PDL: Práctica Procesador

Procesador JavaScript-PDL

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

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

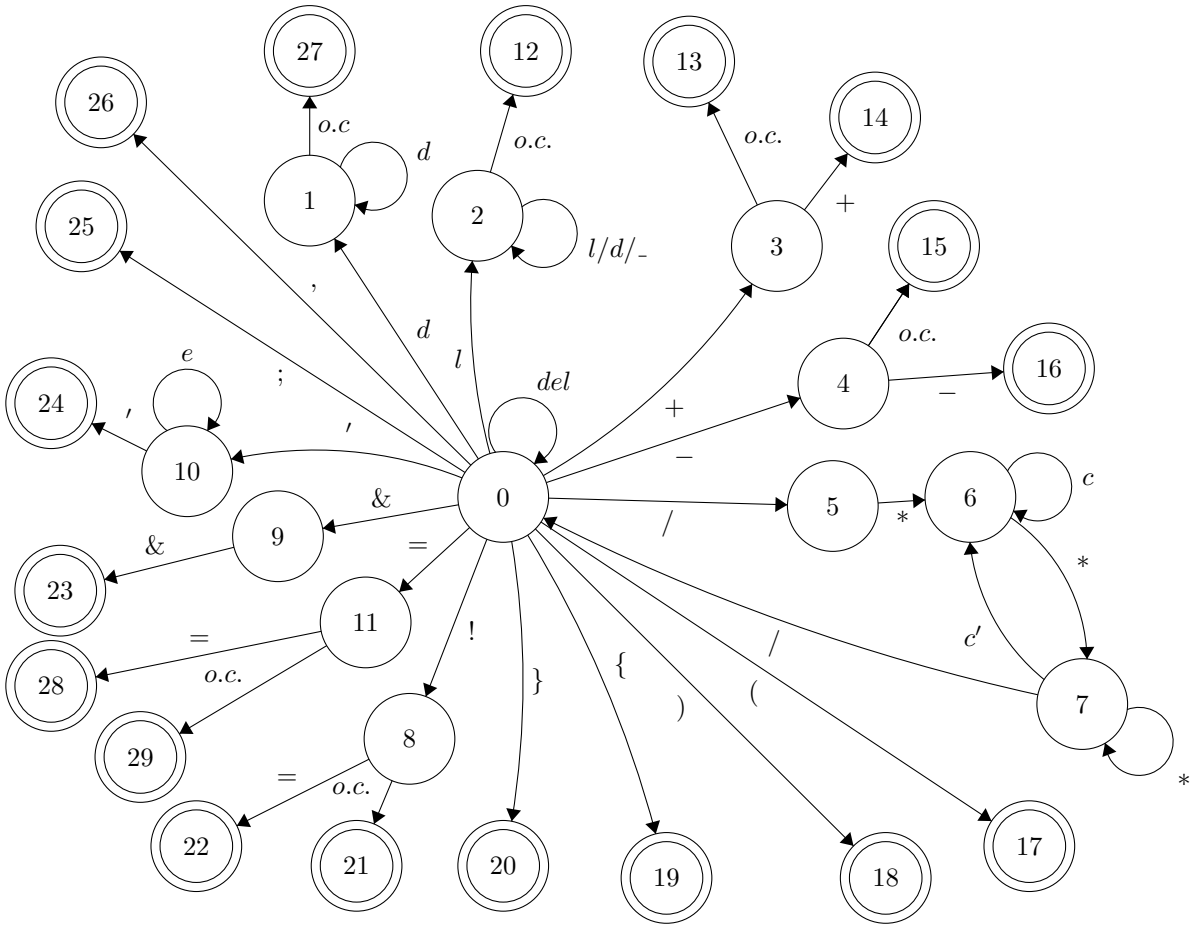
alert	<alert, >
boolean	<boolean, >
else	<else, >
function	<function, >
if	<if, >
input	<input, >
let	<let, >
number	<number, >
return	<return, >
string	<string, >
while	<while, >
false	<false, >
true	<true, >
do	<do, >
autoInc	<Autoincremento (++), >
Número	<constante entera, >
Posición(Número)	<Cadena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

```
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 | ’
```

```
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 Caracter



Leer: Se lee en todos los estados menos en los que pone o.c.
Errores: Cualquier transicion 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")
```

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,equals)
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
```

Prueba 1: *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'>
```

TS:

```
Contenido Tabla Símbolos # 0 :
* LEXEMA : 'a'
  ATRIBUTOS :
    + tipo: unknown
    + despl: unknown
* LEXEMA : 'pp'
  ATRIBUTOS :
    + tipo: unknown
    + despl: unknown
```

Errores