**DAVID DE FRUTOS ZAFRA**

**PABLO GARCÍA GARCÍA**

**GRUPO 119**

|  |  |
| --- | --- |
| ANALIZADOR LÉXICO Y TABLA DE SÍMBOLOS  PRIMERA ENTREGA - PDL | cONTENIDO  - Diseño del Analizador Léxico (tokens, gramática, autómata, acciones semánticas y errores)  - Diseño inicial de la Tabla de Símbolos (descripción de su estructura y organización).  - Anexo con 6 casos de prueba para mostrar el funcionamiento del Procesador construido |

**ANALIZADOR LÉXICO**

**DISEÑO DE TOKENS**

Constante Entera: <NumEnt, valor>

Identificador: <Id, posTS>

String: <String, ->

Int: <Int, ->

Boolean: <Boolean, ->

If: <If, ->

Else: <Else, ->

Let:<Let, ->

Print: <Print, ->

Input: <Input, ->

Return: <Return, ->

Function: <Function, ->

Cadena: <Cadena, Lexema>

Operador suma: <Suma, ->

Operador resta: <Resta, ->

Operador de asignación: <Asig, ->

Operador de asignación con resta: <AsigResta, ->

Operador Lógico And: <And, ->

Operador Menor: <Menor, ->

Operador Mayor: <Mayor, ->

Paréntesis abierto: <PAbierto, ->

Paréntesis cerrado: <PCerrado ->

Llave Abierta: <KAbierta, ->

Llave Cerrada: <KCerrada, ->

Coma: <Coma, ->

Punto y Coma: <PuntComa, ->

**DISEÑO DE GRAMÁTICA**

S → LA | lA | dB| -C | /D | = | < | >| ‘E | &G | + | ( | ) | { | } | , | ; | del S

A → LA | lA | dA | \_A |

L : letra mayúscula l: letra minúscula

d: digíto (0…9)

c: cualquier tipo de carácter o símbolo excepto: ’

c’: cualquier tipo de carácter que no sea eol

B → dB |

C → = |

D → /F

E → cE | ’

F → c’F | eolS

G → &

Patrón de fondo

Descripción generada automáticamente**DISEÑO DE AUTÓMATA**

**ACCIONES SEMÁNTICAS**

Para comprobar las acciones semánticas que se generan se verán paso a paso cada una de las transiciones del autómata previamente diseñado.

0-0: Leer

0-1: lexema = l | L; Leer

1-1: lexema += (l | L | d | \_); Leer

1-7: pal = buscarPalReservada(lexema)

If pal != null -> Gen\_Token(pal,-)

else buscar lexema en TS

if no esta -> añadir a TS: Gen\_Token(Id,posTS); Leer

0-2: valor = valorA(d); Leer

2-2: valor = valor\*10 + valorA(d); Leer

2-8: if valor > 32767 then Error(“El máximo entero valido será el 32767”)

else Gen\_Token(NumEnt,valor); Leer

0-3: Leer

3-9: Gen\_Token(AsigResta,-); Leer

3-10: Gen\_Token(Resta,-); Leer

0-4: Leer

4-5: Leer

5-5: Leer

5-0: Leer

0-6: cont = 0; Leer

6-6: lexema += c; cont += 1; Leer

6-11: if cont > 64 then Error(“Una cadena no puede contener más de 64 caracteres”)

else Gen\_Token(Cadena, lexema); Leer

0-12: Gen\_Token(PuntComa,-); Leer

0-13: Gen\_Token(Coma,-); Leer

0-14: Gen\_Token(KCerrada,-); Leer

0-15: Gen\_Token(KAbierta,-); Leer

0-16: Gen\_Token(PCerrado,-); Leer

0-17: Gen\_Token(PAbierto,-); Leer

0-18: Leer

18-23: Gen\_Token(And,-); Leer

0-19: Gen\_Token(Suma,-); Leer

0-20: Gen\_Token(Mayor,-); Leer

0-21: Gen\_Token(Menor,-); Leer

0-22: Gen\_Token(Asig,-); Leer

**ERRORES**

Los ya descritos en las transiciones 2-8 y 6-11 en referencia a las limitaciones del lenguaje en la representación de enteros y de cadenas.

Si llega un carácter que no pertenece al alfabeto de entrada.

Cualquier transición no descrita en el autómata generará un caso de error.

**DISEÑO TABLA DE SÍMBOLOS**

En nuestro programa vamos a seguir la siguiente estructura para rellenar la tabla de símbolos:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Lexema | Tipo | Despl | numParam | TipoParamXX | TipoRetorno | EtiqFuncion |

Siendo cada columna:

* Lexema: nombre identificador
* Tipo: representa el tipo del identificador.
* Despl: valor numérico que representa la dirección relativa que tendrá cada variable (el desplazamiento).
* numParam: valor numérico que representa el número de parámetros formales que tiene un identificador de tipo subprograma.
* TipoParamXX: representa el tipo del XXº parámetro de un subprograma. XX representa un número de hasta dos dígitos, cuyos valores irán desde el 1 hasta el valor del atributo numParam.
* TipoRetorno: representa el tipo que devuelve un identificador de tipo función.
* EtiqFuncion: representa la etiqueta que se asocia a un identificador de tipo función.

En nuestra tabla no añadimos ModoParamXX y Param debido a que vamos a realizar nuestro compilador en java y vamos a utilizar siempre el paso por valor.

Esta tabla la vamos a rellenar siguiendo el formato pedido, se pasará a un fichero externo (TS.txt) para poder ver todos los cambios mientras se va almacenando y destruyendo en memoria. También tendremos en cuenta las posibles anidaciones.

La tabla de símbolos funciona de la siguiente manera:

1. Al generar un identificador (Transición 1-7), se comprueba si ya está en la tabla.
2. En caso de que no estuviera, se añade una nueva entrada rellenando los datos de está con la información disponible.
3. Si se lee una nueva función se genera una nueva tabla de símbolos especifica. Esta tabla “hija” será destruida al terminar la función.

**PRUEBAS**

***PRUEBA 1****:* **correcta**

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 num1\_, int num2\_)

{

let int number;

number = 88 - num1\_-num2\_;

return number;

}

number = 0;

print(operacion(b,a));

*TOKENS.txt*

<let,>

<int,>

<id,1>

<puntComa,>

<let,>

<int,>

<id,2>

<puntComa,>

<let,>

<int,>

<id,3>

<puntComa,>

<print,>

<pAbierto,>

<cadena,Introduce el primer operando>

<pCerrado,>

<puntComa,>

<input,>

<pAbierto,>

<id,1>

<pCerrado,>

<puntComa,>

<print,>

<pAbierto,>

<cadena,Introduce el segundo operando>

<pCerrado,>

<puntComa,>

<input,>

<pAbierto,>

<id,2>

<pCerrado,>

<puntComa,>

<function,>

<id,4>

<int,>

<pAbierto,>

<int,>

<id,5>

<coma,>

<int,>

<id,6>

<pCerrado,>

<kabierta,>

<let,>

<int,>

<id,3>

<puntComa,>

<id,3>

<asig,>

<numEnt,8888>

<resta,>

<id,5>

<resta,>

<id,6>

<puntComa,>

<return,>

<id,3>

<puntComa,>

<kCerrada,>

<id,3>

<asig,>

<numEnt,0>

<puntComa,>

<print,>

<pAbierto,>

<id,4>

<pAbierto,>

<id,2>

<coma,>

<id,1>

<pCerrado,>

<pCerrado,>

<puntComa,>

*TS.txt*

TablaGlobal #01:

\*'a'

\*'b'

\*'number'

\*'operacion'

\*'num1\_'

\*'num2\_'

***PRUEBA 2****:* **correcta**

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 num2,int num1)

{

let int res;

res=num1-num2;

return ((res));

}

number = 0;

print(operacion(b,a));

*TOKENS.txt*

<let,>

<int,>

<id,1>

<puntComa,>

<let,>

<int,>

<id,2>

<puntComa,>

<let,>

<int,>

<id,3>

<puntComa,>

<print,>

<pAbierto,>

<cadena,Introduce el primer operando>

<pCerrado,>

<puntComa,>

<input,>

<pAbierto,>

<id,1>

<pCerrado,>

<puntComa,>

<print,>

<pAbierto,>

<cadena,Introduce el segundo operando>

<pCerrado,>

<puntComa,>

<input,>

<pAbierto,>

<id,2>

<pCerrado,>

<puntComa,>

<function,>

<id,4>

<int,>

<pAbierto,>

<int,>

<id,5>

<coma,>

<int,>

<id,6>

<pCerrado,>

<kabierta,>

<let,>

<int,>

<id,7>

<puntComa,>

<id,7>

<asig,>

<id,6>

<resta,>

<id,5>

<puntComa,>

<return,>

<pAbierto,>

<pAbierto,>

<id,7>

<pCerrado,>

<pCerrado,>

<puntComa,>

<kCerrada,>

<id,3>

<asig,>

<numEnt,0>

<puntComa,>

<print,>

<pAbierto,>

<id,4>

<pAbierto,>

<id,2>

<coma,>

<id,1>

<pCerrado,>

<pCerrado,>

<puntComa,>

*TS.txt*

TablaGlobal #01:

\*'a'

\*'b'

\*'number'

\*'operacion'

\*'num2'

\*'num1'

\*'res'

***PRUEBA 3:*****correcta**

let int n1;let int n2;

let boolean l1;let boolean l2;

let

string

cad

;

input (n1);

l1 = l2;

if (! l2) cad = 'hello';

n2 -= n1 - 378;

print( 33

-

n1

+

n2);

function ff boolean(boolean ss)

{

l2 = l1;

if (l2) l1 = ff (ss);

varglobal = 8888;

return (ss);

}

if (ff(l1))

print (varglobal);

*TOKENS.txt*

<let,>

<int,>

<id,1>

<puntComa,>

<let,>

<int,>

<id,2>

<puntComa,>

<let,>

<boolean,>

<id,3>

<puntComa,>

<let,>

<boolean,>

<id,4>

<puntComa,>

<let,>

<string,>

<id,5>

<puntComa,>

<input,>

<pAbierto,>

<id,1>

<pCerrado,>

<puntComa,>

<id,3>

<asig,>

<id,4>

<puntComa,>

<if,>

<pAbierto,>

<negacion,>

<id,4>

<pCerrado,>

<id,5>

<asig,>

<cadena,hello>

<puntComa,>

<id,2>

<asigResta,>

<id,1>

<resta,>

<numEnt,378>

<puntComa,>

<print,>

<pAbierto,>

<numEnt,33>

<resta,>

<id,1>

<suma,>

<id,2>

<pCerrado,>

<puntComa,>

<function,>

<id,6>

<boolean,>

<pAbierto,>

<boolean,>

<id,7>

<pCerrado,>

<kabierta,>

<id,4>

<asig,>

<id,3>

<puntComa,>

<if,>

<pAbierto,>

<id,4>

<pCerrado,>

<id,3>

<asig,>

<id,6>

<pAbierto,>

<id,7>

<pCerrado,>

<puntComa,>

<id,8>

<asig,>

<numEnt,8888>

<puntComa,>

<return,>

<pAbierto,>

<id,7>

<pCerrado,>

<puntComa,>

<kCerrada,>

<if,>

<pAbierto,>

<id,6>

<pAbierto,>

<id,3>

<pCerrado,>

<pCerrado,>

<print,>

<pAbierto,>

<id,8>

<pCerrado,>

<puntComa,>

*TS.txt*

TablaGlobal #01:

\*'n1'

\*'n2'

\*'l1'

\*'l2'

\*'cad'

\*'ff'

\*'ss'

\*'varglobal'

***PRUEBA 4****:* **incorrecta**

let int a ;

let int 1b ;

let int number;

print ( 'Introduce el primer operandoIntroduce el primer operandoIntroduce el primer operandoIntroduce el primer operando' );

input (a);

print ('Introduce el segundo operando');input(b);

function operacion

int (int num1\_, int num2\_)

{

let int number;

number = 888888 - num1\_-num2\_;

return number;

}

number = 0;

print(operacion(b,a));

SALIDA:

Error Linea 4: la cadena ha excedido el maximo de caracteres.

Error Linea 11: el valor supera el entero maximo del lenguaje

***PRUEBA 5****:* **incorrecta**

let boolean b;let intb x;

input (2x);

print (x);

input (\_z);

print (x+z);

b=x<=z;if (b)

x =

x + 67777

- z

- 1

+ (2

- y

+ 7);

SALIDA:

Error Linea 6: entrada no reconocida

Error Linea 10: el valor supera el entero maximo del lenguaje

***PRUEBA 6****:* **incorrecta**

let int n1\_2;

let boolean \_l1;

let string cad;

let int n2;

let boolean l2;

input (n1111111111);

l1 += l2;

if (! l2) cad = 'hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello hello’;

n2 -= n1 + 378;

print( 33

+

n1

-

n2);

function ff boolean(boolean ss)

{

varglobal = 8;

if (l1) l2 = ff (ss);

return ss;

}

if (ff(l2))

print (varglobal);

SALIDA:

Error Linea 2: entrada no reconocida

Error Linea 8: la cadena ha excedido el maximo de caracteres.