Universidad Rafael Landívar

Facultad de Ingeniería

Lenguajes Formales Y Autómatas, sección 01

Catedrático: Ing. Moisés Alonso

**Proyecto Práctico Fase 3**

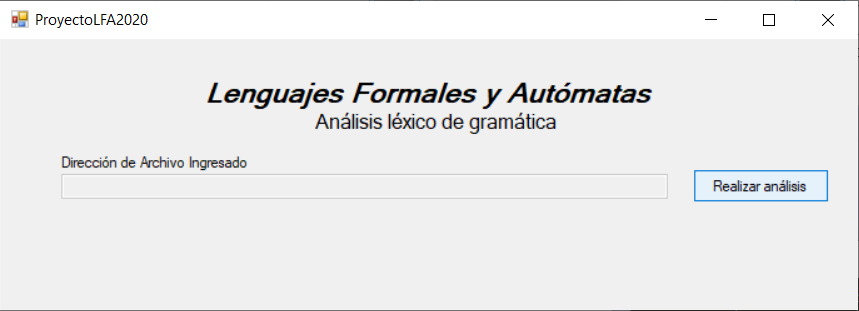
**“Generador de Código”**

José Eduardo Meléndez De la Rosa 1059918

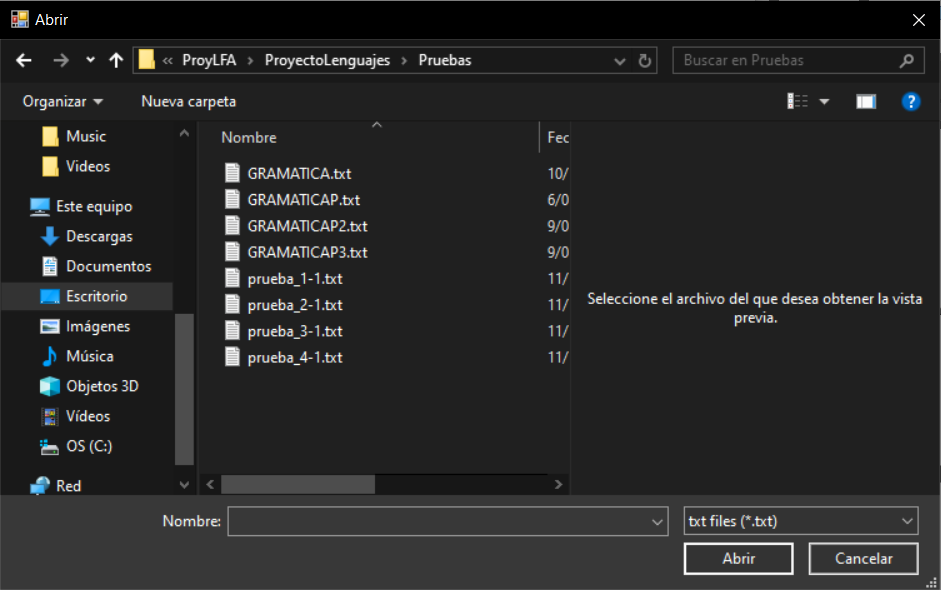
Guatemala, 9 de abril del 2020

# Manual de Usuario

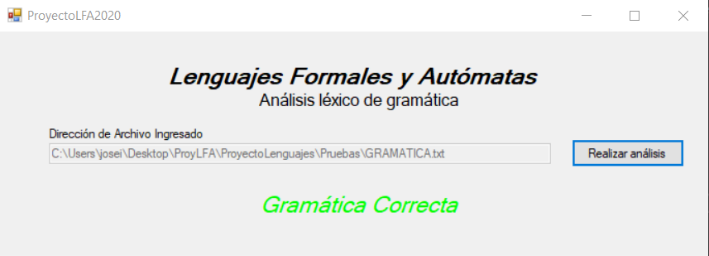
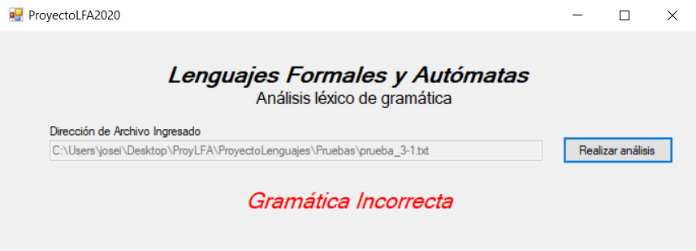
La primera ventana que se abre muestra un botón con el texto “Realizar Análisis”



Al hacer clic en el botón se abre una ventana de dialogo de apertura de archivo en el que solo se muestran archivos con extensión “.txt”, si se da el caso en que la carga del archivo es incorrecta, dicha ventana se abre de nuevo.



Se realizan las verificaciones correspondientes al análisis léxico de la gramática y se muestra una de la ventana inicial cambia a uno de los siguientes estados según corresponda.



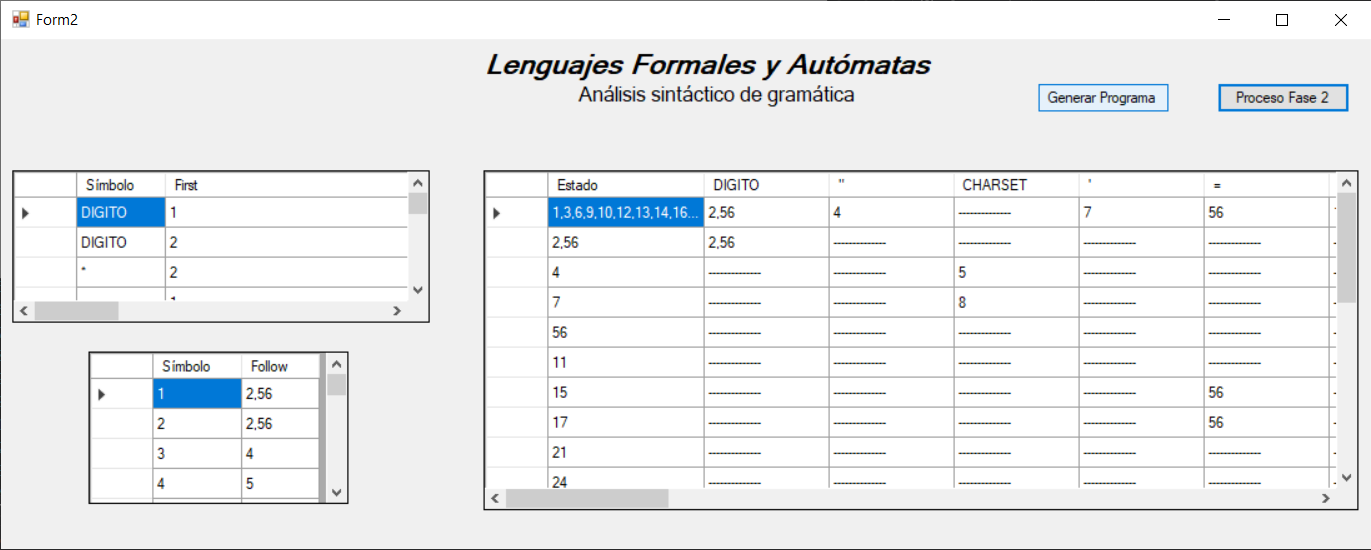
Si la gramática ingresada es correcta se abre la ventana que se muestra a continuación



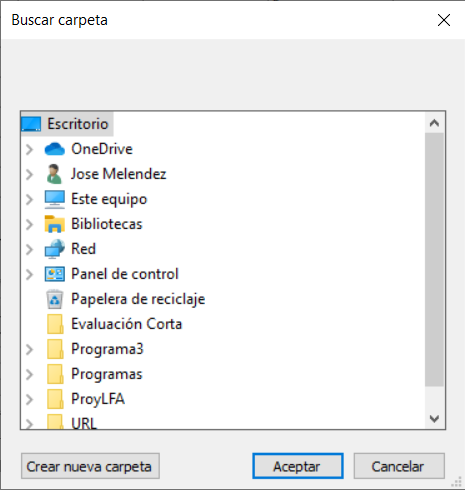
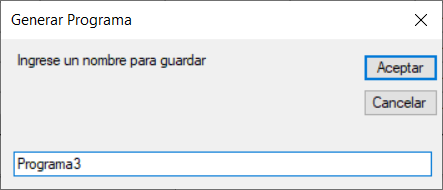
Se debe hacer clic sobre el botón con el texto “Proceso Fase 2”



Una vez realizados los cálculos las tablas se llenan con la información requerida por el análisis sintáctico de la gramática.



Después de generar el autómata el botón “generar” programa se habilita, al hacer click sobre dicho botón emergen una serie de ventanas que le permiten al usuario seleccionar una ubicación en la que se va generar el programa que permite evaluar si una cadena de texto es admitida o no dentro del leguaje definido con el archivo de gramática que se cargó previamente.

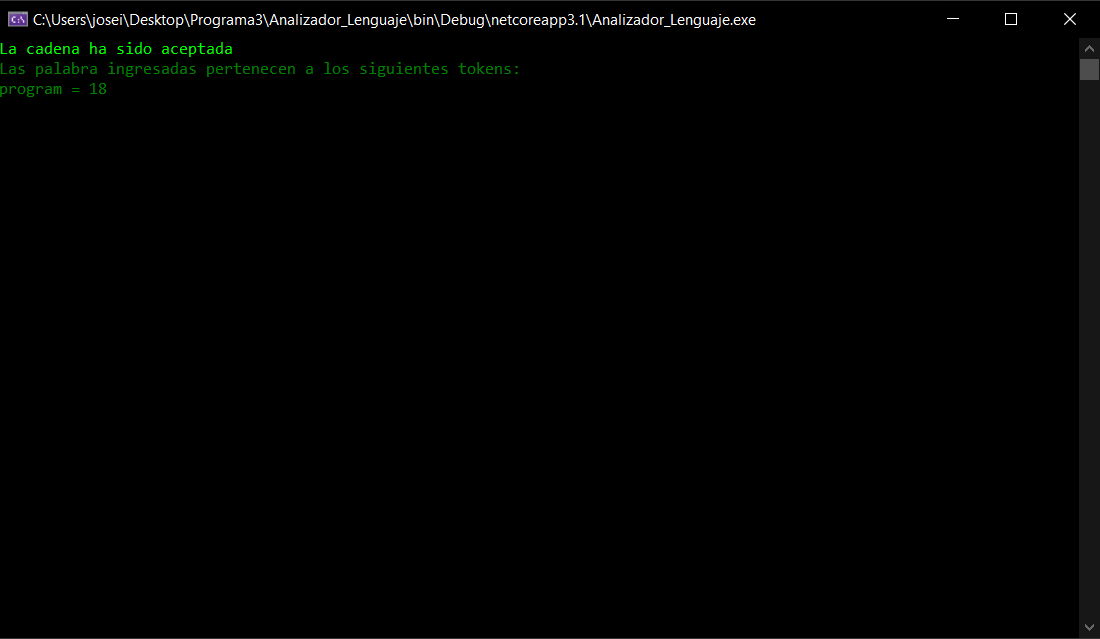
Lo siguiente que se debe hacer es dirigirse a la ubicación elegida y abrir la solución de visual studio “Analizador Lenguaje.sln”, dejar cargar por un momento y ejecutar la solución.

Una vez ejecutada la solución aparece la siguiente ventana, la cual da la opción de ingresar el texto que se desea analizar, en este paso de la solución el usuario debe ingresar el número “1” o “2” según lo desee.

Si se elige la opción 1 únicamente se debe arrastrar el archivo de texto .txt que contiene el texto a analizar sobre la consola de ejecución.

Si se elige la opción 2 el usuario debe escribir manualmente el texto que desea analizar.

**En ambos casos el usuario debe separar las cadenas por un espacio, se debe tomar en cuenta que el símbolo espacio no es reconocido por este analizador por lo que el símbolo espacio ayuda al analizador a separar las palabras para indicar lo que se considera como válido.**

****

Una vez analizada la cadena de entrada se muestra el siguiente texto en consola, indicando el token al que pertenece cada una de las palabras ingresadas.

# Pseudocódigo

## Definición de los valores First, Last de los nodos hoja

Entradas: Árbol de expresión, Cola de NodosHoja, Contador de Nodos

Algoritmo:

1. Si el nodo tiene hijo izquierdo
   1. Visitar hijo Izquierdo
   2. Ir a paso1.
2. Si el nodo tiene hijo derecho
   1. Visitar hijo derecho
   2. Ir a paso 1.
3. Si el nodo no tiene hizo izquierdo y no tiene hijo derecho
   1. Igualar los valores “first” y “Last” del nodo al valor del contador
   2. Agregar el Nodo a la Cola de Nodos
   3. Incrementar el valor del contador en 1.

## Calcular valores First, Last Y Nullable del resto de Nodos del árbol

Entradas: Árbol de expresión, Cola de NodosArbol

Algoritmo:

1. Si el nodo tiene hijo izquierdo
   1. Visitar hijo Izquierdo
   2. Ir a paso1.
2. Si el nodo tiene hijo derecho
   1. Visitar hijo derecho
   2. Ir a paso 1.
3. Si el carácter del nodo es “+”
   1. Igualar los valores first y last a los valores first y last del hijo izquierdo del nodo
   2. Si el hijo es anulable, hacer anulable el nodo
4. Si el carácter del nodo es “\*”
   1. Igualar los valores first y last a los valores first y last del hijo izquierdo del nodo
   2. hacer anulable el nodo.
5. Si el carácter del nodo es “?”
   1. Igualar los valores first y last a los valores first y last del hijo izquierdo del nodo
   2. hacer anulable el nodo.
6. Si el carácter del nodo es “.”
   1. Si el hijo izquierdo es anulable
      1. Igualar First del nodo a la unión del first de los hijos izquierdo y derecho
      2. Si no Igualar first al first del hijo izquierdo.
   2. Si el hijo derecho es anulable
      1. Igualar last del nodo a la unión del last de los hijos izquierdo y derecho
      2. Si no, Igualar last al last del hijo derecho.
   3. Si el hijo izquierdo y derecho son anulables
      1. Hacer anulable el nodo
7. Si el carácter del nodo es “|”
   1. Igualar First del nodo a la unión del first de los hijos izquierdo y derecho
   2. Igualar Last del nodo a la unión del last de los hijos izquierdo y derecho
   3. Si el hijo derecho o el hijo izquierdo es anulable
      1. Hacer anulable el nodo
8. Inicializar nodo auxiliar e igualar los valores first, last y nullable a los del nodo actual.
9. Encolar el nodo en la cola de NodosArbol

## Calcular Follow de los nodos hoja

Entradas: Árbol de expresión, Cola de NodoHoja

Algoritmo:

1. Si el nodo tiene hijo izquierdo
   1. Visitar hijo Izquierdo
   2. Ir a paso1.
2. Si el nodo tiene hijo derecho
   1. Visitar hijo derecho
   2. Ir a paso 1.
3. Si el carácter del nodo es “+” o “\*”
   1. Para cada valor last del hijo izquierdo
   2. Ir al NodoHoja cuyos valores first y last sean iguales al valor que se está buscando
   3. Agregar todos los valores first del hijo izquierdo al follow del nodo actual.
4. Si el carácter del nodo es “.”
   1. Para cada valor last del hijo izquierdo
   2. Ir al NodoHoja cuyos valores first y last sean iguales al valor que se está buscando
   3. Agregar todos los valores first del hijo derecho al follow del nodo actual.

## Calcular las transiciones

Entrada: Lista de enteros con el estado inicial, cola de nodos hoja, lista de símbolos usados, Cola estados nuevos, Cola de estados visitados, Diccionario de Estados Analizados

\*\*La estructura del diccionario de estados utilizado es

|  |  |  |
| --- | --- | --- |
| Diccionario< List<List<int>>,Diccionario<string, List<int>>> | | |
| List<List<int>> | Diccionario<string, List<int>> | |
| string | List<int> |

Algoritmo

1. Encolar estado inicial en cola estados nuevos
2. Mientras la cantidad de estados nuevos sea mayor a cero
   1. Inicializar nuevo diccionario Diccionario<string, List<int>> y agregar todos los símbolos utilizados junto con una lista de enteros vacía y nombrarlo transiciones
   2. Desencolar el primer estado de la cola de estados nuevos y agregar junto con el diccionario de transiciones al diccionario de estados.
   3. Para cada valor en la lista de enteros del estado inicial visitar, obtener valor del nodo hoja.
   4. Ir al nodo hoja cuyo first y last sea igual al valor que se está comparando y obtener su follosw y carácter.
   5. Acceder al diccionario de estados en el estado actual e ir al diccionario de transiciones cuya llave sea igual al carácter del nodo hoja, igualar la lista de enteros al follow del nodo hoja.
   6. Si el follow del nodo hoja no existe en la cola de estados visitados y tampoco en la cola de estados nuevos, encolar el follow en la lista de estados nuevos.

## Simplificar Autómata

Entrada: Diccionario de estados analizados, lista de estados visitados, lista de símbolos usados, diccionario de nombres de estados simplificados, diccionario de estados simplificados

Algoritmo:

1. Iniciar un contador con valor 0.
2. Para cada estado en la lista de estados visitados, añadir un valor al diccionario de nombres de estados, con el contador como llave, y el estado visitado como valor.
3. Crear un diccionario con los símbolos utilizados.
4. Para cada llave del diccionario estados analizados añadir la llave en la que el valor de diccionario nombres de estado es igual, y como valor el diccionario de transiciones del diccionario de estados analizados, cambiando el estado por su nombre.

## Generar código

Entradas: Cola de líneas, diccionario de estados simplificados

Algoritmo:

1. Encolar código para la creación de una variable Estado = 0;
2. Encolar código para el uso de un switch.
3. Para cada llave del diccionario de estados simplificados
   1. Encolar código para el caso de que la variable estado sea igual a la llave actual
   2. Para cada valor dentro del diccionario de transiciones de la llave actual.
   3. Si el valor del símbolo actual no es nulo
      1. Encolar condición para igual la variable estado al valor en esta posición del diccionario.
   4. Encolar cierre para caso que se está evaluando.

## Evaluación de texto

Entradas: línea de texto, switch generado, lista de estados de aceptación

Algoritmo:

1. Separar la línea de texto por los espacios ingresados.
   1. Para cada palabra en la línea de texto
      1. Para cada carácter en la palabra
      2. Recorrer el switch generado
   2. Si al evaluar todos los caracteres de la palabra evaluada, la variable estado es igual a alguno de los valores en la lista de estados de aceptación la palabra actual es aceptada.