Analizador Léxico y Tabla de Símbolos

Aarón Cabero Blanco
Daniel Tomás Sanchez
Alejandro Cuadrón Lafuente

1 de noviembre de 2020

En este documento se realizará la memoria del analizador léxico y la tabla de símbolos correspondiente a la primera entrega de la practica de PDL.

Índice

1. Tokens

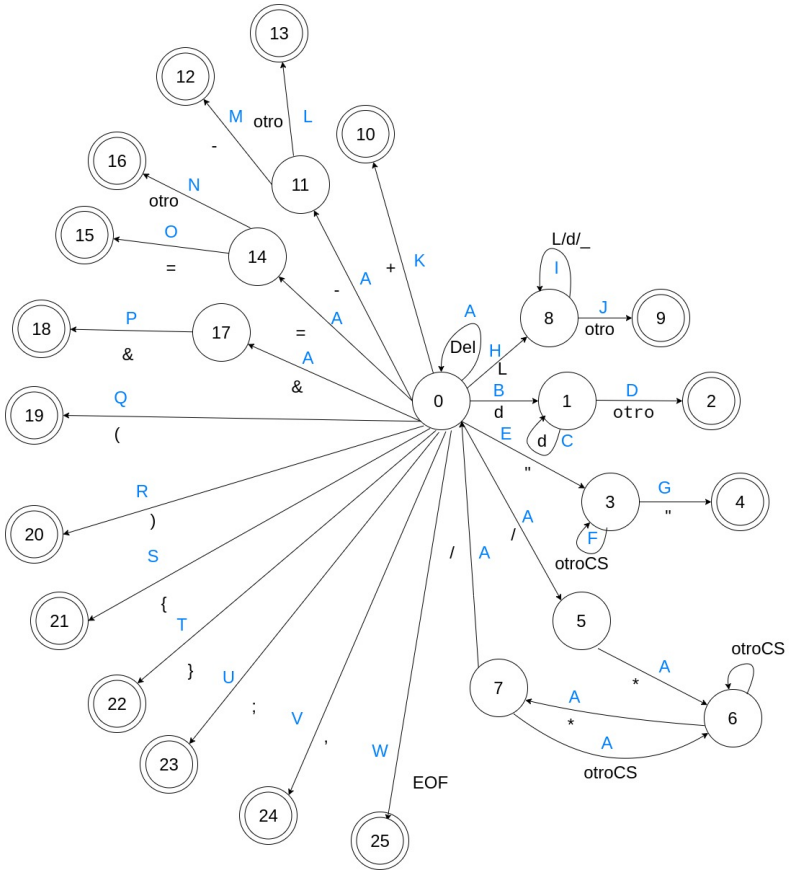
- Identificador <ID, punteroTS>
- Constante Entera <CTEENTERA, valor>
- String <CADENA, lexema>
- False <CTELOGICA, 0>
- True <CTELOGICA, 1>
- Palabra Reservada Number <NUMBER, ->
- Palabra Reservada String <STRING, ->
- Palabra Reservada Boolean <BOOLEAN, ->
- Palabra Reservada Let <LET, ->
- Palabra Reservada Alert <ALERT, ->
- Palabra Reservada Input <INPUT, ->
- Palabra Reservada Function <FUNCTION, ->
- Palabra Reservada Return <RETURN, ->
- Palabra Reservada If <IF, ->
- Palabra Reservada For <FOR, ->
- – <OPESP, ->
- + <OPARIT, 0>
- - <OPARIT, 1>
- = <OPASIGN, ->
- == <OPREL, ->
- && <OPLOG, ->
- (<ABREPAR, ->
-) <CIERRAPAR,>
- { <ABRELLAVE, ->
- } <CIERRALLAVE, ->
- , <COMA, ->
- ; <PUNTOYCOMA, ->
- End Of File <EOF, ->

2. Gramática Regular

Axioma = A
A → del A | dD | "S | /C | lI | + | -M | =E | &N | (|) | { | } | ; | , | EOF
D → dD | λ
S → "| cS
C → *C'
C' → *C" | cC'
C" → /A | cC'
I → dI | lI | _I | λ
M → - | λ
E → = | λ
N → &

Siendo *d* un dígito, *l* una letra, *c* cualquier otro carácter y *del* un delimitador.

3. Autómata Finito Determinista



4. Acciones Semánticas

A: leer
B: number = int(d), leer
C: number = number*10 + int(d), leer
D: if number>32767
    pError("Número fuera de rango")
  else
    genToken(CTEENTERA,number);
E: string = ", contador = 0, leer
F: string = string + otroCS, contador++ leer
G: if contador>64
    pError(Çadena demasiado larga")
  else
    genToken(CADENA,string)
  leer
H: string = l, leer
I: string = string + l/D/_ , leer
J: if palabrasReservadas.contains(string)
    if string == "number"
      genToken(NUMBER,-)
    elif string == "string"
      genToken(STRING,-)
    elif string == "boolean"
      genToken(BOOLEAN,-)
    elif string == "let"
      genToken(LET,-)
    elif string == .ªlert"
      genToken(ALERT,-)
    elif string == ïnput"
      genToken(INPUT,-)
    elif string == return"
      genToken(RETURN,-)
    elif string == ïf"
      genToken(IF,-)
    else
      genToken(FOR,-)
  elif ((puntero = TS.get(string)) == None)
    TS.update(string)
    puntero = TS.get(string);
    genToken(ID,puntero)
K: genToken(OPARIT,0), leer
L: genToken(OPARIT,1)
M: genToken(OPESP,-), leer
N: genToken(OPASIGN, -)
O: genTokeN(OPREL, -), leer

P: genToken(OPLOG, -), leer
Q: genToken(ABREPAR, -), leer
R: genToken(CIERRAPAR, -), leer
S: genToken(ABRELLAVE, -), leer
T: genToken(CIERRALLAVE, -), leer
U: genToken(COMA, -), leer
V: genToken(PUNTOYCOMA, -), leer
W: genToken(EOF, -), leer

5. Errores

Error léxico (siempre se lanza cuando el Analizador Léxico encuentra un error).

1. Cadena con longitud mayor de 64 caracteres.
2. Número fuera de rango (mayor de 32767).
3. Carácter ilegal.

Todo error va acompañado de la *linea* y *columna* en el que se ha encontrado dicho error.

6. Formato inicial de la Tabla de Símbolos

La tabla de símbolos sigue el formato presentado en la página web de la asignatura:

```
TS Global #0:
* Lexema: 'variable1'
* Lexema: 'funcion1'
```

Por el momento la tabla solo dispone de la capacidad de guardar los *tokens* y asignarles un número para su identificación. Los otros módulos del procesador serán los encargados de terminar la tabla añadiendo el resto de campos necesarios.
Las tablas generadas a partir del ámbito de las funciones tendrán el mismo formato presentado arriba pero con un número asignado correspondiente a su posición en la tabla de símbolos global.

7. Casos de Prueba

7.1. Casos de Prueba Correctos

7.1.1. Caso de Prueba Correcto Nº1

El programa empleado es el siguiente:

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

Los *tokens* generados son los siguientes:
<LET , >
<STRING , >
<ID , 0>
<PUNTOYCOMA , >
<FUNCTION , >
<ID , 1>
<ABPAREN , >
<STRING , >
<ID , 2>
<CEPAREN , >
<ABLLAVE , >
<ALERT , >
<ABPAREN , >
<ID , 2>
<CEPAREN , >
<PUNTOYCOMA , >
<CELLAVE , >
<FUNCTION , >
<ID , 3>
<ABPAREN , >
<CEPAREN , >
<ABLLAVE , >
<ALERT , >
<ABPAREN , >
<CADENA , "Introduce un texto">

<CEPAREN , >
<PUNTOYCOMA , >
<INPUT , >
<ABPAREN , >
<ID , 0>
<CEPAREN , >
<PUNTOYCOMA , >
<CELLAVE , >
<ID , 3>
<ABPAREN , >
<CEPAREN , >
<PUNTOYCOMA , >
<LET , >
<STRING , >
<ID , 4>
<PUNTOYCOMA , >
<ID , 4>
<OPASIG , >
<ID , 0>
<PUNTOYCOMA , >
<ID , 1>
<ABPAREN , >
<ID , 4>
<CEPAREN , >
<PUNTOYCOMA , >
<EOF , >
La tabla de símbolos generada es la siguiente:

CONTENIDO DE LA TABLA # 0 :
* LEXEMA : 'texto'
———————————
LEXEMA : 'print'
———————————
LEXEMA : 'msg'
———————————
LEXEMA : 'pideTexto'
———————————
LEXEMA : 'textoAux'
———————————

7.1.2. Caso de Prueba Correcto Nº2

El programa empleado es el siguiente:

```
let string texto;
function print (string msg){
    alert ("Mensage introducido:");
    alert (msg);
} function pideTexto (){
    alert (Ïntroduce un texto");
    input (texto);
}
pideTexto();
print (texto);
```

Los *tokens* generados son los siguientes:

<LET , >
<STRING , >
<ID , 0>
<PUNTOYCOMA , >
<FUNCTION , >
<ID , 1>
<ABPAREN , >
<STRING , >
<ID , 2>
<CEPAREN , >
<ABLLAVE , >
<ALERT , >
<ABPAREN , >
<CADENA , "Mensage introducido:">
<CEPAREN , >
<PUNTOYCOMA , >
<ALERT , >
<ABPAREN , >
<ID , 2>
<CEPAREN , >
<PUNTOYCOMA , >
<CELLAVE , >
<FUNCTION , >
<ID , 3>
<ABPAREN , >
<CEPAREN , >
<ABLLAVE , >
<ALERT , >
<ABPAREN , >
<CADENA , "Introduce un texto">
<CEPAREN , >

<PUNTOYCOMA , >
<INPUT , >
<ABPAREN , >
<ID , 0>
<CEPAREN , >
<PUNTOYCOMA , >
<CELLAVE , >
<ID , 3>
<ABPAREN , >
<CEPAREN , >
<PUNTOYCOMA , >
<ID , 1>
<ABPAREN , >
<ID , 0>
<CEPAREN , >
<PUNTOYCOMA , >
<EOF , >

La tabla de símbolos generada es la siguiente:

CONTENIDO DE LA TABLA # 0 :
* LEXEMA : 'texto'
———————————————
LEXEMA : 'print'
———————————————
LEXEMA : 'msg'
———————————————
LEXEMA : 'pideTexto'
———————————————

7.1.3. Caso de Prueba Correcto Nº3

El programa empleado es el siguiente:

```
let number a;
let number b;
let number int;
alert ("Introduce el primer operando");
input (a);
alert ("Introduce el segundo operando");
input (b);
function number operacion (number num1, number num2){
    return num1 + num2-77;
}
int = 0;
alert (operacion (a, b));
```

Los *tokens* generados son los siguientes:
<LET , >
<NUMBER , >
<ID , 0>
<PUNTOYCOMA , >
<LET , >
<NUMBER , >
<ID , 1>
<PUNTOYCOMA , >
<LET , >
<NUMBER , >
<ID , 2>
<PUNTOYCOMA , >
<ALERT , >
<ABPAREN , >
<CADENA , Ïntroduce el primer operando»
<CEPAREN , >
<PUNTOYCOMA , >
<INPUT , >
<ABPAREN , >
<ID , 0>
<CEPAREN , >
<PUNTOYCOMA , >
<ALERT , >
<ABPAREN , >
<CADENA , Ïntroduce el segundo operando»
<CEPAREN , >
<PUNTOYCOMA , >
<INPUT , >
<ABPAREN , >
<ID , 1>
<CEPAREN , >
<PUNTOYCOMA , >
<FUNCTION , >
<NUMBER , >
<ID , 3>
<ABPAREN , >
<NUMBER , >
<ID , 4>
<COMA , >
<NUMBER , >
<ID , 5>
<CEPAREN , >
<ABLLAVE , >
<RETURN , >

<ID , 4>
<OPARIT , 0>
<ID , 5>
<OPARIT , 1>
<CTEENTERA , 77>
<PUNTOYCOMA , >
<CELLAVE , >
<ID , 2>
<OPASIG , >
<CTEENTERA , 0>
<PUNTOYCOMA , >
<ALERT , >
<ABPAREN , >
<ID , 3>
<ABPAREN , >
<ID , 0>
<COMA , >
<ID , 1>
<CEPAREN , >
<CEPAREN , >
<PUNTOYCOMA , >
<EOF , >

La tabla de símbolos generada es la siguiente:

CONTENIDO DE LA TABLA # 0 :
* LEXEMA : 'a'
————————————————————
LEXEMA : 'b'
————————————————————
LEXEMA : 'int'
————————————————————
LEXEMA : 'operacion'
————————————————————
LEXEMA : 'num1'
————————————————————
LEXEMA : 'num2'
————————————————————

7.2. Casos de Prueba Fallidos

7.2.1. Caso de Prueba Fallido Nº1

El programa empleado es el siguiente:

```
var number b;
b = 5;
print("hola);
```

Errores generados:

Error lexico:
Illegal character ' " ' en la linea 3 y columna 7

7.2.2. Caso de Prueba Fallido Nº2

El programa empleado es el siguiente:

```
var number a=3;
if((a==2) & (b==1))
alert("No se que poner")
```

Errores generados:

Error léxico:
Illegal character '&' en la linea 2 y columna 11

7.2.3. Caso de Prueba Fallido Nº3

El programa empleado es el siguiente:

```
alert("Vamos a crear un número muy grande");
let number x=2341234134;
```

Errores generados:

Error léxico:
Número fuera de rango: "2341234134" en la linea 2 y columna 14