Memoria

Compilador

Grupo 74

Ignacio Gómez Valverde

Óscar Martínez Garcés

Paula Labandeira Campos

**Índice**

1. Diseño Analizador Léxico:
   1. Tokens
   2. Gramática
   3. Autómata
   4. Acciones Semánticas y Errores
2. Diseño inicial de la Tabla de Símbolos
3. Anexo de casos de prueba
   1. Código
   2. Pruebas

1.Diseño del Analizador Léxico

**1.1 Tokens:**

< ALERT , - > (palabra reservada alert)

< BOOLEAN , - > (palabra reservada boolean)

< BREAK, - > (palabra reservada break)

< CASE, - > (palabra reservada case)

< FUNCTION, - > (palabra reservada function)

< IF , - > (palabra reservada if)

< INPUT, - > (palabra reservada input)

<LET , - > (palabra reservada let)

<NUMBER,-> (palabra reservada number)

<RETURN,-> (palabra reservada return)

<STRING,-> (palabra reservada string)

<SWITCH,-> (palabra reservada switch)

<CONSTENTERA, valor> (constante entera)

<CADENA, lexema> (cadena)

<ID, -> (identificador)

<DEFAULT, -> (palabra reservada default)

<EOF, -> (palabra reservada end of file)

<opasig, 1> (=)

<opasig, 2> (|=)

<punt, 1> (,)

<punt, 2> (;)

<punt, 3> (:)

<PARENTABRI, -> ( ‘(‘ )

<PARENTCERR, -> ( ‘)’ )

<oparitmetico, 1> (+)

<oparitmetico, 2> (-)

<oparitmetico, 3> (/)

<oplogico, 1> ( || )

<oprel, 1> (==)

**1.2 Gramática:**

Leyenda:

c = cualquier carácter - {del}

del = {EOL, EOF}

delb = del U {ϸ}, tab

b = l U d U \_

l = letras

d = dígito

S -> delb S **|** /A **|** dB **|** “C **|** lD **|** ϸS | =E | |F | ; S | ( S | ) S | { S | } S | : S | + S | - S | EOF

A -> /A’ **|** λ

A’ -> cA’ **|** del S

B -> dB **|** λ

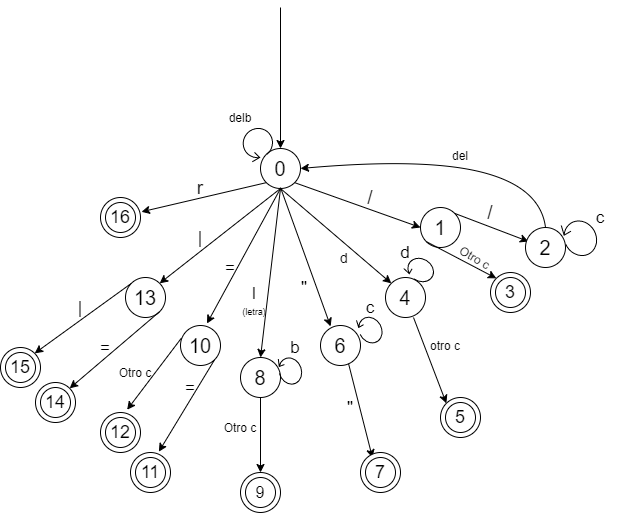
C -> cC **|** “

D -> bD **|** λ

E -> = **|** λ

F -> = **|** |

**1.3 Autómata:**

****

**1.4 Acciones semánticas:**

0-0: Leer

0-1: Leer

1-2: Leer

2-0: Leer

2-2: Leer

1-3: Generar token (op\_arimético,3)

0-4: valor= valor(d); Leer

4-4: valor=valor\*10+valor(d); Leer

4-5: if (valor >2147483647){

error(“Valor fuera de rango.”)

}else{

Generar token (CONST\_ENTERA, valor)

}

0-6: Leer; Lexema⊕Ø

6-6: Lexema:=lexema ⊕c; Leer

6-7: if(cadena\_fuera\_rango()){

error(“Tamaño de cadena excedido.”)

}else{

Generar token (CADENA,lexema); Leer

}

0-8: Lexema:=lexema ⊕ l;Leer

8-8: Lexema:=lexema ⊕ b; Leer

8-9: lex= buscar\_pal\_reservada(lexema)

if lex !=null then{

Gen token (lex,-);

}Else{

if Zona\_Declaracion=true then{

lex= buscar\_TS(lex)

if lex =null then {

p:=insert\_TS(lex)

Generar token(Id, lex)

} Else{ error(“ID ya declarado”)

Else{

lex= buscar\_TS(lex)

if lex= null then error(“el ID no está declarado”)

Else Generar token (Id, lexl)

}

} Leer;

0-10: Leer

10-11: Genera token(op\_logico,1);Leer

10-12:Generar token(op\_asig,1)

0-13: Leer;

13-15: Generar token (|=,-);leer

13-14: Generar token(op\_logico,2);leer

0-16: Generar token(PARENTABRI,-);leer

0-17: Generar token(PARENTCERR,-); leer

0-18:Generar token(CORCHABRI, -); leer

0-19:Generar token(CORCHCERR, -); leer

0-20: Generar token(op\_aritmetico, 1); leer

0-21: Generar token(op\_aritmetico, 2); leer

0-22: Generar token(punt,1);leer

0-23:Generar token (punt, 2); leer

0-24:Generar token(punt,3);leer

0-25:Generar token(EOF,-);

2. Diseño inicial de la Tabla de Símbolos

La tabla de símbolos tiene una organización lineal y memoria dinámica. Actualmente sólo está implementada la función del analizador léxico por lo que sólo se creará una tabla de símbolos para el programa principal. Una vez se implemente el analizador semántico las funciones se guardarán en la pila de tablas de símbolos hasta que estas dejen de ser necesarias (en ese momento se eliminarán para liberar espacio).

Los campos de la tabla de símbolos son:

* Lexema: Lexema del identificador.
* Tipo: Entero, lógico, función o cadena.
* Desplazamiento: Dirección relativa(Para enteros, arrays, lógicos…).
* Número de Parámetros: número de parámetros que recibe de entrada.
* Tipo de Parámetro: El tipo del parámetro que recibe.
* Modo de paso: El modo en el que se pasan los parámetros 0 (referencia) y 1 (valor).
* Tipo de Parámetro: Tipo de parámetro que devuelve la función.
* Etiqueta: Nombre que se le asigna a la función.

La tabla sería de este modo:

**TS\_principal**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Lexema** | **Tipo** | **Desplaz** | **Nºde params** | **Tipo params** | **Modo paso** | **Tipo Dev** | **Etiqueta** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

3.Anexo. Casos de prueba

**CASO 1.**

Codigo

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)

{

varglobal = 03;

if (l1) l2 = ff (ss);

return ss;

}

if (ff(l1)) alert (varglobal);

------------------------------------------------------------------------

Tabla de Simbolos

Tabla Programa Principal # 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'

-------------------------------------------------------------------------

Tokens

<LET, >

<NUMBER, >

<ID, 0>

<punt, 2>

<LET, >

<BOOLEAN, >

<ID, 1>

<punt, 2>

<LET, >

<STRING, >

<ID, 2>

<punt, 2>

<LET, >

<NUMBER, >

<ID, 3>

<punt, 2>

<LET, >

<BOOLEAN, >

<ID, 4>

<punt, 2>

<ALERT, >

<PARENTABRI, >

<CADENA, "PdL">

<PARENTCERR, >

<punt, 2>

<INPUT, >

<PARENTABRI, >

<ID, 5>

<PARENTCERR, >

<punt, 2>

<INPUT, >

<PARENTABRI, >

<ID, 0>

<PARENTCERR, >

<punt, 2>

<ID, 1>

<opasig, 1>

<ID, 4>

<punt, 2>

<IF, >

<PARENTABRI, >

<ID, 1>

<oplogico, 1>

<ID, 4>

<PARENTCERR, >

<ID, 2>

<opasig, 1>

<CADENA, "hello">

<punt, 2>

<ID, 3>

<opasig, 1>

<ID, 0>

<oparitmetico, 2>

<CONSTENTERA, 378>

<punt, 2>

<ALERT, >

<PARENTABRI, >

<CONSTENTERA, 33>

<oparitmetico, 1>

<ID, 0>

<oparitmetico, 1>

<ID, 3>

<PARENTCERR, >

<punt, 2>

<FUNCTION, >

<BOOLEAN, >

<ID, 6>

<PARENTABRI, >

<BOOLEAN, >

<ID, 7>

<PARENTCERR, >

<CORCHABRI, >

<ID, 8>

<opasig, 1>

<CONSTENTERA, 3>

<punt, 2>

<IF, >

<PARENTABRI, >

<ID, 1>

<PARENTCERR, >

<ID, 4>

<opasig, 1>

<ID, 6>

<PARENTABRI, >

<ID, 7>

<PARENTCERR, >

<punt, 2>

<RETURN, >

<ID, 7>

<punt, 2>

<CORCHCERR, >

<IF, >

<PARENTABRI, >

<ID, 6>

<PARENTABRI, >

<ID, 1>

<PARENTCERR, >

<PARENTCERR, >

<ALERT, >

<PARENTABRI, >

<ID, 8>

<PARENTCERR, >

<punt, 2>

<EOF, ->

**CASO 2:**

Código

let number a;

let number b;

let boolean bbb;

a = 3;

b = a;

let boolean c;

c = a == b;

if (c) b = 1;

if (b == a) b = 4;

a = a + b;

alert (a);

alert(b);

------------------------------------------------------------------------------------------------------------------------

Tabla de Símbolos

Tabla Programa Principal # 1:

\* LEXEMA : 'a'

\* LEXEMA : 'b'

\* LEXEMA : 'bbb'

\* LEXEMA : 'c'

------------------------------------------------------------------------------------------------------------------------

Tokens

<LET, >

<NUMBER, >

<ID, 0>

<punt, 2>

<LET, >

<NUMBER, >

<ID, 1>

<punt, 2>

<LET, >

<BOOLEAN, >

<ID, 2>

<punt, 2>

<ID, 0>

<opasig, 1>

<CONSTENTERA, 3>

<punt, 2>

<ID, 1>

<opasig, 1>

<ID, 0>

<punt, 2>

<LET, >

<BOOLEAN, >

<ID, 3>

<punt, 2>

<ID, 3>

<opasig, 1>

<ID, 0>

<oprel, 1>

<ID, 1>

<punt, 2>

<IF, >

<PARENTABRI, >

<ID, 3>

<PARENTCERR, >

<ID, 1>

<opasig, 1>

<CONSTENTERA, 1>

<punt, 2>

<IF, >

<PARENTABRI, >

<ID, 1>

<oprel, 1>

<ID, 0>

<PARENTCERR, >

<ID, 1>

<opasig, 1>

<CONSTENTERA, 4>

<punt, 2>

<ID, 0>

<opasig, 1>

<ID, 0>

<oparitmetico, 1>

<ID, 1>

<punt, 2>

<ALERT, >

<PARENTABRI, >

<ID, 0>

<PARENTCERR, >

<punt, 2>

<ALERT, >

<PARENTABRI, >

<ID, 1>

<PARENTCERR, >

<punt, 2>

<EOF, >

**CASO 3:**

Código

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));

----------------------------------------------------------------------------------------------------------------------

Tabla de Símbolos

Tabla Programa Principal # 1:

\* LEXEMA : 'a'

\* LEXEMA : 'b'

\* LEXEMA : 'int'

\* LEXEMA : 'operacion'

\* LEXEMA : 'num1'

\* LEXEMA : 'num2'

-----------------------------------------------------------------------------------------------------------------------

Tokens

<LET, >

<NUMBER, >

<ID, 0>

<punt, 2>

<LET, >

<NUMBER, >

<ID, 1>

<punt, 2>

<LET, >

<NUMBER, >

<ID, 2>

<punt, 2>

<ALERT, >

<PARENTABRI, >

<CADENA, "Introduce el primer operando">

<PARENTCERR, >

<punt, 2>

<INPUT, >

<PARENTABRI, >

<ID, 0>

<PARENTCERR, >

<punt, 2>

<ALERT, >

<PARENTABRI, >

<CADENA, "Introduce el segundo operando">

<PARENTCERR, >

<punt, 2>

<INPUT, >

<PARENTABRI, >

<ID, 1>

<PARENTCERR, >

<punt, 2>

<FUNCTION, >

<NUMBER, >

<ID, 3>

<PARENTABRI, >

<NUMBER, >

<ID, 4>

<punt, 1>

<NUMBER, >

<ID, 5>

<PARENTCERR, >

<CORCHABRI, >

<RETURN, >

<ID, 4>

<oparitmetico, 1>

<ID, 5>

<oparitmetico, 1>

<CONSTENTERA, 77>

<punt, 2>

<CORCHCERR, >

<ID, 2>

<opasig, 1>

<CONSTENTERA, 0>

<punt, 2>

<ALERT, >

<PARENTABRI, >

<ID, 3>

<PARENTABRI, >

<ID, 0>

<punt, 1>

<ID, 1>

<PARENTCERR, >

<PARENTCERR, >

<punt, 2>

<EOF, >

**CASO 4: (con errores)**

Código

let number a;

let String b;

let number int;

alert ("Introduce el primer operando);

a = 30;

alert ("Introduce el segundo operando");

input (b);

function number operacion (number num1, number num2)

{

return num1 + num2+77;

}

int = 0;

alert (operacion (a, b));

---------------------------------------------------------------------------------------------------------------------------

Tokens

Cadena sin comillas de cierre. Linea: 4

**CASO 5: (con errores)**

Código

let String b;

let number a;

let number int;

a=10;

b=20;

if( a < b){

alert ("Introduce el operando");

input (b);

}

function number operacion (number num1, number num2)

{

return num1 + num2+77;

}

int = 0;

alert (operación (a, b));

---------------------------------------------------------------------------------------------------------------------------

Tokens

Error en el carácter: < . Línea: 6

---------------------------------------------------------------------------------------------------------------------------

**CASO 6: (con errores)**

Código

let number a;

let number b;

let number bbb;

a = 3;

b = a;

if (a == b) b = 1;

a = a / b;

alert (a);

alert(b);

bbb=3000000000;

---------------------------------------------------------------------------------------------------------------------

Tokens

Valor fuera de rango. Linea: 10

La estructura consiste en dos paquetes (tablas y analizadores):

* El paquete tablas contiene las clases Tabla Simbolos (clase principal para manejar la clase de símbolos), Token (para crear objetos de tipo Token con diferentes constructores dependiendo del token definido), TablaPalabrasR (contiene todas las palabras reservadas) y ObjetoTS ( Contiene todos los campos que puede tener un identificador en la Tabla de Símbolos).
* El paquete Analizadores contiene la clase Main (clase principal que se encarga de la entrada y salida del compilador) y la clase AnalizadorLexico (clase que funciona como un analizador léxico generando un token con el método leerToken).