

PDL: Práctica Procesador

Procesador JavaScript-PDL

Serrano, Arrese Francisco Javier
Cañibano, Lopez Alberto
Vallejo, Collados Jesús

grupo? Creo que 14

Procesadores de Lenguajes
Universidad Politécnica de Madrid
Curso 2020-2021

PALABRAS RESERVADAS

alert
 boolean
 else
 function
 if
 input
 let
 number
 return
 string
 while
 false
 true
 do


TOKENS

alert	<alert, >
boolean	<boolean, >
else	<else, >
function	<function, >
if	<if, >
input	<input, >
let	<let, >
number	<number, >
return	<return, >
string	<string, >
while	<while, >
false	<false, >
true	<true, >
do	<do, >
autoInc	<Autoincremento (++), >
Número	<constante entera, >
Posición(Número)	<Cadena ('), >
Número	<Identificador, >
equal	<=, >
colon	<:, >
semicolon	<;, >
openPar	<(, >
closePar	<), >
openBra	<{, >
closeBra	<}, >
plus	<+, >
minus	<-, >
and	<&&, >
not	<!, >
notEquals	<!=, >
equals	<==, >

? =
 → valores
 → lexema
 → posición en TS
 Atributos del token

GRAMÁTICA

S:--> del S | dA | lB | +C | -D | /F | (|) | { | } | =G | !G | , | ; | &H | 'J
A:--> dA | lambda
B:--> lB | dB | _B | lambda
C:--> + | lambda
D:--> - | lambda
F:--> *E
E:--> cE | *I | /
I:--> *I | c'E
G:--> = | lambda
H:--> &
J:--> eJ | '

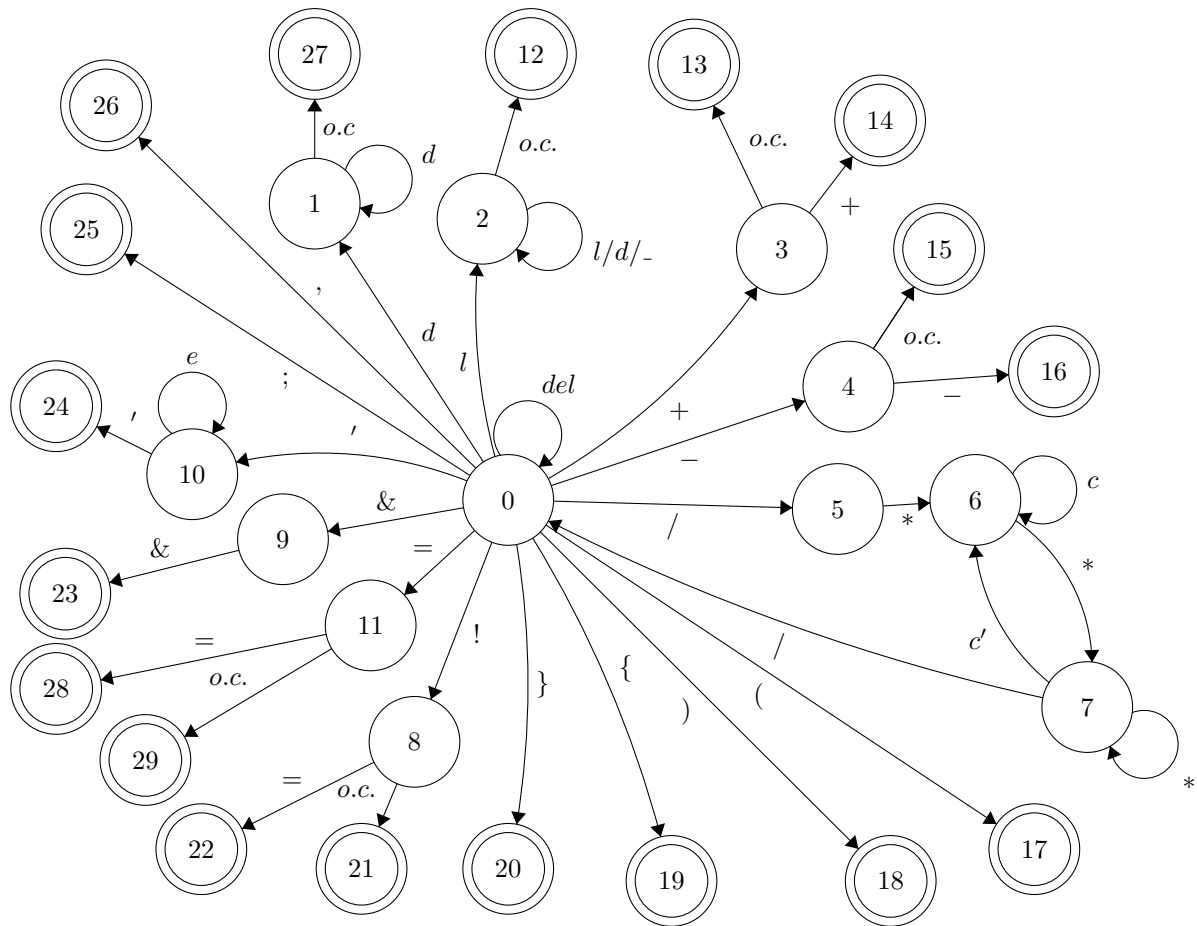


%-----Automata-----

d -- digito
l -- letra minuscula
del -- delimitador(blanco, tab, EOL)
c -- caracteres - (*)
c' -- caracteres - (* /)
e -- caracteres - (')

AUTÓMATA FINITO DETERMINISTA

d	Dígito
l	Letra
del	Delimitador
c	Caracteres -{*}
c'	Caracteres -{*/*}
e	Caracteres -{'}
o.c.	Otro Carácter

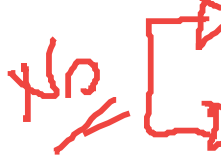


ACCIONES SEMÁNTICAS

Leer: Se lee en todos los estados menos en los que pone o.c.


Errores: Cualquier transición no declarada dara error.

Caso 0-1:



```
if siguienteCaracter==d
    numero=valor(d)
else
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 1-1:



```
if siguienteCaracter==d
    numero=numero+d
else
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 1-27:

GenerarToken(wholeConst,numero)

Caso 0-2:

```
if siguienteCaracter==l
    lexema=l
else
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 2-2:

```
if siguienteCaracter== l | d | '_'
    lexema=lexema+(l|d|'_')
else
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 2-12:

GenerarToken(ID,posicionTablaSimbolos)

Caso 0-3:

```
if siguienteCaracter=='+'
    //Nada
else
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 3-13:

GenerarToken(aritOp,plus)

Caso 3-14:

```
if siguienteCaracter=='+'
    GenerarToken(autoIncOp,autoinc)
else
    Error("SIMBOLO NO RECONOCIDO")
```

2º transitar de 0 a 1 es porque
has leído un dígito. Si se
ha leído otra cosa, no es
error sino que se hace
otra transición distinta

Si se lee otro, se
transita 1 a 2

Cambiado todo

Caso 0-4:

```
if siguienteCaracter=='-'  
    //Nada  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 4-15:

```
GenerarToken(aritOp,minus)
```

Caso 4-16:

```
if siguienteCarcter=='-'  
    GenerarToken(autoIncOp,autoinc)  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 0-5:

```
if siguienteCaracter=='/'  
    //NADA  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 5-6:

```
if siguienteCaracater=='*'  
    //NADA  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 6-6:

```
if siguienteCaracater=='c'  
    //NADA  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 6-7:

```
if siguienteCaracater=='*'  
    //NADA  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 7-6:

```
if siguienteCaracater=='c'  
    //NADA  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 7-0:

```
if siguienteCaracater=='/'  
    //NADA  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 0-26:

```
if siguienteCaracter==','  
    GenerarToken(separator,colon)  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 0-25:

```
if siguienteCaracter==';'  
    GenerarToken(separator,semicolon)  
else  
    Error("SIMBOLO NO RECONOCIDO") )
```

Caso 0-20:

```
if siguienteCaracter=='}'  
    GenerarToken(separator,closeBraq)  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 0-19:

```
if siguienteCaracter=='{'  
    GenerarToken(separator,openBraq)  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 0-18:

```
if siguienteCaracter==')'  
    GenerarToken(separator,closePar)  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 0-17:

```
if siguienteCaracter=='('  
    GenerarToken(separator,openPar)  
else  
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 0-10:

```
if siguienteCaracter==' '  
    lexema=' '  
else  
    Error ("SIMBOLO NO RECONOCIDO")
```

Caso 10-10:

```
if siguienteCaracter==e  
    lexema=lexema+e  
else  
    Error ("SIMBOLO NO RECONOCIDO")
```

Caso 10-24:

```
if siguienteCaracater==' '  
    GenerarToken(chain,posicionTablaSimbolos)//revisar  
else  
    Error ("SIMBOLO NO RECONOCIDO")
```

Caso 0-8:

```
if siguienteCaracter=='!'  
  
else  
    Error ("SIMBOLO NO RECONOCIDO")
```

Caso 8-21:

```
GenerarToken(logOp,not)
```

Caso 8-22:

```
if siguienteCaracater == '='  
    GenerarToken(relOp,notEquals)  
else  
    Error ("SIMBOLO NO RECONOCIDO")
```

Caso 0-11:

```
if siguienteCaracater == '='  
  
else  
    Error ("SIMBOLO NO RECONOCIDO")
```

Caso 11-28:

```
if siguienteCaracater == '='  
    GenerarToken(relOp,equal)  
else  
    Error ("SIMBOLO NO RECONOCIDO")
```

Caso 11-29:

```
GenerarToken(asigOp,equal)
```


Caso 0-9:

```
if siguienteCaracter == '&'

else
    Error("SIMBOLO NO RECONOCIDO")
```

Caso 9-23:

```
if siguienteCaracter == '&'
    GenerarToken(logOp,and)
else
    Error("SIMBOLO NO RECONOCIDO")
```

TABLA DE SIMBOLOS

El valor de los atributos y numero de tabla seran corregidos con el valor real mas adelante.

```
Contenido Tabla Símbolos # N :
* LEXEMA : 'x'
ATRIBUTOS :
+ tipo: unknown
+ despl: unknown
```

CASOS DE PRUEBA

Prueba 1: *CASO CORRECTO*

Código:

```
number a = 1;
string pp = 'hola';

/* hola

disculpa*/
if (a && a) {
    a = 2;
}
```

Tokens:

```
<number,>
<ID,a>
<asigOp,equal>
<wholeConst,1>
<separator,semicolon>
<string,>
<ID,pp>
<asigOp,equal>
<chain,'hola'>
```

no lexema sino posición en TS

TS:

Contenido Tabla Símbolos # 0 :

```
* LEXEMA : 'a'
  ATRIBUTOS :
    + tipo: unknown
    + despl: unknown
* LEXEMA : 'pp'
  ATRIBUTOS :
    + tipo: unknown
    + despl: unknown
```

Errores

Prueba 2: *CASO CORRECTO*

Código:

```
function padre(c) {

    let b = c;
    b++;

    c - b
```

```

    return c
}

```

Tokens:

```

<function,>
<ID,padre>
<separator,openPar>
<ID,c>
<separator,closePar>
<separator,openBraque>
<let,>
<ID,b>
<asigOp,equal>
<ID,c>
<separator,semicolon>
<ID,b>
<autoIncOp,autoinc>
<separator,semicolon>
<ID,c>
<aritOp,minus>
<ID,b>
<return,>
<ID,c>
<separator,closeBraque>

```

TS:

Contenido Tabla Símbolos # 0 :

```

* LEXEMA : 'padre'
  ATRIBUTOS :
    + tipo: unknown
    + Despl: -1
* LEXEMA : 'c'
  ATRIBUTOS :
    + tipo: unknown
    + Despl: -1
* LEXEMA : 'b'
  ATRIBUTOS :
    + tipo: unknown
    + Despl: -1

```

Errores

Prueba 3: *CASO CORRECTO*

Código:

```

boolean verdadero = true;
boolean grupo14 = true;
boolean aprobado = true;
do {

```

10

aprobado

```

verdadero = false;
if (true) {
    verdadero = true
}

} while (grupo14 = aprobado)

```

Tokens:

```

<boolean,>
<ID,verdadero>
<asigOp,equal>
<true,>
<separator,semicolon>
<boolean,>
<ID,grupo14>
<asigOp,equal>
<true,>
<separator,semicolon>
<boolean,>
<ID,aprovado>
<asigOp,equal>
<true,>
<separator,semicolon>
<do,>
<separator,openBraq>
<ID,verdadero>
<asigOp,equal>
<false,>
<separator,semicolon>
<if,>
<separator,openPar>
<true,>
<separator,closePar>
<separator,openBraq>
<ID,verdadero>
<asigOp,equal>
<true,>
<separator,closeBraq>
<separator,closeBraq>
<while,>
<separator,openPar>
<ID,grupo14>
<asigOp,equal>
<ID,aprovado>
<separator,closePar>

```

TS:

```

Contenido Tabla Símbolos # 0 :
* LEXEMA : 'verdadero'

```

```

    ATRIBUTOS :
        + tipo: unknown
        + Despl: -1
* LEXEMA : 'grupo14'
    ATRIBUTOS :
        + tipo: unknown
        + Despl: -1
* LEXEMA : 'aprovado'
    ATRIBUTOS :
        + tipo: unknown
        + Despl: -1

```

Errores:

Prueba 4: *CASO INCORRECTO*

Código:

```

    number 1a = 1
string pp = 'hola
numbe;
if (hola
    else {

```

&

Tokens:

```

<number,>
<wholeConst,1>
<ID,a>
<asigOp,equal>
<wholeConst,1>
<string,>
<ID,pp>
<asigOp,equal>
<ID,numbe>
<separator,semicolon>
<if,>
<separator,openPar>
<ID,hola>
<else,>
<separator,openBraq>

```

TS:

```

Contenido Tabla Símbolos # 0 :
* LEXEMA : 'a'

```

```

    ATRIBUTOS :
+ tipo: unknown
+ Despl: -1
* LEXEMA : 'pp'
    ATRIBUTOS :
+ tipo: unknown
+ Despl: -1
* LEXEMA : 'numbe'
    ATRIBUTOS :
+ tipo: unknown
+ Despl: -1
* LEXEMA : 'hola'
    ATRIBUTOS :
+ tipo: unknown
+ Despl: -1

```

Errores:

```

++ Error: ' cadena no se cierra en ningun momento,abierto en caracter: 13 ,linea: 2
++ Error: & esta solo en caracter: 1 ,linea: 8

```

Prueba 5: *CASO INCORRECTO*


Código:

```

1manolo == >

- -

```

'Esto esta escrito un sabado por la tarde 

```

/*Sin embargo
ha sido un poco tedioso

```

Tokens:

```

<wholeConst,1>
<ID,manolo>
<relOp,equals>
<aritOp,minus>
<aritOp,minus>

```

TS:

```

Contenido Tabla Símbolos # 0 :
* LEXEMA : 'manolo'
  ATRIBUTOS :
    + tipo: unknown
    + Despl: -1

```

Errores:

```
++ Error: Caracter no reconocido:[>] en caracter: 11 ,linea: 1
++ Error: ' cadena no se cierra en ningun momento,abierto en caracter: 5 ,linea: 6
```

Prueba 6: *CASO INCORRECTO*

Código:

```
&
!!
===
else
if
{{
    ((
```

→ También tienes
un comentario sin
Cerrar

Tokens:

```
<logOp,not>
<logOp,not>
<relOp,equals>
<asigOp,equal>
<else,>
<if,>
<separator,openBraque>
<separator,openBraque>
<separator,openPar>
<separator,openPar>
```

TS:

Errores:

```
++ Error: & esta solo en caracter: 1 ,linea: 1
```