Universidad Rafael Landívar

Facultad de Ingeniería

Lenguajes Formales y Autómatas, sec 01

Mgtr. Moises Alonso

Proyecto Práctico Fase: 2

José Alejandro Montenegro Monzón

1229918

Guatemala 13 de abril de 2020

**ÍNDICE**

1. **Entradas, Procesos, Salidas y PseudoCódigo…………………………… 3**
   1. **Entrada………………………………………………………………… 3**
   2. **Proceso………………………………………………………………… 3**
   3. **Salida………………………………………………………………….. 3**
   4. **Pseudocódigo………………………………………………………….. 3**
2. **Expresiones y Arboles de Expresiones…………………………………… 4**
3. **MANUAL DE USUARIO………………………………………………… 9**
4. **Entradas, Procesos, Salidas y PseudoCódigo**
5. **Entrada:** La única entrada de este proyecto es la lectura del archivo de texto. Si el archivo de texto no recibe un archivo de texto, sino recibe otra clase de archivo, éste no continuará la ejecución.
6. **Salida:** La salida será la validación de la estructura del archivo de texto. Además de las tablas de estados, la tabla de follow y la tabla de First, Last y Nullable.
7. **Proceso:**

* Separar por listas cada sección del Archivo (SETS, TOKENS, ACTIONS, ERROR)
* Crear los árboles con sus respectivas expresiones
* Recorrer cada frase que contendrá cada una de la lista y recorrer el árbol en In-Orden
* Recorrer cada Nodo y validar cada una de las frases en las listas.
* Si el resultado de cada
* Se verifica que el archivo posee elementos de la sección SETS
* Se crea la expresión regular, dependiendo de la cantidad de tokens recibidos en el archivo TOKENS.
* Se crea el árbol con la expresión creada y se empieza a recorrer, con el fin de crear la tabla First, Last y Nullable.
* Una vez creada dicha tabla, se vuelve a recorrer el árbol para crear la tabla follow.
* Para finalizar se realiza un tercer recorrido al árbol, para crear la tabla de estados.

1. Pseudocódigo del método main()

TAux.Remove(TAux[0]);

var Exp = flfn.ObtenerExpR(TAux, SAux);

if(Exp != string.Empty)

{

Expr.Text=(Exp);

//crear el arbol de expresiones First, Last, Nullable

var Tree = new Tree();

Tree.Raiz = nuevo.CreateTreeP2(Exp);

Ex = Tree.Raiz;

var contador = 1;

flfn.IngresarFLH(Tree.Raiz, ref contador);

flfn.RecorrerFLN(Tree.Raiz);

var diccionario = new Dictionary<int, string>();

for (int x = 1; x < contador; x++)

{

diccionario.Add(x, string.Empty);

}

//generar tabla follow

diccionario = flfn.TablaFollow(Tree.Raiz, diccionario, ref contador);

//generar tabla de S (tabla de estados)

var ListaSimbolos = new List<string>();

ListaSimbolos = flfn.ObtenerSímbolos(Tree.Raiz, ListaSimbolos);

ListaSimbolos.Remove(ListaSimbolos.Last());

var dic = new Dictionary<string, string[]>();

dic.Add(Tree.Raiz.elemento.First, null);

dic = flfn.Transición(Tree.Raiz, dic, ListaSimbolos, diccionario);

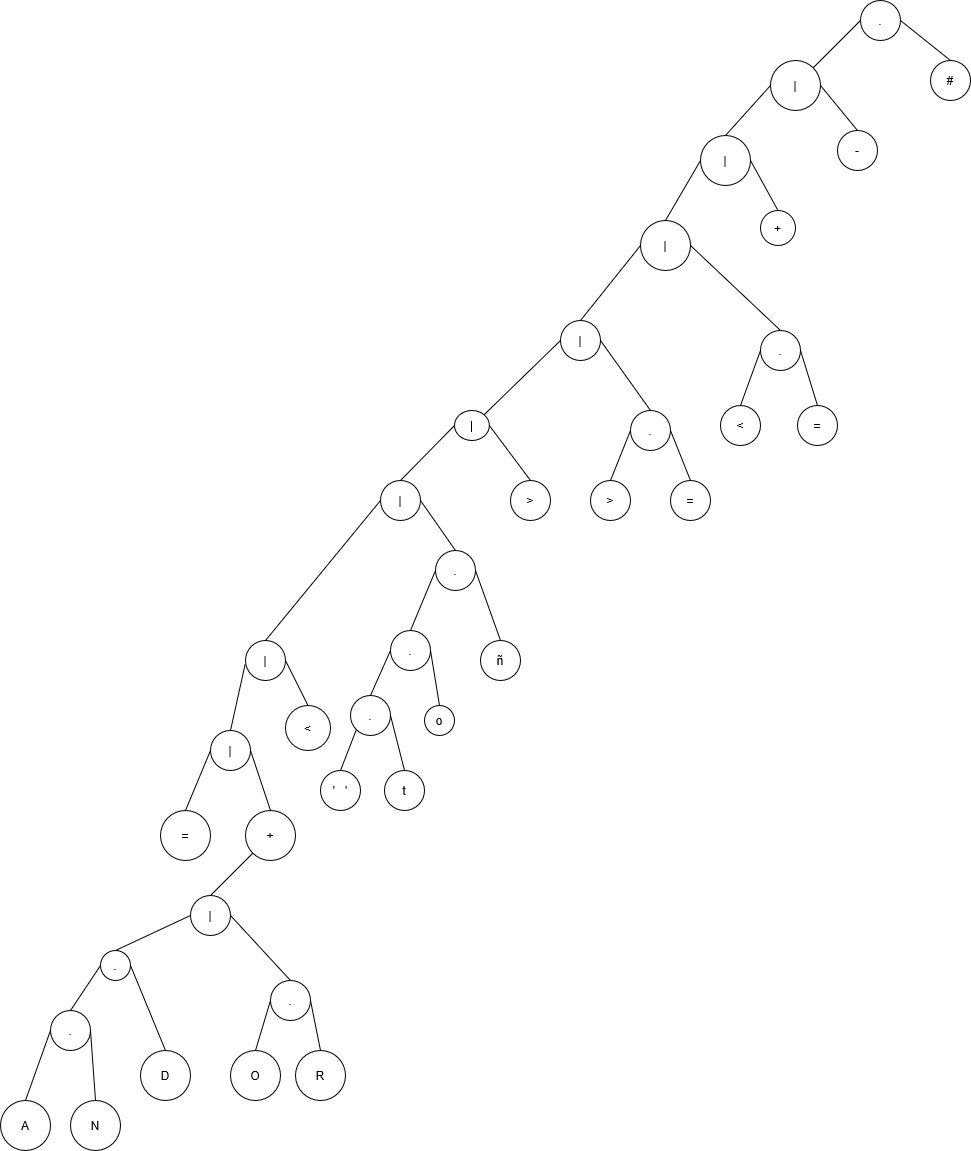
//Mostrar Tabla FLN

var matrix = new List<string[]>();

flfn.MostrarFLN(Tree.Raiz, ref matrix);

1. **Expresiones y Arboles de Expresiones**
   1. Prueba 4

(('='.('A'.'N'.'D'|'O'.'R')+|'<'|' '.'t'.'o'.'ñ'|'>'|'>'.'='|'<'.'='|'+'|'-').#)



**Tabla First, Last, NULLABLE**

|  |  |  |  |
| --- | --- | --- | --- |
| Símbolo | First | Last | Nullable |
| = | 1 | 1 | False |
| A | 2 | 2 | False |
| N | 3 | 3 | False |
| . | 2 | 3 | False |
| D | 4 | 4 | False |
| . | 2 | 4 | False |
| O | 5 | 5 | False |
| R | 6 | 6 | False |
| . | 5 | 6 | False |
| | | 2,5 | 4,6 | False |
| + | 2,5 | 4,6 | False |
| . | 1 | 4,6 | False |
| < | 7 | 7 | False |
| | | 1,7 | 4,6,7 | False |
| ‘ ‘ | 8 | 8 | False |
| \t | 9 | 9 | False |
| . | 8 | 9 | False |
| o | 10 | 10 | False |
| . | 8 | 10 | False |
| ñ | 11 | 11 | False |
| . | 8 | 11 | False |
| | | 1,7,8 | 4,6,7,11 | False |
| > | 12 | 12 | False |
| | | 1,7,8,12 | 4,6,7,11,12 | False |
| > | 13 | 13 | False |
| = | 14 | 14 | False |
| . | 13 | 14 | False |
| | | 1,7,8,12,13 | 4,6,7,11,12,14 | False |
| < | 15 | 15 | False |
| = | 16 | 16 | False |
| . | 15 | 16 | False |
| | | 1,7,8,12,13,15 | 4,6,7,11,12,14,16 | False |
| + | 17 | 17 | False |
| | | 1,7,8,12,13,15,17 | 4,6,7,11,12,14,16,17 | False |
| - | 18 | 18 | False |
| | | 1,7,8,12,13,15,17,18 | 4,6,7,11,12,14,16,17,18 | False |
| # | 19 | 19 | False |
| . | 1,7,8,12,13,15,17,18 | 19 | False |

**Tabla Follow**

|  |  |
| --- | --- |
| **Símbolo** | **Follow** |
| 1 | 2,5 |
| 2 | 3 |
| 3 | 4 |
| 4 | 2,5,19 |
| 5 | 6 |
| 6 | 2,5,19 |
| 7 | 19 |
| 8 | 9 |
| 9 | 10 |
| 10 | 11 |
| 11 | 19 |
| 12 | 19 |
| 13 | 14 |
| 14 | 19 |
| 15 | 16 |
| 16 | 19 |
| 17 | 19 |
| 18 | 19 |
| 19 |  |

**Tabla de Estados**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Estado | = | A | N | D | O | R | < | ‘‘ | \t | o | ñ | > | + | - |
| 1,7,8,12,13,15,17,18 | 2,5 |  |  |  |  |  | 16,19 | 9 |  |  |  | 14,19 | 19 | 19 |
| 2,5 |  | 3 |  |  | 6 |  |  |  |  |  |  |  |  |  |
| 16,19 | 19 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 9 |  |  |  |  |  |  |  |  | 10 |  |  |  |  |  |
| 14,19 | 19 |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 19 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 3 |  |  | 4 |  |  |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  | 2,5,19 |  |  |  |  |  |  |  |  |
| 10 |  |  |  |  |  |  |  |  |  | 11 |  |  |  |  |
| 4 |  |  |  | 2,5,19 |  |  |  |  |  |  |  |  |  |  |
| 2,5,19 |  | 3 |  |  | 6 |  |  |  |  |  |  |  |  |  |
| 11 |  |  |  |  |  |  |  |  |  |  | 19 |  |  |  |

* 1. Prueba\_2.1

(('='|'<'|' '.'t'.'o'.'ñ'|'>'|'>'.'='|'<'.'='|'+'|'-').#)

**Tabla First, Last, NULLABLE**

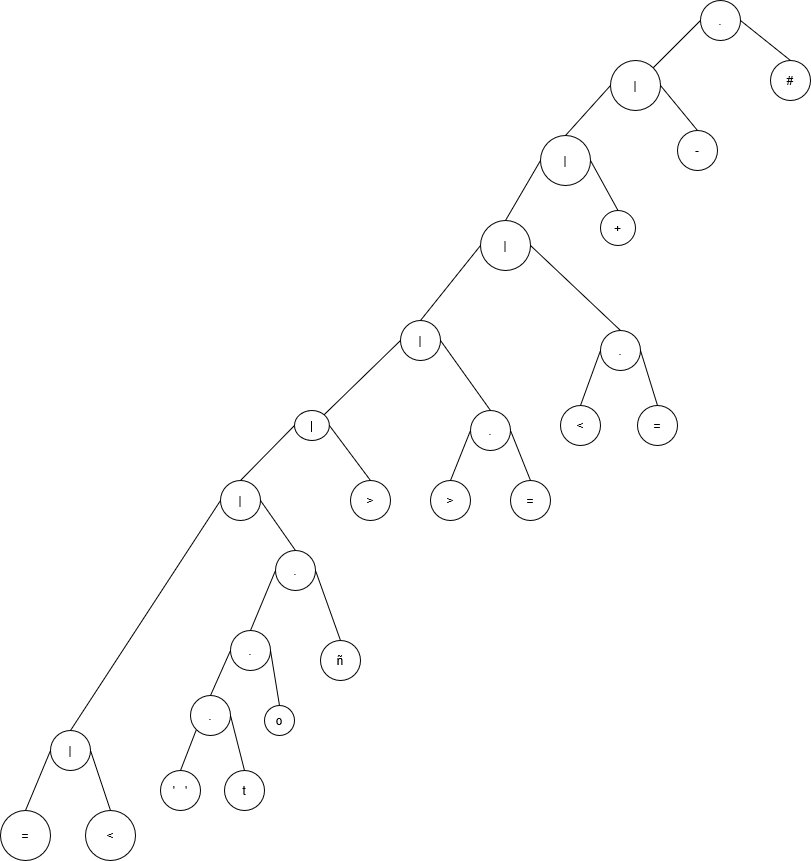
|  |  |  |  |
| --- | --- | --- | --- |
| Símbolo | First | Last | Nullable |
| = | 1 | 1 | False |
| < | 2 | 2 | False |
| | | 1,2 | 1,2 | False |
| ‘ ’ | 3 | 3 | False |
| t | 4 | 4 | False |
| . | 3 | 4 | False |
| o | 5 | 5 | False |
| . | 3 | 5 | False |
| ñ | 6 | 6 | False |
| . | 3 | 6 | False |
| | | 1,2,3 | 1,2,6 | False |
| > | 7 | 7 | False |
| | | 1,2,3,7 | 1,2,6,7 | False |
| > | 8 | 8 | False |
| = | 9 | 9 | False |
| . | 8 | 9 | False |
| | | 1,2,3,7,8 | 1,2,6,7,9 | False |
| < | 10 | 10 | False |
| = | 11 | 11 | False |
| . | 10 | 11 | False |
| | | 1,2,3,7,8,10 | 1,2,6,7,9,11, | False |
| + | 12 | 12 | False |
| | | 1,2,3,7,8,10,12 | 1,2,6,7,9,11,12 | False |
| - | 13 | 13 | False |
| | | 1,2,3,7,8,10,12,13 | 1,2,6,7,9,11,12,13 | False |
| # | 14 | 14 | False |
| . | 1,2,3,7,8,10,12,13 | 14 | False |

**Tabla Follow**

|  |  |
| --- | --- |
| **Símbolo** | **Follows** |
| 1 | 14 |
| 2 | 14 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |
| 6 | 14 |
| 7 | 14 |
| 8 | 9 |
| 9 | 14 |
| 10 | 11 |
| 11 | 14 |
| 12 | 14 |
| 13 | 14 |
| 14 |  |

**Tabla de Estados**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Estado | = | < | ‘\t‘ | t | o | ñ | > | + | - |
| 1,2,3,7,8,10,12,13 | 14 | 11,14 | 4 |  |  |  | 9,14 | 14 | 14 |
| 14 |  |  |  |  |  |  |  |  |  |
| 2,14 | 14 |  |  |  |  |  |  |  |  |
| 4 |  |  |  | 5 |  |  |  |  |  |
| 9,14 | 14 |  |  |  | 6 |  |  |  |  |
| 5 |  |  |  |  |  |  |  |  |  |
| 6 |  |  |  |  |  | 14 |  |  |  |



* 1. Prueba 1

El archivo de prueba uno, no posee correctamente la estructura en Tokens, por lo que no aceptará dicho archivo.

* 1. Prueba 2

El archivo de prueba dos no posee en SETS la palabra Prueba, por lo que no podrá aceptar dicha expresión. La palabra Lectura no se encuentra en SETS.

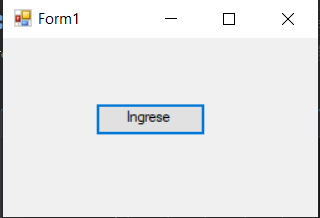
* 1. Prueba 3

Al igual que el archivo de prueba 2, el archivo de prueba 3 posee la palabra LECTURA, la cual no está escrita en SETS, debido a que SETS, ni siquiera está disponible.

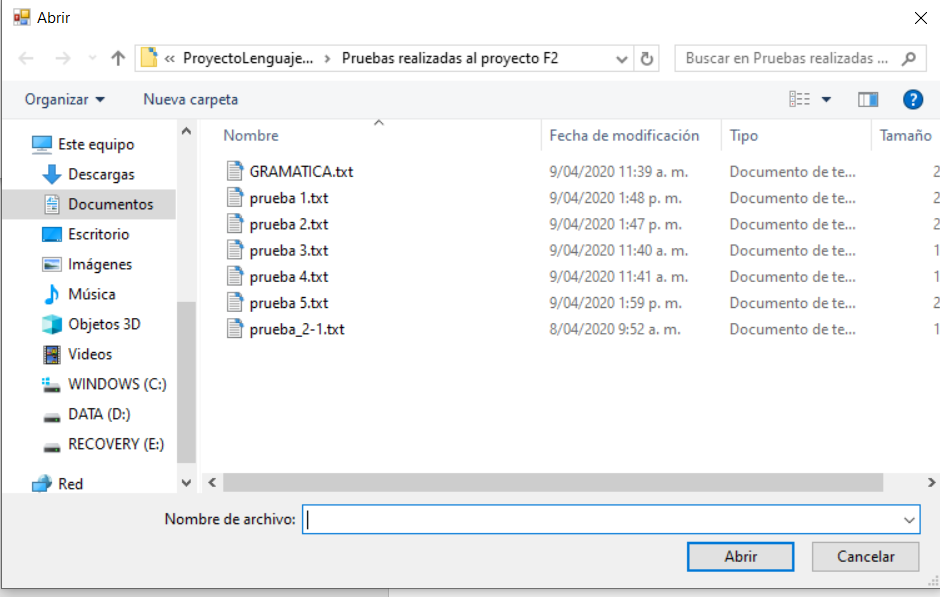
* 1. Prueba 5

El archivo de prueba, no posee la palabra CHARSETS, en SETS, por lo que no se aceptará dicho archivo tampoco.

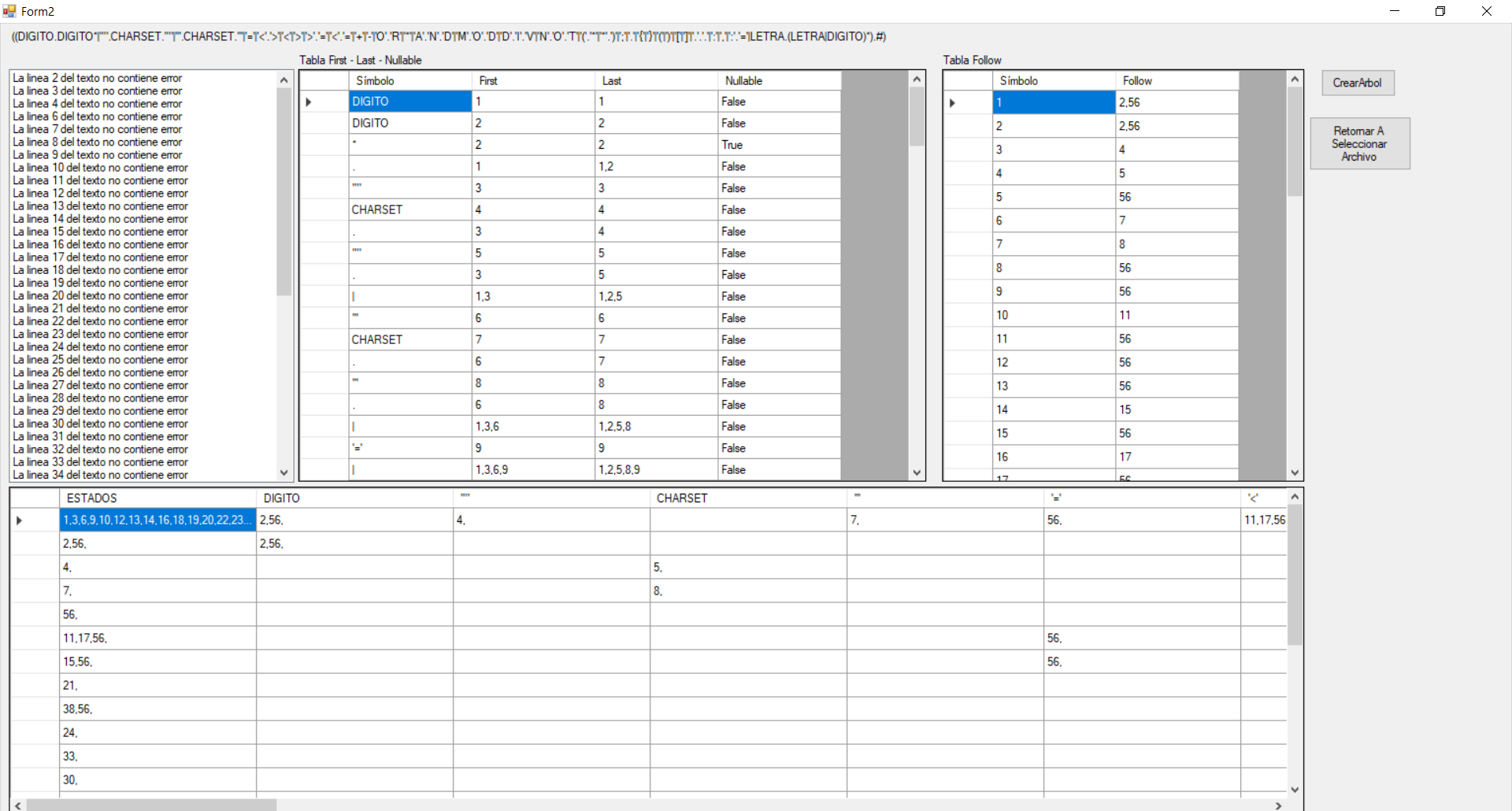
1. **Manual de Usuario**



Al iniciar la aplicación, debe de presionar el botón “Ingrese”, para poder seleccionar un archivo de texto.



Después el usuario puede seleccionar el archivo de texto que desee ingresar. En caso de que el usuario no seleccione un archivo de texto, la aplicación no le permitirá continuar.



Tras haber seleccionado un archivo de texto valido, en el programa aparecerán 4 tablas, Las primeras tres de arriba, en el orden de izquierda a derecha son:

1. **Tabla de validación de archivo**

En esta tabla se mostrará la línea que posea un error en el archivo, en caso el archivo no indique ningún error, ésta mostrará que dicha línea no poseerá ningún error y al finalizar la validación del archivo, se mostrará al final de la línea, El archivo no posee ningún error.

1. **Tabla First-Last-Nullable**

En la parte superior del Form, se mostrará la expresión regular creada. En la tabla First-Last-Nullable, se mostrará cada nodo del árbol creado en base a la expresión. Los nodos contendrán su estado First, Last y si es Nullable.

1. **Tabla Follow**

Una vez se obtiene la primera tabla, se vuelve a recorrer el árbol con el fin de calcular el Follow de cada uno de los nodos que no sea hoja.

Para finalizar la tabla de abajo, se crea la tabla de estados. En ésta indicará a cuales estados se pueden llegar con dicho símbolo.