Universidad San Carlos de Guatemala

Lenguajes Formales y de Programación

Auxiliar Daniel González González

Manual Tecnico

David Enrique Lux Barrera

201931344

Quetzaltenango, 2020

Manual Técnico acerca del IDE

Introducción

El sistema elaborado con fines de análisis de Lexemas se enfocó principalmente en dar solución a problemas de palabras que no se adecuan a cierto lenguaje, principalmente se buscó una solución para determinar aquellas palabras o símbolos que fueran parte del lenguaje que el cliente solicito.

Actualmente la aplicación solo posee PDF de información, no posee ningún otro dato ingresado y todo se maneja con memoria dinámica.

La práctica consiste en analizar carácter por carácter, texto que sea ingresado, ya sea escrito o por medio de un archivo que se cargue, la organización de la información, así como imagen o descripción si es que posee.

Tecnología Utilizada para la realización del programa:

- Visual Studio 2019

Requerimientos de Usuario

Dado que la aplicación fue desarrollada en el lenguaje de C#, se necesita que el usuario cumpla con estos requerimientos:

1. Sistema Operativo Windows 7 o Superior de 32 bits o 64 bits.
2. Tener instalado .NET Framework 3.5.
3. Mínimo 100 MB libres en el Disco Duro.
4. Mínimo 1 GB de Memoria RAM.

Iniciación de la aplicación:

Para Iniciar la Aplicación:

- Ir al Escritorio
- Dar doble Click en el Programa “Proyecto1Consola.sln”.

Solución Inicial

Para la solventar las peticiones de la aplicación:

1. Se Crearán métodos necesarios para manejar los diferentes menús que tendrá la aplicación.
2. Se Creará un método para generar nuevas pestañas.
3. Se Creará un método para generar las vallas publicitarias.
4. Se Creará una Clase que contenga los métodos necesarios para el análisis léxico.
5. Una vez terminado todas las funcionalidades se procederá a realizar pruebas para comprobar que todo funcione bien, y de encontrarse algún error se procederá a corregirlo para que funcione en su totalidad la aplicación.

Descripción de la fundamentación de archivos en el programa:

1. **Clases:**
    a. **AnalizadorLexico:** Contiene la clase donde estarán los métodos del analizador Léxico.
    b. **AnalizadorSemantico**
    c. **Archivos:** Contiene las clases que permiten al menuStrip1 del Form principal (Form1) poder generar las acciones de abrir, cerrar, etc.
    d. **Compilador:** Contiene las clases para poder generar una compilación en donde reconoce o analiza las acciones echas en el RichTextBox1, para generar una compilación y analizar los métodos creados en las clases de Analizador léxico, Analizador semántico y LectorSintactico, para así mostrar errores o si está bien el lenguaje ingresado.
    e. **LectorSintactico:** En el lector sintáctico se almacenan las clases para poder asignar los errores que se generen a la hora de la compilación del código escrito en el cuadro de texto
    f. **Graphiz:** En el se contiene la estructura de la creación del árbol del código que se desarrolle en la caja de texto de la misma y esta se generar en formato de imagen el cual mostrar los pasos de la creación del analizador.

2. **Métodos:**
    a. **Form2:** Diseño gráfico del IDE
        i. **Form2_Load():** Es el que llama a todos los procesos que actuaran en primera instancia, desde el cambio de colores reconocidos por el analizador léxico y las opciones de la habilitación del richTextBox1.
        ii. **richTextBox1_TextChanged_1():** Pinta las acciones que se generen dentro del cuadro de texto reconociendo la sintaxis del código generado.
        iii. **richTextBox1_KeyUp():** Manejar sólo a nivel de forma y no permitir que otros controles para recibir eventos de teclado.
        iv. **ANALIZAR_COMPILAR():** Se genera las acciones del poder analizador el código y compararlo con las cadenas de las clases de analizadores y genera un proceso de aceptación o devuelve los errores que se an encontrado.
        v. **nuevoToolStripMenuItem_Click():** Genera un nuevo campo para poder trabajar(Limpia las entradas creadas en el richTextBox1)
        vi. **cerrarProyectoToolStripMenuItem_Click():**Cierra el proyecto de manera definitiva.
        vii. **crearToolStripMenuItem_Click():** Abre un archivo con extensión .gtE para poder trabajar en dicho archivo.
        viii. **menuStrip1_ItemClicked():** Es el encargado de mostrar los ítems de archivo, edición y ayuda dentro de la visualización del programa.
        ix. **button1_Click():** Es el encargado de enviar la información de lo que se ha ingresado en el cuadro de texto y llama al campo de analizar_compilador, el cual es el botón que aparece en el diseño del Form2

x. **informacionToolStripMenuItem_Click():** Muestra la información del creador del archivo.

xi. **guardarComoToolStripMenuItem_Click():** Es el encargado de que lo llame a la clase Archivo, para poder generar la acción de "guardar como", abriendo una pestaña nueva donde se indicara donde será guardado

xii. **guardarToolStripMenuItem_Click():**Es el encargado de que lo llame a la clase Archivo, para poder generar la acción de "guardar" las modificaciones de dicho archivo.

xiii. **salirToolStripMenuItem_Click():** Cierra el programa de manera definitiva

b. Form1

i. Form1_Load(): Es el encargado de llamar al método de Loading para mandarle instrucciones al ProgressBar en la pantalla de carga.

ii. loading(): Es el encargado de regular tanto la velocidad como la distancia que recórrela la carga del ProgressBar, es decir, crea la imagen de carga y al final del proceso indicado abre el form2.

iii. Running(): Se desarrolla para hacer que el Loading() pueda abrir el Form2

c. ErroresDeTokens

i. FormMostrarErrores_Load(): Iguala los errores generados en el richTextBox1 con los errores asignados en la clase del AnalizadorLexico, para asi mostrarlos en unas tablas los errores que se han cometido.

ii. List<string> ListaErrores: es el que almacena los errores en una lista.

d. ArbolForms

i. FileDotEngine : IDotEngine: muestra la creación del código en forma del árbol de archivos.