Manual Técnico

Clase Analizador

La clase analizador es la encargada de realizar el análisis del texto incluido en el editor, esta clase cuenta con los siguientes métodos:

- **public List<Token> entrada(String fichero)** recibe como parámetro un string que es el texto que se desea analizar, dentro de este se encuentra la funcionalidad de un switch controlar las transiciones de los estados.
- **public void AgregarToken(Token.Tipos tipo)** recibe como parámetro un tipo de token definido en el analizador, se encarga de agregar un token valido a una lista.
- **public List<Nodos> AgregarNodo(List<Token> entrada)** método encargado de realizar la estructuración de los nodos para realizar el grafico correspondiente
- **public int TomarCodigo(int actual,List<Token> tokenActual,int tamaño)** guardar el código de un bloque de la lista de token generada por el primer método
- **public string TomarNombre(int actual, List<Token> tokenActual, int tamaño)** guarda el nombre de un bloque de la lista de token generada por el primer método
- **public List<int> TomarSuperiores(int actual, List<Token> tokenActual,int tamaño)** método que se encarga de generar una y guardar en una lista de enteros todos los superiores de un trabajador.

Clase Nodos

Abstracción de objeto trabajador, contiene todos los atributos que posee un trabajador como, código, nombre y superiores

Clase Token

Clase contenedora de los diferentes tokens validos dentro de la aplicación, posee atributos como el lexema valido, el tipo de token que es, la columna y fila donde fueran localizados

- **public String getLexema()** método encargado de devolver el lexema valido guardado en el objeto
- **public int getFila()** método encargado de devolver la fila en donde se encontró un lexema especifico.
- **public int getColumna()** método que se encarga de devolver la posición del final de lexema.
- **public string getTipo()** método encargado de devolver el tipo de token que se quiere guardar.

Clase Archivos

Clase que se encargar de generar los archivos necesarios para la creación de la página HTML, contiene los siguientes métodos.

- **public String ArchivoDot(List<Nodos> entrada)** Método que se encarga de generar el archivo .dot necesario para generar el grafico del organigrama, este recibe como parámetro una lista de los trabajadores que identificaron durante el análisis.
- **public String EjecutarConsola(string direccion)** Método encargado de ejecutar la consola de Windows (cmd) para la creación de la imagen por medio de la herramienta Graphviz. Recibe como parámetro la dirección del archivo .dot
- **public void GenerarHtml(String entrada)** Método encargado de generar la página HTML correspondiente en donde se estará contenido el organigrama creado, recibe como parámetro la dirección de la imagen a usar.
- **public void GenerarTablaToken(List<Token> auxiliar)** método encargado de generar una página HTML en donde contiene una tabla de todos los tokens identificados durante el análisis.
- **public void GenerarTablaErrores(List<Token> auxiliar)** método encargado de generar una página HTML que contiene una tabla de los errores identificados durante el análisis.

Formulario Form1

Contiene la interfaz gráfica de la aplicación.

- **menuStrip1** contiene las herramientas de la cinta superior de la interfaz
- **explorador** OpenFileDialog utilizado para permitirle al usuario poder abrir un archivo desde cualquier dirección
- **Guardar** SaveFileDialog usado para que el usuario pueda guardar un archivo ya existente verificando que sea válido o un archivo nuevo.
- **Guardar Como** SaveFileDialog usado para guardar una archivo ya existen con otro nombre y nueva dirección
- **areaTra** RichTextBox que le permite al usuario realizar la edición del texto

Métodos:

- **public void Pintar(List<Token> entrada)** método encargado de pintar aquellos token validos dentro del editor de texto, recibe como parámetro la lista de tokens que se genero

Eventos:

- **private void analizarToolStripMenuItem_Click(object sender, EventArgs e)** evento que se encarga de correr el elemento explorador y así mismo extrae el contenido del elemento areaTra para poder analizarlo
- **private void abrirToolStripMenuItem_Click(object sender, EventArgs e)** Evento que se encarga de llamar al elemento explorador

- **private void guardarToolStripMenuItem_Click(object sender, EventArgs e)** Evento que se encarga de llamar al elemento Guardar
- **private void guardarComoToolStripMenuItem_Click(object sender, EventArgs e)** Evento que se encarga de llamar al elemento GuardarComo.
- **private void acercaDeToolStripMenuItem_Click(object sender, EventArgs e)** Evento que abre una venta, donde se muestran los datos del creador de la aplicación.
- **private void salirToolStripMenuItem_Click(object sender, EventArgs e)** Evento que se encarga de finalizar la aplicación.

Formulario Form2

Ventana que contiene los datos del creador, contenidas en un label

D = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

L = { [A-Z], [a-z] }