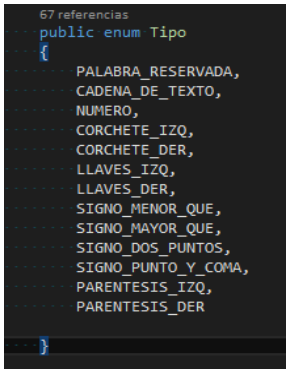
MANUAL TÉCNICO

Este manual es una explicación de la lógica utilizada para realizar dicha práctica, la principal herramienta utilizada fue el autómata finito determinista (que estaré adjuntando al final del archivo) ya que este se utilizó de base para llevar a cabo el analizador.

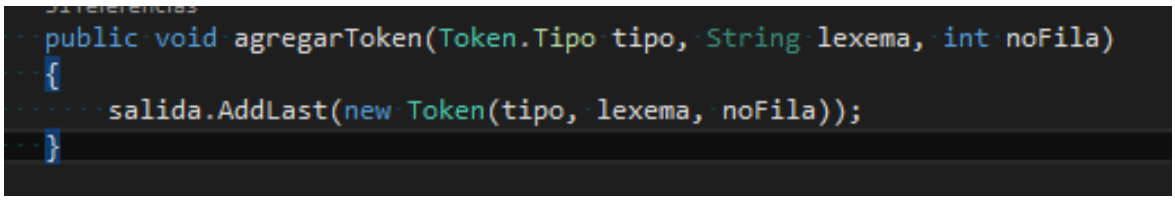
Luego de realizar el AFD, proseguimos a codificarlo; primeramente, eh creado 2 clases, una clase Token y otra Analizador_Lexico; en la clase Token, se utilizó un tipo de dato enumerado para asignar los diferentes tipos de tokens que se utilizarán en esta práctica de la siguiente forma:



Así mismo, hemos declarado 3 atributos de la clase, un atributo de tipo Tipo llamado tipoToken, uno tipo String llamado lexema y uno tipo int o entero llamado fila. En el constructor de esta clase se inicializan los atributos con las variables que este mismo recibe. Posteriormente se realizar los métodos para acceder a los atributos, siendo el mas extenso el atributo tipoToken en el cual con ayuda de una sentencia **switch** se obtiene el tipo de token al que pertenece el lexema reconocido:

El la clase Analizador_Lexico se declararon 4 atributos, uno de tipo LinkendList<Token> (que es una lista dinámica de la clase Token, que se utilizara para ir agregando los diferentes tipos de tokens encontrados) llamada salida, una variable tipo int o entero llamada estado, un tipo String llamada lexAux y una tipo int llamada fila que se inicializo en 1.

En esta clase se crearon 2 métodos, uno de ellos es agregarToken, que recibe tres parámetros un Token.Tipo tipo, un String lexema y un entero noFila, este método lo que hace es agregar el token a la lista de Tokens enviando los parámetros al constructor de la clase Token:

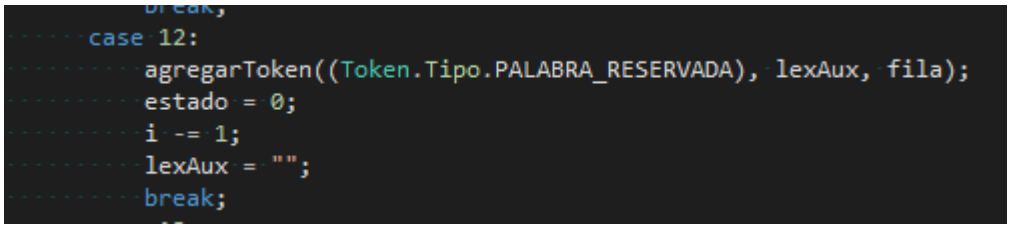


Y el otro método llamado escanear recibirá un parámetro de tipo String entrada, que será el texto que este en la pestaña actualmente seleccionada para así comenzar el análisis léxico; para esto eh utilizado una sentencia **switch** para manejar los estados del AFD, utilizando la variable estado que se inicializa en 0, la variable de lexAux, que se utiliza para ir almacenando los caracteres uno por uno y concatenándolos para así formar las palabras del lenguaje, una variable tipo Char llamada c para poder tomar carácter por carácter de la cadena y una lista de Tokens, el parámetro entrada se inicializa en una concatenación de si misma y el símbolo # esto para saber cuando se llega al fin de la cadena.

Para comenzar, se hace un for, que iniciara en 0 y terminara cuando sea menor al tamaño de la cadena de entrada e ira aumentando de uno en uno, luego dentro de nuestro for a la variable c, le asignamos el valor de la entrada con una propiedad para tomar el carácter según la posición que este en nuestro for, de esta forma ira aumentando el contador para ir cambiando el carácter que se toma de la entrada.

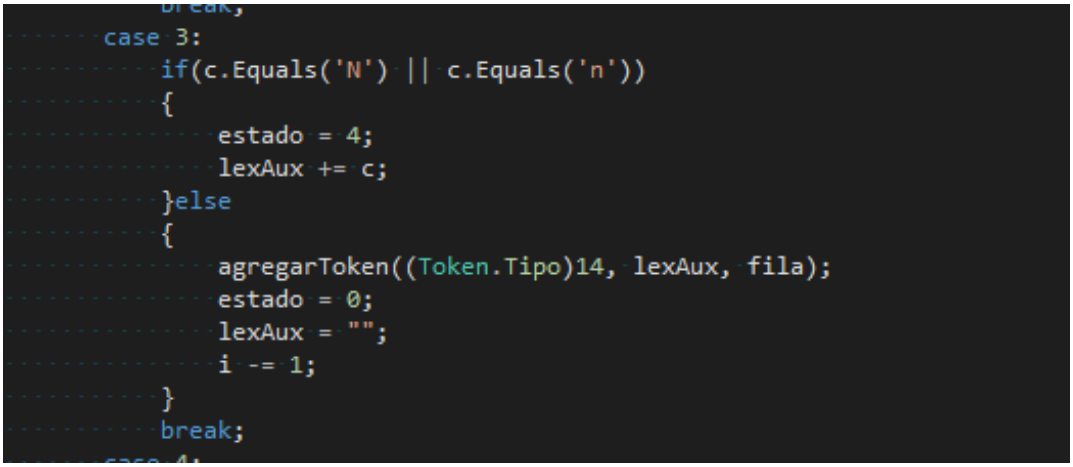
Si el estado es 0 entrara al case 0 de nuestro switch, en donde según el valor de la variable c, y viendo nuestro AFD, será comparado con cada uno de los símbolos posibles de transición, si entra a un estado de aceptación, se guardara el token, enviando el valor de la variable c, el tipo de token al que pertenece y la fila donde fue encontrada, si no es así a la variable lexAux

se le concatenara el valor de c, y se enviara al estado correspondiente según el AFD, y si el valor de c no coincide con ninguno de los símbolos de transición aceptados, comparamos si el valor de c es igual a # y si el contador del for es igual al tamaño de la entrada -1, si se cumple quiere decir que el análisis ya termino y se completo exitosamente. Si no se cumple se envía el error a la lista de tokens y el estado sigue siendo 0. Cuando un estado diferente del 0 es un estado de aceptación al agregar el token a la lista de tokens se debe limpiar la vaiable lexAux y la variable estado devolverla a 0, y la variable contador se deber reducir en uno para que se vuelva a considerar el carácter que se tomo cuando se llego a este estado:



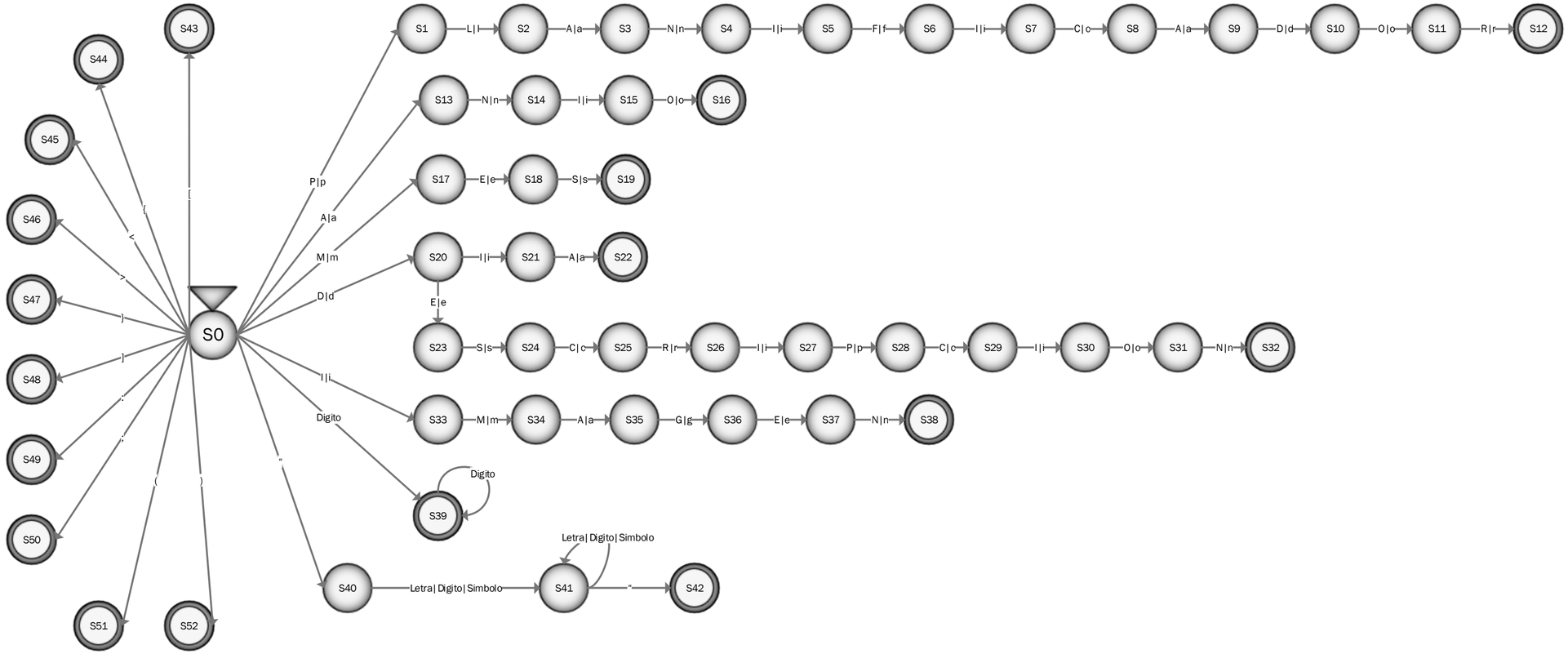
```
        break;
    case 12:
        agregarToken((Token.Tipo.PALABRA_RESERVADA), lexAux, fila);
        estado = 0;
        i -= 1;
        lexAux = "";
        break;
```

Del mismo modo cuando no coincide la variable c con el carácter que pertenece a determinado estado según el AFD, se debe agregar el token como desconocido y reiniciar todas las variables y el contador del for también debe regresar una posición:



```
        break;
    case 3:
        if(c.Equals('N') || c.Equals('n'))
        {
            estado = 4;
            lexAux += c;
        }else
        {
            agregarToken((Token.Tipo)14, lexAux, fila);
            estado = 0;
            lexAux = "";
            i -= 1;
        }
        break;
```

Por último si en el estado 0 c es un salto de línea la variable fila incrementara en 1 y permanecerá en el estado 0, y si c es un espacio en blanco permanecerá en el estado c.


S0
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
S34
S35
S36
S37
S38
S39
S40
S41
S42
S43
S44
S45
S46
S47
S48
S49
S50
S51
S52
L|l
A|a
N|n
I|i
F|f
I|i
C|c
A|a
D|d
O|o
R|r
P|p
A|a
M|m
D|d
E|e
S|s
C|c
R|r
I|i
P|p
C|c
I|i
O|o
N|n
I|i
M|m
A|a
G|g
E|e
N|n
Digito
Digito
Letra| Digito| Simbolo
Letra| Digito| Simbolo
Letra| Digito| Simbolo
"