UNIVERSIDAD POLITECNICA DE MADRID

PRACTICA: ANALIZADOR LEXICO

Autores: Jiménez Pérez Juan, N:190204

Hernandez Pérez Jesus, N: 190295

Yanez Soffia Miguel, N: 190322

25 de octubre del 2021

Índice

Contenido

Diseño de la practica

Para el desarrollo de la práctica, hemos escogido el lenguaje de programación Java con la librería JFlex.

Consideraciones

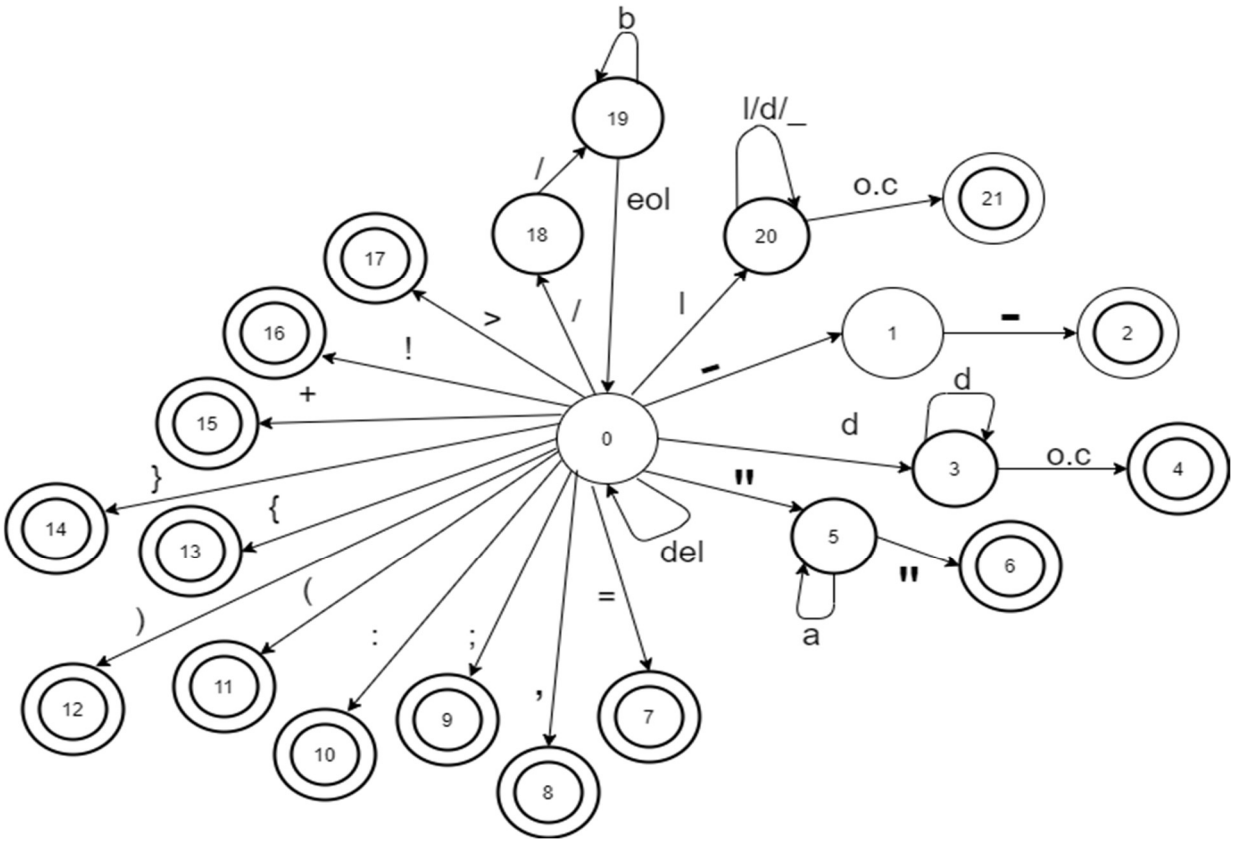
- Hemos utilizado los tokens obligatorios y de tokens de grupo la negación (!) y el mayor (>).
- Los comentarios se hacen con el doble barra diagonal (//) y finalizan con un salto de línea.
- Los enteros están en el rango [0 , 32767].
- Las cadenas están dentro de comillas.
- Un identificador es cualquier combinación de letras, números y barra baja (_) que comience con una letra.

Analizador Léxico

Tokens

- Boolean: <boolean , >
- Break: <break, >
- Case: <case, >
- Function: <function, >
- If: <if, >
- Input: <input, >
- Int: <int, >
- Let: <let, >
- Print: <print, >
- Return: <return, >
- String: <string, >
- Switch: <switch, >
- Autodecremento: <opAutodecremento, >
- Constante entera: <entero, numero >
- Cadena: <cadena, lexema >
- Identificador: <id, numero >
- “=”: <opAsignacion, >
- “,”: <coma, >
- “;”: <puntoComa, >
- “:”: <dosPuntos, >
- “(“: <parentesisAbierto, >
- “)”: < parentesisCerrado, >
- “{“: <llaveAbierta, >
- “}”: <llaveCerrada,

Autómata



Gramática

l = letras minúsculas y mayúsculas

d = {0..9}

c = cualquier carácter

del = {espacio, eol, tab …}

a = c – { ”}

b = c – {eol}

o.c = otro caracter (diferente a letra y digito)

S → del S | l A | - C | d E | “ G | / K | = | , | ; | : | (|) | { | } | + | ! | >

A → l A | d A | _ A | o.c

C → -

E → d E | o.c

G → a G | “

K → / L

L → b L | eol S

Acciones semánticas

0-0: Leer.

0-1: Leer.

1-2: Leer; Gen_Token(opAutodecremento, -).

0-3: Numero = valor_ASCII(d); Leer.

3-3: Numero = numero*10 + valor_ASCII(d); Leer.

3-4: Leer;IF(numero > 32767) THEN error(“Numero fuera de rango.”)
ELSE Gen_Token(entero, numero).

0-5: Lexema := “;Leer

5-5: Lexema := concatenar(Lexema, a); Leer

5-6: Lexema := concatenar(Lexema, “); Leer; Gen_Token(cadena, lexema)

0-7: Leer; Gen_Token(opAsignacion, -)

0-8: Leer; Gen_Token(coma, -)

0-9: Leer; Gen_Token(puntoComa, -)

0-10: Leer; Gen_Token(dosPuntos, -)

0-11: Leer; Gen_Token(parentesisAbierto, -)

0-12: Leer; Gen_Token(parentesisCerrado, -)

0-13: Leer; Gen_Token(llaveAbierta, -)

0-14: Leer; Gen_Token(llaveCerrada, -)

0-15: Leer; Gen_Token(suma, -)

0-16: Leer; Gen_Token(negacion, -)

0-17: Leer; Gen_Token(mayor, -)

0-18: Leer;

18-19: Leer;

19-19: Leer;

19-0: Leer;

0-20: Lexema := l;Leer

20-20: Lexema := concatenar(Lexema, l/d,_); Leer

20-21:

IF lexema == boolean THEN leer; Gen_Token(Boolean, -)

ELSE IF lexema == break THEN leer; Gen_Token(break, -)

ELSE IF lexema == case THEN leer; Gen_Token(case, -)

ELSE IF lexema == function THEN leer; Gen_Token(function, -)

ELSE IF lexema == if THEN leer; Gen_Token(if, -)

ELSE IF lexema == input THEN leer; Gen_Token(input, -)

ELSE IF lexema == int THEN leer; Gen_Token(int, -)

ELSE IF lexema == print THEN leer; Gen_Token(print, -)

ELSE IF lexema == return THEN leer; Gen_Token(return, -)

ELSE IF lexema == string THEN leer; Gen_Token(string, -)

ELSE IF lexema == switch THEN leer; Gen_Token(switch, -)

ELSE IF Zona_Declaracion = true THEN

{ p=buscar_TS (lexema)

IF p=null THEN {p:=insertar_TS(lexema) Gen_Token (identificador, p) }

ELSE error (“identificador ya declarado”) }

ELSE { p=buscar_TS (lexema)

IF p=null THEN error (“identificador NO declarado”)

ELSE Gen_Token (identificador, p) } }

Error

Todas las transiciones no consideradas corresponden a casos de error.

Tabla de símbolos

Para esta primera entrega, existirá solamente una tabla de símbolos (ya que es solo el analizador léxico el que esta trabajando). Así que lo único que sabremos de esta tabla es el lexema.

Casos de prueba

Anexamos a la entrega los casos de prueba, de los cuales tres funcionan correctamente y tres devuelven error. De cada uno podremos ver el código que utilizamos, los tokens que produce y la tabla de símbolos.

Caso 1 (correcto)

Codigo:

```
let string      cadena;
input(cadena);
let boolean logico1;
let boolean logico2;
let int         int2;
int1 = 873;

int2 = 378;
if (! logico2) cadena = " hello";

function ff string(string sss)
{
        global = 33;
        logico1 =  logico2;
        if (logico1) sss = ff (cadena);
        return sss;
}

function funcion string (string logico2)
```

```
{       let int var;

  switch (int1){
        case 0:         logico1 = int1 > int2;break;
        case 8888: print(0);
        case 3333: logico2="";
  }

  return logico2;
}
print(((ff((funcion(cadena))))));
```

Tokens:

```
<let , >
<string , >
<id , 0>
<puntoComa , >
<input , >
<parentesisAbierto , >
<id , 0>
<parentesisCerrado , >
<puntoComa , >
<let , >
<boolean , >
<id , 1>
<puntoComa , >
<let , >
```

<boolean , >
<id , 2>
<puntoComa , >
<let , >
<int , >
<id , 3>
<puntoComa , >
<id , 4>
<opAsignacion , >
<entero , 873>
<puntoComa , >
<id , 3>
<opAsignacion , >
<entero , 378>
<puntoComa , >
<if , >
<parentesisAbierto , >
<negacion , >
<id , 2>
<parentesisCerrado , >
<id , 0>
<opAsignacion , >
<cadena , " hello">
<puntoComa , >
<function , >
<id , 5>
<string , >
<parentesisAbierto , >

<string , >
<id , 6>
<parentesisCerrado , >
<llaveAbierta , >
<id , 7>
<opAsignacion , >
<entero , 33>
<puntoComa , >
<id , 1>
<opAsignacion , >
<id , 2>
<puntoComa , >
<if , >
<parentesisAbierto , >
<id , 1>
<parentesisCerrado , >
<id , 6>
<opAsignacion , >
<id , 5>
<parentesisAbierto , >
<id , 0>
<parentesisCerrado , >
<puntoComa , >
<return , >
<id , 6>
<puntoComa , >
<llaveCerrada , >
<function , >

<id , 8>

<string , >

<parentesisAbierto , >

<string , >

<id , 2>

<parentesisCerrado , >

<llaveAbierta , >

<let , >

<int , >

<id , 9>

<puntoComa , >

<switch , >

<parentesisAbierto , >

<id , 4>

<parentesisCerrado , >

<llaveAbierta , >

<case , >

<entero , 0>

<dosPuntos , >

<id , 1>

<opAsignacion , >

<id , 4>

<mayor , >

<id , 3>

<puntoComa , >

<break , >

<puntoComa , >

<case , >

<entero , 8888>

<dosPuntos , >

<print , >

<parentesisAbierto , >

<entero , 0>

<parentesisCerrado , >

<puntoComa , >

<case , >

<entero , 3333>

<dosPuntos , >

<id , 2>

<opAsignacion , >

<cadena , "">

<puntoComa , >

<llaveCerrada , >

<return , >

<id , 2>

<puntoComa , >

<llaveCerrada , >

<print , >

<parentesisAbierto , >

<parentesisAbierto , >

<parentesisAbierto , >

<id , 5>

<parentesisAbierto , >

<parentesisAbierto , >

<id , 8>

<parentesisAbierto , >

<id , 0>

<parentesisCerrado , >

<parentesisCerrado , >

<parentesisCerrado , >

<parentesisCerrado , >

<parentesisCerrado , >

<parentesisCerrado , >

<puntoComa , >

Tabla de símbolos

CONTENIDOS DE LA TABLA #0 :

* lexema: 'ff'
  Atributos:
--------- ----------
* lexema: 'int2'
  Atributos:
--------- ----------
* lexema: 'sss'
  Atributos:
--------- ----------
* lexema: 'int1'
  Atributos:
--------- ----------
* lexema: 'logico2'
  Atributos:
--------- ----------

* lexema: 'var'
  Atributos:
--------- ----------
* lexema: 'cadena'
  Atributos:
--------- ----------
* lexema: 'logico1'
  Atributos:
--------- ----------
* lexema: 'global'
  Atributos:
--------- ----------
* lexema: 'funcion'
  Atributos:
--------- ----------

Caso 2 (correcto)

Código

```
let int z;
let boolean boolean_1;
let int x;
let string ss;
let int xx;
let boolean boolean_2;

function f1 int(int f1, boolean b1)
{
                    input (z);
```

```
                boolean_1 = ! boolean_2;
                xx = f1+x;
                print//Alerta
                (ss);
                return xx;
}
function f2 boolean( int f2 , boolean b1 )
{
                input (y);
                print ((4+5+77+(088+f2)));
                return (!!b1);
}
x = x + 6+ z+ 1+ (2+ y + 6);

print (f1 (x, f2 (3, boolean_2)));
```

Tokens

```
<let , >
<int , >
<id , 0>
<puntoComa , >
<let , >
<boolean , >
<id , 1>
<puntoComa , >
<let , >
<int , >
<id , 2>
```

<puntoComa , >

<let , >

<string , >

<id , 3>

<puntoComa , >

<let , >

<int , >

<id , 4>

<puntoComa , >

<let , >

<boolean , >

<id , 5>

<puntoComa , >

<function , >

<id , 6>

<int , >

<parentesisAbierto , >

<int , >

<id , 6>

<coma , >

<boolean , >

<id , 7>

<parentesisCerrado , >

<llaveAbierta , >

<input , >

<parentesisAbierto , >

<id , 0>

<parentesisCerrado , >

<puntoComa , >

<id , 1>

<opAsignacion , >

<negacion , >

<id , 5>

<puntoComa , >

<id , 4>

<opAsignacion , >

<id , 6>

<suma , >

<id , 2>

<puntoComa , >

<print , >

<parentesisAbierto , >

<id , 3>

<parentesisCerrado , >

<puntoComa , >

<return , >

<id , 4>

<puntoComa , >

<llaveCerrada , >

<function , >

<id , 8>

<boolean , >

<parentesisAbierto , >

<int , >

<id , 8>

<coma , >

<boolean , >
<id , 7>
<parentesisCerrado , >
<llaveAbierta , >
<input , >
<parentesisAbierto , >
<id , 9>
<parentesisCerrado , >
<puntoComa , >
<print , >
<parentesisAbierto , >
<parentesisAbierto , >
<entero , 4>
<suma , >
<entero , 5>
<suma , >
<entero , 77>
<suma , >
<parentesisAbierto , >
<entero , 088>
<suma , >
<id , 8>
<parentesisCerrado , >
<parentesisCerrado , >
<parentesisCerrado , >
<puntoComa , >
<return , >
<parentesisAbierto , >

<negacion , >
<negacion , >
<id , 7>
<parentesisCerrado , >
<puntoComa , >
<llaveCerrada , >
<id , 2>
<opAsignacion , >
<id , 2>
<suma , >
<entero , 6>
<suma , >
<id , 0>
<suma , >
<entero , 1>
<suma , >
<parentesisAbierto , >
<entero , 2>
<suma , >
<id , 9>
<suma , >
<entero , 6>
<parentesisCerrado , >
<puntoComa , >
<print , >
<parentesisAbierto , >
<id , 6>
<parentesisAbierto , >

<id , 2>

<coma , >

<id , 8>

<parentesisAbierto , >

<entero , 3>

<coma , >

<id , 5>

<parentesisCerrado , >

<parentesisCerrado , >

<parentesisCerrado , >

<puntoComa , >

Tabla de símbolos

CONTENIDOS DE LA TABLA #0 :

* lexema: 'ss'
  Atributos:
--------- ----------
* lexema: 'xx'
  Atributos:
--------- ----------
* lexema: 'x'
  Atributos:
--------- ----------
* lexema: 'y'
  Atributos:
--------- ----------

* lexema: 'z'
  Atributos:
--------- ----------
* lexema: 'boolean_2'
  Atributos:
--------- ----------
* lexema: 'boolean_1'
  Atributos:
--------- ----------
* lexema: 'f1'
  Atributos:
--------- ----------
* lexema: 'f2'
  Atributos:
--------- ----------
* lexema: 'b1'
  Atributos:
--------- ----------

Caso 3 (correcto)

Código

```
let int   a   ;
let int    b  ;
let int number;
print ( "Introduce el primer operando" );
input (a);
print ("Introduce el segundo operando");input(b);
```

```
function  operacion  int(int num_1, int num_2)
{
        return  num_1 + num_2+77;
}

number = 0;
print(operacion(b,a));

}
function f2 boolean( int f2 , boolean b1 )
{
                input (y);
                print ((4+5+77+(088+f2)));
                return (!!b1);
}

z=
  x + 6  + z + 1+ (2 + 6);
let 30;
print (f5 (x, f2 (3, boolean_2)));
```

Tokens

```
<let , >
<int , >
<id , 0>
<puntoComa , >
<let , >
<int , >
```

<id , 1>

<puntoComa , >

<let , >

<int , >

<id , 2>

<puntoComa , >

<print , >

<parentesisAbierto , >

<cadena , "Introduce el primer operando">

<parentesisCerrado , >

<puntoComa , >

<input , >

<parentesisAbierto , >

<id , 0>

<parentesisCerrado , >

<puntoComa , >

<print , >

<parentesisAbierto , >

<cadena , "Introduce el segundo operando">

<parentesisCerrado , >

<puntoComa , >

<input , >

<parentesisAbierto , >

<id , 1>

<parentesisCerrado , >

<puntoComa , >

<function , >

<id , 3>

<int , >
<parentesisAbierto , >
<int , >
<id , 4>
<coma , >
<int , >
<id , 5>
<parentesisCerrado , >
<llaveAbierta , >
<return , >
<id , 4>
<suma , >
<id , 5>
<suma , >
<entero , 77>
<puntoComa , >
<llaveCerrada , >
<id , 2>
<opAsignacion , >
<entero , 0>
<puntoComa , >
<print , >
<parentesisAbierto , >
<id , 3>
<parentesisAbierto , >
<id , 1>
<coma , >
<id , 0>

<parentesisCerrado , >

<parentesisCerrado , >

<puntoComa , >

<llaveCerrada , >

<function , >

<id , 6>

<boolean , >

<parentesisAbierto , >

<int , >

<id , 6>

<coma , >

<boolean , >

<id , 7>

<parentesisCerrado , >

<llaveAbierta , >

<input , >

<parentesisAbierto , >

<id , 8>

<parentesisCerrado , >

<puntoComa , >

<print , >

<parentesisAbierto , >

<parentesisAbierto , >

<entero , 4>

<suma , >

<entero , 5>

<suma , >

<entero , 77>

<suma , >
<parentesisAbierto , >
<entero , 088>
<suma , >
<id , 6>
<parentesisCerrado , >
<parentesisCerrado , >
<parentesisCerrado , >
<puntoComa , >
<return , >
<parentesisAbierto , >
<negacion , >
<negacion , >
<id , 7>
<parentesisCerrado , >
<puntoComa , >
<llaveCerrada , >
<id , 9>
<opAsignacion , >
<id , 10>
<suma , >
<entero , 6>
<suma , >
<id , 9>
<suma , >
<entero , 1>
<suma , >
<parentesisAbierto , >

<entero , 2>

<suma , >

<entero , 6>

<parentesisCerrado , >

<puntoComa , >

<let , >

<entero , 30>

<puntoComa , >

<print , >

<parentesisAbierto , >

<id , 11>

<parentesisAbierto , >

<id , 10>

<coma , >

<id , 6>

<parentesisAbierto , >

<entero , 3>

<coma , >

<id , 12>

<parentesisCerrado , >

<parentesisCerrado , >

<parentesisCerrado , >

<puntoComa , >

Tabla de símbolos

CONTENIDOS DE LA TABLA #0 :

* lexema: 'a'

Atributos:

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

* lexema: 'b'

Atributos:

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

* lexema: 'boolean_2'

Atributos:

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

* lexema: 'operacion'

Atributos:

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

* lexema: 'f2'

Atributos:

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

* lexema: 'b1'

Atributos:

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

* lexema: 'f5'

Atributos:

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

* lexema: 'number'

Atributos:

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

* lexema: 'x'

Atributos:

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

* lexema: 'y'

Atributos:

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

* lexema: 'z'

  Atributos:

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

* lexema: 'num_1'

  Atributos:

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

* lexema: 'num_2'

  Atributos:

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

Caso 4 (error)

Código

```
// Ejemplo de caso de error


let int66 = 32768;

let string      cadena="Las constantes cadena van encerradas entre comillas dobles o entre comillas simples";

input(str);

let boolean bool1;

let boolean logico2;

let int         int2;

int1 = 32;


int2 = 5;

if (! logico2 && bool1) cadena = " analizador";


function ff string(string sss)
```

```
{
        global = 33/5;
        logico1 =  logico2;
        logico1;
        if (logico1 || logico2) sss = ff (cadena);
        return sss;
}

function funcion string (string logico2)
{       let int var;

  switch (int1){
        case 0:         logico1 = int1 < int2;break;
        case 8888: print(0);
        case 3333: logico2="";
  }

  return logico2;
}
```

Tokens

```
<let , >
<id , 0>
<opAsignacion , >
ERROR: Entero 32768 esta fuera de rango
<puntoComa , >
<let , >
```

<string , >

<id , 1>

<opAsignacion , >

ERROR: Cadena "Las constantes cadena van encerradas entre comillas dobles o entre comillas simples" esta fuera de rango

<puntoComa , >

<input , >

<parentesisAbierto , >

<id , 2>

<parentesisCerrado , >

<puntoComa , >

<let , >

<boolean , >

<id , 3>

<puntoComa , >

<let , >

<boolean , >

<id , 4>

<puntoComa , >

<let , >

<int , >

<id , 5>

<puntoComa , >

<id , 6>

<opAsignacion , >

<entero , 32>

<puntoComa , >

<id , 5>

<opAsignacion , >

<entero , 5>

<puntoComa , >

<if , >

<parentesisAbierto , >

<negacion , >

<id , 4>

ERROR: Simbolo & no definido

ERROR: Simbolo & no definido

<id , 3>

<parentesisCerrado , >

<id , 1>

<opAsignacion , >

<cadena , " analizador">

<puntoComa , >

<function , >

<id , 7>

<string , >

<parentesisAbierto , >

<string , >

<id , 8>

<parentesisCerrado , >

<llaveAbierta , >

<id , 9>

<opAsignacion , >

<entero , 33>

ERROR: Simbolo / no definido

<entero , 5>

<puntoComa , >

<id , 10>

<opAsignacion , >

<id , 4>

<puntoComa , >

<id , 10>

<opAsignacion , >

<puntoComa , >

<if , >

<parentesisAbierto , >

<id , 10>

ERROR: Simbolo | no definido

ERROR: Simbolo | no definido

<id , 4>

<parentesisCerrado , >

<id , 8>

<opAsignacion , >

<id , 7>

<parentesisAbierto , >

<id , 1>

<parentesisCerrado , >

<puntoComa , >

<return , >

<id , 8>

<puntoComa , >

<llaveCerrada , >

<function , >

<id , 12>

<string , >

<parentesisAbierto , >

<string , >

<id , 4>

<parentesisCerrado , >

<llaveAbierta , >

<let , >

<int , >

<id , 13>

<puntoComa , >

<switch , >

<parentesisAbierto , >

<id , 6>

<parentesisCerrado , >

<llaveAbierta , >

<case , >

<entero , 0>

<dosPuntos , >

<id , 10>

<opAsignacion , >

<id , 6>

ERROR: Simbolo < no definido

<id , 5>

<puntoComa , >

<break , >

<puntoComa , >

<case , >

<entero , 8888>

<dosPuntos , >

<print , >

<parentesisAbierto , >

<entero , 0>

<parentesisCerrado , >

<puntoComa , >

<case , >

<entero , 3333>

<dosPuntos , >

<id , 4>

<opAsignacion , >

<cadena , "">

<puntoComa , >

<llaveCerrada , >

<return , >

<id , 4>

<puntoComa , >

<llaveCerrada , >

Tabla de símbolos

CONTENIDOS DE LA TABLA #0 :

* lexema: 'ff'
  Atributos:
--------- ----------
* lexema: 'logico2'
  Atributos:
--------- ----------
* lexema: 'var'

Atributos:

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

* lexema: 'cadena'

Atributos:

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

* lexema: 'logico1'

Atributos:

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

* lexema: 'global'

Atributos:

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

* lexema: 'str'

Atributos:

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

* lexema: 'int2'

Atributos:

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

* lexema: 'sss'

Atributos:

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

* lexema: 'int1'

Atributos:

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

* lexema: 'int66'

Atributos:

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

* lexema: 'true'

Atributos:

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

* lexema: 'funcion'

  Atributos:

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

* lexema: 'bool1'

  Atributos:

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

Caso 5 (error)

Código

```
let int z=4%2;

function f1 int(int z)
{
		input (x);
		let string = "cadena superior a los caracteres permitidos por javaScript";
		x = 5624642;
		print(z)//Imprimimos z
}

let string ss;
let int xx;
let boolean boolean_2;

function funcionF2 boolean(int f1, boolean b1)
{
```

```
                input (z);
                boolean_1 = ! boolean_2;
                xx = f1+x;
                print//Alerta
                (ss);
                return xx;
}



function funcionF2 string( int f2 , boolean b1 )
{
                input (y);
                print ((4+5+77+(088+f2)));
                return (b1<b2);
}

x =x + 6458554+ z * 1/(2+ aux* 6);
print (x);
```

Tokens

```
<let , >
<int , >
<id , 0>
<opAsignacion , >
<entero , 4>
ERROR: Simbolo % no definido
<entero , 2>
```

<puntoComa , >

<function , >

<id , 1>

<int , >

<parentesisAbierto , >

<int , >

<id , 0>

<parentesisCerrado , >

<llaveAbierta , >

<input , >

<parentesisAbierto , >

<id , 2>

<parentesisCerrado , >

<puntoComa , >

<let , >

<string , >

<opAsignacion , >

<cadena , "cadena superior a los caracteres permitidos por javaScript">

<puntoComa , >

<id , 2>

<opAsignacion , >

ERROR: Entero 5624642 esta fuera de rango

<puntoComa , >

<print , >

<parentesisAbierto , >

<id , 0>

<parentesisCerrado , >

<llaveCerrada , >

<let , >
<string , >
<id , 3>
<puntoComa , >
<let , >
<int , >
<id , 4>
<puntoComa , >
<let , >
<boolean , >
<id , 5>
<puntoComa , >
<function , >
<id , 6>
<boolean , >
<parentesisAbierto , >
<int , >
<id , 1>
<coma , >
<boolean , >
<id , 7>
<parentesisCerrado , >
<llaveAbierta , >
<input , >
<parentesisAbierto , >
<id , 0>
<parentesisCerrado , >
<puntoComa , >

<id , 8>

<opAsignacion , >

<negacion , >

<id , 5>

<puntoComa , >

<id , 4>

<opAsignacion , >

<id , 1>

<suma , >

<id , 2>

<puntoComa , >

<print , >

<parentesisAbierto , >

<id , 3>

<parentesisCerrado , >

<puntoComa , >

<return , >

<id , 4>

<puntoComa , >

<llaveCerrada , >

<function , >

<id , 6>

<string , >

<parentesisAbierto , >

<int , >

<id , 9>

<coma , >

<boolean , >

<id , 7>
<parentesisCerrado , >
<llaveAbierta , >
<input , >
<parentesisAbierto , >
<id , 10>
<parentesisCerrado , >
<puntoComa , >
<print , >
<parentesisAbierto , >
<parentesisAbierto , >
<entero , 4>
<suma , >
<entero , 5>
<suma , >
<entero , 77>
<suma , >
<parentesisAbierto , >
<entero , 088>
<suma , >
<id , 9>
<parentesisCerrado , >
<parentesisCerrado , >
<parentesisCerrado , >
<puntoComa , >
<return , >
<parentesisAbierto , >
<id , 7>

ERROR: Simbolo < no definido

<id , 11>

<parentesisCerrado , >

<puntoComa , >

<llaveCerrada , >

<id , 2>

<opAsignacion , >

<id , 2>

<suma , >

ERROR: Entero 6458554 esta fuera de rango

<suma , >

<id , 0>

ERROR: Simbolo * no definido

<entero , 1>

ERROR: Simbolo / no definido

<parentesisAbierto , >

<entero , 2>

<suma , >

<id , 12>

ERROR: Simbolo * no definido

<entero , 6>

<parentesisCerrado , >

<puntoComa , >

<print , >

<parentesisAbierto , >

<id , 2>

<parentesisCerrado , >

<puntoComa , >

Tabla de símbolos

CONTENIDOS DE LA TABLA #0 :

* lexema: 'ss'
  Atributos:
--------- ----------
* lexema: 'xx'
  Atributos:
--------- ----------
* lexema: 'aux'
  Atributos:
--------- ----------
* lexema: 'boolean_2'
  Atributos:
--------- ----------
* lexema: 'f1'
  Atributos:
--------- ----------
* lexema: 'boolean_1'
  Atributos:
--------- ----------
* lexema: 'f2'
  Atributos:
--------- ----------
* lexema: 'b1'
  Atributos:
--------- ----------

* lexema: 'b2'
  Atributos:
--------- ----------
* lexema: 'funcionF2'
  Atributos:
--------- ----------
* lexema: 'x'
  Atributos:
--------- ----------
* lexema: 'y'
  Atributos:
--------- ----------
* lexema: 'z'
  Atributos:
--------- ----------

Caso 6 (error)

Código

```
let int x;
let boolean b;
let int z= 55346436;
input (x);
print (x);

input (z);
print (x+z);
print("cadena de caracteres superior al permitido por el lenguaje javaScript");
```

```
b=x<z;if (b||z)



x =x - 6* z+ 1+ [(2%y)/7)];



let int     n1;
let boolean l1;
let string  cad;
let boolean l2;
input (n1);
l1 = l2;
if (! l2) cad = "hello";
n2 = n1 + 378;

print(6534756);
function ff boolean(boolean ss)
{
          l2 = ! l1 && l2;
          if (l2) l1 = ff (ss);
          varglobal =7567844;
          return (ss);
}
if (ff(l1))
    print (varglobal);
```

Tokens

<let , >

<int , >

<id , 0>

<puntoComa , >

<let , >

<boolean , >

<id , 1>

<puntoComa , >

<let , >

<int , >

<id , 2>

<opAsignacion , >

ERROR: Entero 55346436 esta fuera de rango

<puntoComa , >

<input , >

<parentesisAbierto , >

<id , 0>

<parentesisCerrado , >

<puntoComa , >

<print , >

<parentesisAbierto , >

<id , 0>

<parentesisCerrado , >

<puntoComa , >

<input , >

<parentesisAbierto , >

<id , 2>

<parentesisCerrado , >

<puntoComa , >

<print , >

<parentesisAbierto , >

<id , 0>

<suma , >

<id , 2>

<parentesisCerrado , >

<puntoComa , >

<print , >

<parentesisAbierto , >

ERROR: Cadena "cadena de caracteres superior al permitido por el lenguaje javaScript" esta fuera de rango

<parentesisCerrado , >

<puntoComa , >

<id , 1>

<opAsignacion , >

<id , 0>

ERROR: Simbolo < no definido

<id , 2>

<puntoComa , >

<if , >

<parentesisAbierto , >

<id , 1>

ERROR: Simbolo | no definido

ERROR: Simbolo | no definido

<id , 2>

<parentesisCerrado , >

<id , 0>

<opAsignacion , >

<id , 0>

ERROR: Simbolo - no definido

<entero , 6>

ERROR: Simbolo * no definido

<id , 2>

<suma , >

<entero , 1>

<suma , >

ERROR: Simbolo [no definido

<parentesisAbierto , >

<entero , 2>

ERROR: Simbolo % no definido

<id , 3>

<parentesisCerrado , >

ERROR: Simbolo / no definido

<entero , 7>

<parentesisCerrado , >

ERROR: Simbolo] no definido

<puntoComa , >

<let , >

<int , >

<id , 4>

<puntoComa , >

<let , >

<boolean , >

<id , 5>

<puntoComa , >
<let , >
<string , >
<id , 6>
<puntoComa , >
<let , >
<boolean , >
<id , 7>
<puntoComa , >
<input , >
<parentesisAbierto , >
<id , 4>
<parentesisCerrado , >
<puntoComa , >
<id , 5>
<opAsignacion , >
<id , 7>
<puntoComa , >
<if , >
<parentesisAbierto , >
<negacion , >
<id , 7>
<parentesisCerrado , >
<id , 6>
<opAsignacion , >
<cadena , "hello">
<puntoComa , >
<id , 8>

<opAsignacion , >
<id , 4>
<suma , >
<entero , 378>
<puntoComa , >
<print , >
<parentesisAbierto , >
ERROR: Entero 6534756 esta fuera de rango
<parentesisCerrado , >
<puntoComa , >
<function , >
<id , 9>
<boolean , >
<parentesisAbierto , >
<boolean , >
<id , 10>
<parentesisCerrado , >
<llaveAbierta , >
<id , 7>
<opAsignacion , >
<negacion , >
<id , 5>
ERROR: Simbolo & no definido
ERROR: Simbolo & no definido
<id , 7>
<puntoComa , >
<if , >
<parentesisAbierto , >

<id , 7>

<parentesisCerrado , >

<id , 5>

<opAsignacion , >

<id , 9>

<parentesisAbierto , >

<id , 10>

<parentesisCerrado , >

<puntoComa , >

<id , 11>

<opAsignacion , >

ERROR: Entero 7567844 esta fuera de rango

<puntoComa , >

<return , >

<parentesisAbierto , >

<id , 10>

<parentesisCerrado , >

<puntoComa , >

<llaveCerrada , >

<if , >

<parentesisAbierto , >

<id , 9>

<parentesisAbierto , >

<id , 5>

<parentesisCerrado , >

<parentesisCerrado , >

<print , >

<parentesisAbierto , >

<id , 11>

<parentesisCerrado , >

<puntoComa , >

Tabla de símbolos

CONTENIDOS DE LA TABLA #0 :

* lexema: 'ff'
  Atributos:

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

* lexema: 'ss'
  Atributos:

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

* lexema: 'b'
  Atributos:

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

* lexema: 'n1'
  Atributos:

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

* lexema: 'n2'
  Atributos:

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

* lexema: 'l1'
  Atributos:

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

* lexema: 'l2'
  Atributos:

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

* lexema: 'varglobal'

Atributos:

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

* lexema: 'cad'

Atributos:

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

* lexema: 'x'

Atributos:

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

* lexema: 'y'

Atributos:

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

* lexema: 'z'

Atributos:

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