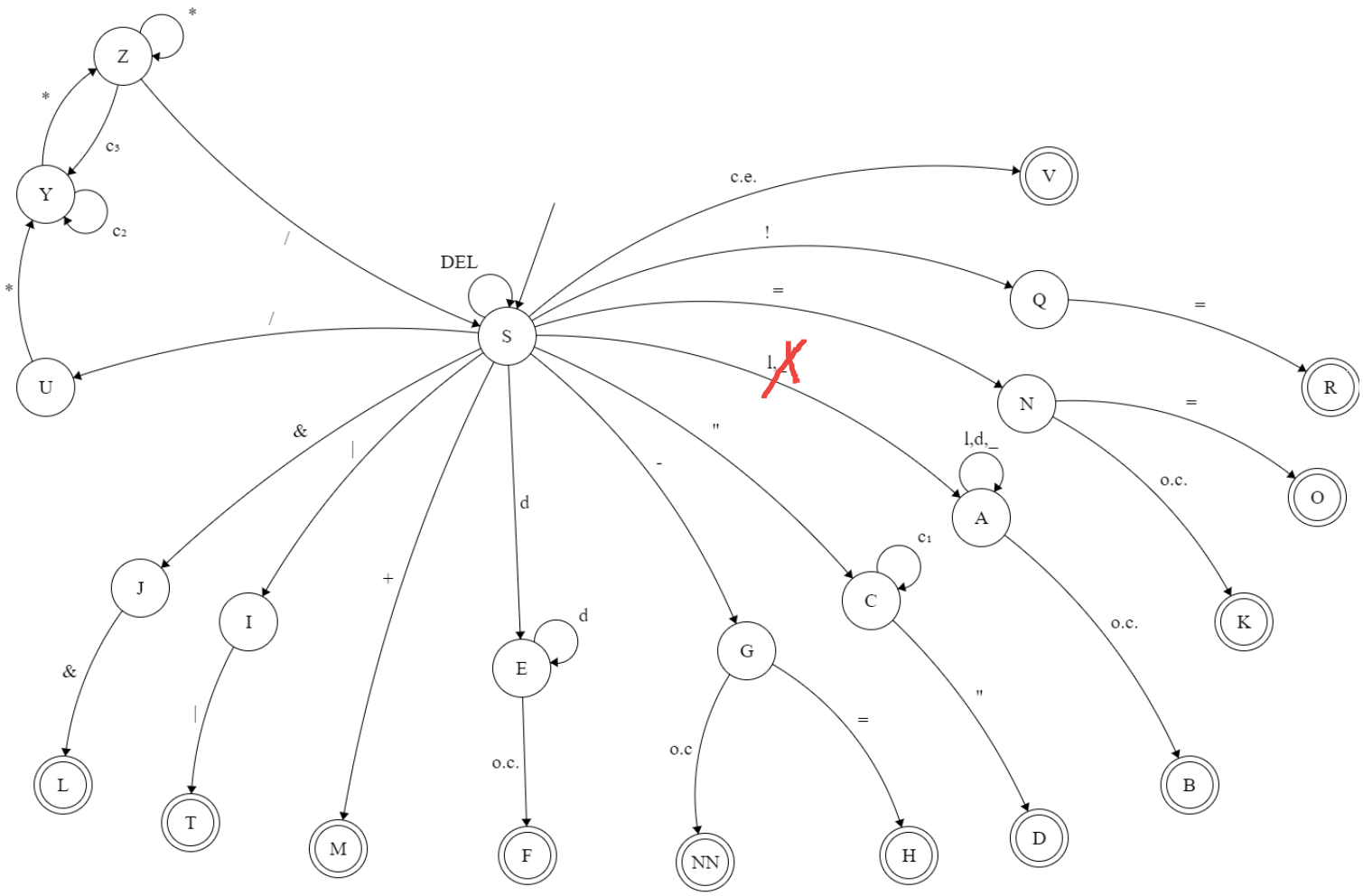
c.e = caracteres especiales: {,},(,), ; , ','

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(lexema.length >128){

error

}else{

if(isReservada){

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

Case-Sensitive JS

}else{

Gen_Token(<identificador,posTs>); // Siendo posTS =

GestorTablaSimbolos.insertar(lexema);

} // El gestor se encarga de los problemas

}

S→C: lexema= “”; leer();

C→C: lexema= lexema⊕ c_1; leer();

C→D:

if(lexema.length >128){ 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 > 2^16 -1){

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>)

Implementación en Java

¿Cómo ejecutarlo?

>Ejecutar main.java

Se ha incorporado un Pop-Up grafico para elegir el archivo a Analizar

Por el momento se hace un dump de todos los datos en el folder de outputs del programa, además al finalizar el análisis, el programa salta todas las tablas de símbolos creadas en formato visual, aunque se encuentra en el fichero de texto como así se preciso

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	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	Desplazamiento	NºParametros	returnType	eti

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

Ejemplo de Output Visual

A pesar de devolver un fichero de texto, según las especificaciones dadas por el profesorado. Se ha decido hacerlo también en un mini interfaz de usuario

Nota: El valor de “error” es porque nos apeteció, no para confundir a nadie

Errores

0	Carácter No Valido
1	Lexema ya en tabla de símbolos
-1	Error de Programador
2	Integer Out Of Bounds
3	String Out Of Bounds
4	Variable Name Out Of Bounds

Anexo

Prueba 1

```
let number n1;
let boolean l1;
let string cad;
let number n2;
let boolean l2;

alert ("PdL");
input (esto_es_un_nombre_de_variable_global_de_tipo_entero);
input (n1);
l1 = l2;
if (l1&& l2) cad = "hello";
n2 = n1 - 378;

alert(     33
           -
           n1
           -
           n2);
function boolean ff(boolean ss)
{
      l2 =  l1;
      if (l2) l1 = ff (ss);
      varglobal = 8888;
      return (ss);
}
if (ff(l1)) alert (varglobal);
```

Tokens Autogenerados

```
<let, >
<number, >
<identificador, 0>
<puntoYcoma, >
<let, >
<boolean, >
<identificador, 1>
<puntoYcoma, >
<let, >
<string, >
<identificador, 2>
<puntoYcoma, >
<let, >
<number, >
<identificador, 3>
<puntoYcoma, >
<let, >
<boolean, >
<identificador, 4>
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<cadena, "PdL">
<cerrarParentesis, >
<puntoYcoma, >
<input, >
<abrirParentesis, >
<identificador, 5>
```

<cerrarParentesis, >
<puntoYcoma, >
<input, >
<abrirParentesis, >
<identificador, 0>
<cerrarParentesis, >
<puntoYcoma, >
<identificador, 1>
<asignacion, >
<identificador, 4>
<puntoYcoma, >
<if, >
<abrirParentesis, >
<identificador, 1>
<opLogico, 1>
<identificador, 4>
<cerrarParentesis, >
<identificador, 2>
<asignacion, >
<cadena, "hello">
<puntoYcoma, >
<identificador, 3>
<asignacion, >
<identificador, 0>
<opAritmetico, 2>
<cteEntera, 378>
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<cteEntera, 33>
<opAritmetico, 2>
<identificador, 0>
<opAritmetico, 2>
<identificador, 3>
<cerrarParentesis, >
<puntoYcoma, >
<function, >
<boolean, >
<identificador, 6>
<abrirParentesis, >
<boolean, >
<identificador, 7>
<cerrarParentesis, >
<abrirCorchete, >
<identificador, 4>
<asignacion, >
<identificador, 1>
<puntoYcoma, >
<if, >
<abrirParentesis, >
<identificador, 4>
<cerrarParentesis, >
<identificador, 1>
<asignacion, >
<identificador, 6>
<abrirParentesis, >
<identificador, 7>
<cerrarParentesis, >

```
<puntoYcoma, >
<identificador, 8>
<asignacion, >
<cteEntera, 8888>
<puntoYcoma, >
<return, >
<abrirParentesis, >
<identificador, 7>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<if, >
<abrirParentesis, >
<identificador, 6>
<abrirParentesis, >
<identificador, 1>
<cerrarParentesis, >
<cerrarParentesis, >
<alert, >
<abrirParentesis, >
<identificador, 8>
<cerrarParentesis, >
<puntoYcoma, >
<EOF, >
```

Tabla Simbolos

```
CONTENIDO DE LA TABLA TSMAIN #1 :

      * LEXEMA : 'n1'
------------------------------------------------------------
      * LEXEMA : 'l1'
------------------------------------------------------------
      * LEXEMA : 'cad'
------------------------------------------------------------
      * LEXEMA : 'n2'
------------------------------------------------------------
      * LEXEMA : 'l2'
------------------------------------------------------------
      * LEXEMA :
'esto_es_un_nombre_de_variable_global_de_tipo_entero'
------------------------------------------------------------
      * LEXEMA : 'ff'
------------------------------------------------------------
      * LEXEMA : 'ss'
------------------------------------------------------------
      * LEXEMA : 'varglobal'
------------------------------------------------------------
```

Prueba 2

```
let string texto;
function print (string msg)
{
      alert (msg);
}
function pideTexto ()
{
      alert ("Introduce un texto");
      input (texto);
}
```

```
pideTexto();
let string textoAux;
textoAux = texto;
print (textoAux);
```

Tokens

```
<let, >
<string, >
<identificador, 0>
<puntoYcoma, >
<function, >
<identificador, 1>
<abrirParentesis, >
<string, >
<identificador, 2>
<cerrarParentesis, >
<abrirCorchete, >
<alert, >
<abrirParentesis, >
<identificador, 2>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<function, >
<identificador, 3>
<abrirParentesis, >
<cerrarParentesis, >
<abrirCorchete, >
<alert, >
<abrirParentesis, >
<cadena, "Introduce un texto">
<cerrarParentesis, >
<puntoYcoma, >
<input, >
<abrirParentesis, >
<identificador, 0>
<cerrarParentesis, >
<puntoYcoma, >
<cerrarCorchete, >
<identificador, 3>
<abrirParentesis, >
<cerrarParentesis, >
<puntoYcoma, >
<let, >
<string, >
<identificador, 4>
<puntoYcoma, >
<identificador, 4>
<asignacion, >
<identificador, 0>
<puntoYcoma, >
<identificador, 1>
<abrirParentesis, >
<identificador, 4>
<cerrarParentesis, >
<puntoYcoma, >
<EOF, >
```

Tabla símbolos

```
CONTENIDO DE LA TABLA TSMAIN #1 :

      * LEXEMA : 'texto'
----------------------------------------------------------------
      * LEXEMA : 'print'
----------------------------------------------------------------
      * LEXEMA : 'msg'
----------------------------------------------------------------
      * LEXEMA : 'pideTexto'
----------------------------------------------------------------
      * LEXEMA : 'textoAux'
----------------------------------------------------------------
```

Prueba 3

```
let number x;
let number z;
let boolean b;
input (x);
alert (x);
input (z);
alert (x+z);
b=x!=z;if (b)
x =
  x + 6
    + z
    - 1
    - (2
    - y
    - 6);
```

Tokens

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

```
<cerrarParentesis, >
<puntoYcoma, >
<alert, >
<abrirParentesis, >
<identificador, 0>
<opAritmetico, 1>
<identificador, 1>
<cerrarParentesis, >
<puntoYcoma, >
<identificador, 2>
<asignacion, >
<identificador, 0>
<opRelacional, 2>
<identificador, 1>
<puntoYcoma, >
<if, >
<abrirParentesis, >
<identificador, 2>
<cerrarParentesis, >
<identificador, 0>
<asignacion, >
<identificador, 0>
<opAritmetico, 1>
<cteEntera, 6>
<opAritmetico, 1>
<identificador, 1>
<opAritmetico, 2>
<cteEntera, 1>
<opAritmetico, 2>
<abrirParentesis, >
<cteEntera, 2>
<opAritmetico, 2>
<identificador, 3>
<opAritmetico, 2>
<cteEntera, 6>
<cerrarParentesis, >
<puntoYcoma, >
<EOF, >
```

Tabla Simbolos

```
CONTENIDO DE LA TABLA TSMAIN #1 :

      * LEXEMA : 'x'
------------------------------------------------------------
      * LEXEMA : 'z'
------------------------------------------------------------
      * LEXEMA : 'b'
------------------------------------------------------------
      * LEXEMA : 'y'
------------------------------------------------------------
```

Prueba 4

```
/*
Error del tipo: Digito muy grande
*/
4654564659999999
6894351654
```

Tokens

```
<EOF, >
```

Tabla Simbolos

```
CONTENIDO DE LA TABLA TSMAIN #1 :
```

Errores

```
Integer Out Of Bounds at Line:4
Integer Out Of Bounds at Line:5
```

Prueba 5

```
/*
Error del tipo: String y variable muy grande
*/

"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

Tokens

```
<cadena,
"aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa">
<EOF, >
```

Tabla Simbolos

```
CONTENIDO DE LA TABLA TSMAIN #1 :
```

Errores

```
String Out Of Bounds -- Desconocemos si es un error que un String
sea mayor a 128.Pensamos que sí pero por mera diligencia at Line:11
Variable Name Out Of Bounds at Line:12
```

Prueba 6

```
/*
Error del tipo: Caracter no valido
*/
????
```

Tokens

```
<EOF, >
```

Tabla Simbolos

```
CONTENIDO DE LA TABLA TSMAIN #1 :
```

Errores

```
Error code 0: Caracter No valido at Line:17
Error code 0: Caracter No valido at Line:17
Error code 0: Caracter No valido at Line:17
Error code 0: Caracter No valido at Line:17
```

Observacion

Las pruebas 4,5 y 6 formaban parte de un mismo fichero que hemos separado, es por ello que las lineas de error estarán algo desplazadas

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