Manual Técnico

Antes de modificar el software lea cuidadosamente este instructivo.

Como Modificar el código

El siguiente programa fue creado en el lenguaje de C# con la ayuda de Visual Studio este código consta de métodos y un formulario el cual forma parte de la interfaz gráfica del programa. Para modificar los métodos ver comentarios dentro del código.

Conjuntos:

L = {a..z, A..Z}

d = {0..9}

A = {“{“, “}”, “[“, “]”, “(“, “)”}

B = {“.”, “;”, “*”, “ “ “, “’”}

C = {“+”, “- “, “=”, “/”, “.”, “<”, “>”, “!”}

Expresión Regular

{d+ (.d+)?# **|** (A|B|C)# **|** L(L|d|_)*# **|** (“string+”|’string+’)#**|** //(*)? String+ (*//)? # **}#**

Tabla de siguientes

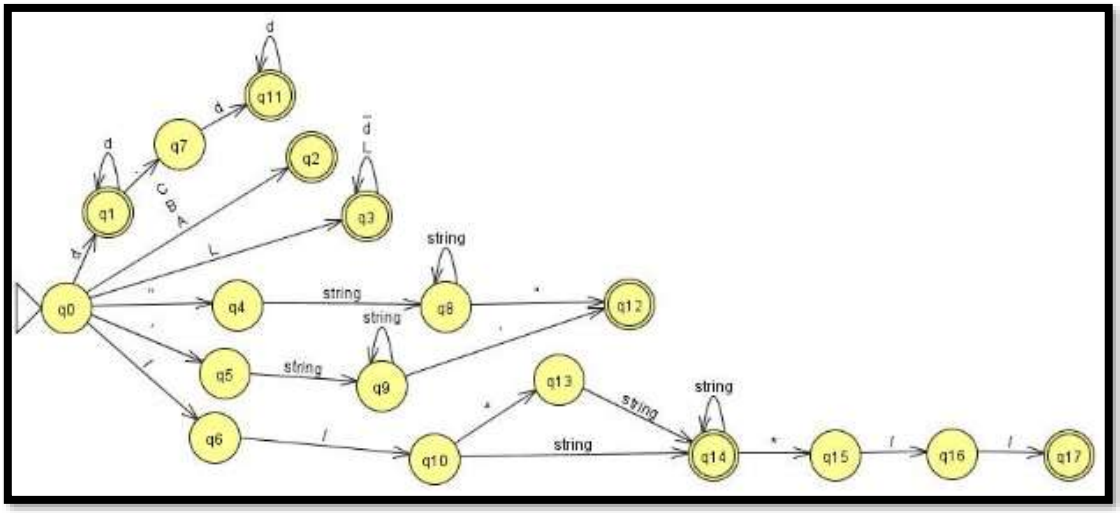
No.	∑	Siguientes
1	d	1,2,4
2	.	3
3	D	3,4
4	#	----------
5	A	8
6	B	8
7	C	8
8	#	----------
9	L	10,11,12
10	L	10,11,12
11	D	10,11,12
12	_	10,11,12
13	#	----------
14	“	15
15	String	15,16
16	“	20
17	‘	18
18	String	18,19
19	‘	20
20	#	----------
21	/	22
22	/	23,24
23	*	24
24	String	24,25,28

25	*	26
26	/	27
27	/	28
28	#	----------

Tabla de Transiciones

Estados\∑	d	.	A	B	C	L	“	string	‘	/	*	_	FDC
S0(1,5,6,7,9,14,17,21)	S1	---	S2	S2	S2	S3	S4	---	S5	S6	---	---	Error
S1(1,2,4)	S1	S7	---	---	---	---	---	---	---	---	---	---	Aceptar
S2(8)	---	---	---	---	---	---	---	---	---	---	---	---	Aceptar
S3(10,11,12,13)	S3	---	---	---	---	S3	---	---	---	---	---	S3	Aceptar
S4(15)								S8					Error
S5(18)								S9					Error
S6(22)										S10			Error
S7(3)	S11												Error
S8(15,16)							S12	S8					Error
S9(18,19)								S9	S12				Error
S10(23,24)								S14			S13		Error
S11(3,4)	S11												Aceptar
S12(20)													Aceptar
S13(24)								S14					Error
S14(24,25,28)								S14			S15		Aceptar
S15(26)										S16			Error
S16(27)										S17			Error
S17(28)	---	---	---	---	---	---	---	---	---	---	---	---	Aceptar

Autómata Finito Determinista (AFD)



Gramática

<INICIO> := class cadena {<ESTRUCTURA>}
<ESTRUCTURA> := static void Main(string[] args){<INSTRUCCION>}
<INSTRUCCION> := <DECLARACION> <INSTRUCCION>
| <IMPRIMIR> <INSTRUCCION>
| <SWITCH> <INSTRUCCION>
| <IF> <INSTRUCCION>
| <FOR> <INSTRUCCION>
<DECLARACION> := <TIPO> <LISTA_VAR>
| <LISTA_VAR>
<TIPO> := int
| float
| char
| string
| String
| bool
<LISTA_VAR> := [] cadena = <VALOR_ARREGLO> <LISTA_VAR>
| cadena <VALOR_ASIGNACION> <LISTA_VAR>
| epsilon
<VALOR_ARREGLO> := new <TIPO>[];
| {<TIPO_VAR> <ARGUMENTO>};
<VALOR_ASIGNACION> := , <OTROS> <VALOR_ASIGNACION>
| = <TIPO_VAR> <EXPRESION> <VALOR_ASIGNACION>
| ; <VALOR_ASIGNACION>
| <VALOR1> <EXPRESION> <VALOR_ASIGNACION>
| epsilon
<OTROS> := <TIPO> cadena
| <TIPO_VAR>
<TIPO_VAR> := num
| cadena
| decimal
| false
| true
<ARGUMENTO> := , <TIPO_VAR> <ARGUMENTO>
| epsilon
<IMPRIMIR> := <GRAFICAR_V>
| Console.WriteLine(<LISTADO_IMPRIMIR>);
<LISTADO_IMPRIMIR> := <TIPO_VAR> <LISTADO>
<LISTADO> := + <TIPO_VAR> <LISTADO>
| epsilon
<GRAFICAR_V> := graficarVector(cadena, cadena);
<SWITCH> := switch(cadena){<ESTRUCTURA_SWITCH>}
<ESTRUCTURA_SWITCH> := case <TIPO_VAR> : <INSTRUCCION> break; <EST_SWITP>
<EST_SWITP> := <ESTRUCTURA_SWITCH>
| default : <INSTRUCCION> break;
<EXPRESION> := <LISTA_ARIT> <M>
| <M>

| epsilon
<M> := <VALOR1>
| (<LISTA_VAR>)
<VALOR1> := cadena
| num
| decimal
<LISTA_ARIT> := /
| *
| +
| -
<IF> := if(<SENTENCIA>){<ARGUMENTO_IF>}<IFP>
<IFP> := else<INST_IF>
| epsilon
<INST_IF> := {<ARGUMENTO_IF>}
| <IF>
<SENTENCIA> := <VALOR1> <LISTA_OP> <VALOR1>
<LISTA_OP> := ==
| >
| < <N>
| !=
<ARGUMENTO_IF> := <INSTRUCCION>
<N> := =
| epsilon
<FOR> := for(<OP>; <OP1>; <INCREMENTO>){<INSTRUCCION>}
<OP> := <TIPO> cadena = <VALOR1>
<OP1> := cadena <LISTA_OP> <VALOR1>
<INCREMENTO> := cadena<INP>
<INP> := ++
| --

LEXEMAS Y SUS CORRELATIVOS

0 -> Ultimo token
1 -> class
2 -> numero
3 -> static
4 -> void
5 -> Main
6 -> (
7 -> args
8 ->)
9 -> {
10 -> }
11 -> int
12 -> float
13 -> bool
14 -> char
15 -> string o String

16 -> ,
17 -> .
18 -> ;
19 -> /
20 -> =
21 -> ==
22 -> >
23 -> <
24 -> !=
25 -> +
26 -> -
27 -> *
28 -> Console
29 -> Write
30 -> [
31 ->]
32 -> new
33 -> if
34 -> else
35 -> switch
36 -> case
37 -> break
38 -> default
39 -> :
40 -> for
41 -> <=
42 -> >=
43 -> while
44 -> true
45 -> false
46 -> cadena
47 -> graficarVector
48 -> decimal

ANEXOS

1. [LFP]PROYECTO2_201403541
    a. Form1
    b. Funcionalidad
    c. Léxico
    d. Sintáctico
    e. Lista
    f. Reporte

```
int numpre;
public int numerror;
Lista TokenActual;
List<Lista> listatokens;
List<Parser> ListaC = new List<Parser>();

public void Parsear(List<Lista> tokens)
{
    listatokens = tokens;
    Lista aux = new Lista(0, "Ultimo", 0, "Ultimo Valor", 0, 0);
    listatokens.Add(aux);

    TokenActual = tokens.ElementAt(0);
    numpre = 0;
    Inicio();
}
```

```
//                                                  Variables para crear reporte sintactico
int num, fila;
string tkn, lex, obtuvo;

public void Parea(int valor)
{
    if (!valor.Equals(TokenActual.Idtkn)) //Verifica que los valores coincidan
    {
        MessageBox.Show("Error se esperaba '" + TokenActual.Lexema + "' y se obtuvo token: '" + TokenActual.Lexema + "'" + "en la fila: " + TokenActual.Fila,"Error");

        numerror = 1;

        num = TokenActual.Numero;       fila = TokenActual.Fila;            tkn = TokenActual.Tkn;
        lex = TokenActual.Lexema;       obtuvo = TokenActual.Lexema;
        Parser aux = new Parser();
        aux.num = num;
        aux.fila = fila;
        aux.tkn = tkn;
        aux.lex = lex;
        aux.obtuvo = obtuvo;
        ListaC.Add(aux);
    }
    if (!TokenActual.Idtkn.Equals(0)) // Verifica si ya llego al tope de la lista.
    {
        numpre += 1;
        TokenActual = listatokens.ElementAt(numpre);
    }
    numerror = 0;
}
```