Universidad Rafael Landívar

Facultad de ingeniería

Lenguajes Formales y Autómatas

Ing. Moisés Antonio Alonso González

**Documentación**

Dulce María Fernanda García Diaz – 1244621

Angie Paola Schumann Canjura – 1201119

Guatemala 27 de febrero del año 2023

ENLACE AL REPOSITORIO DE GITHUB: <https://github.com/dulcemfgarcia/ProyectoLFA.git>

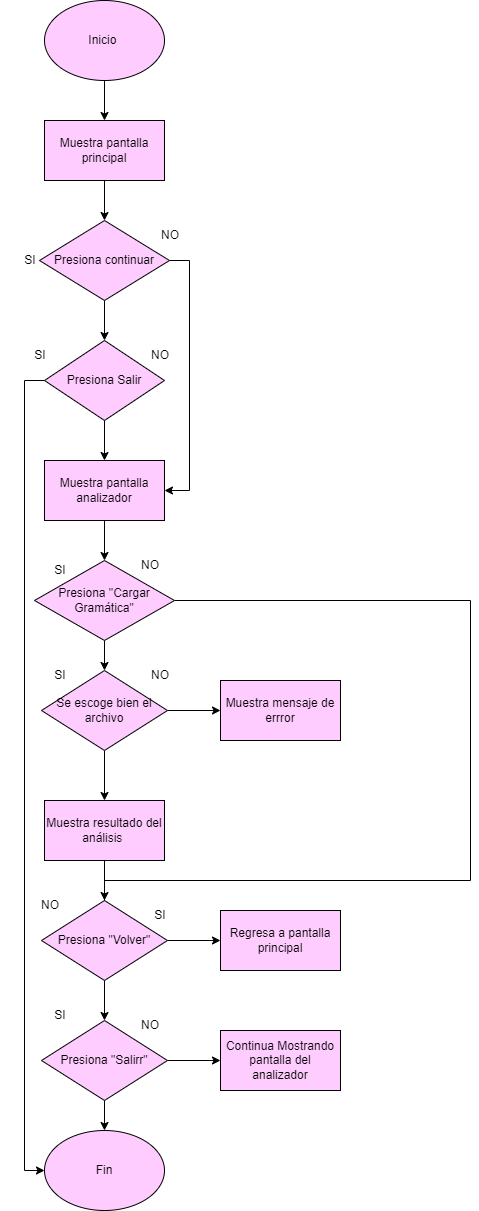
**DEFINICIONES**

* Cadena: Secuencia finita de símbolos sobre un alfabeto.
* Alfabeto: cualquier conjunto finito y no vacío de símbolos.
* Expresión Regular: Conjunto de palabras reconocidas por la computadora sobre un alfabeto específico.

**PANTALLAS**

|  |  |
| --- | --- |
| **Pantalla de Inicio** | **Pantalla del analizador** |
| Captura de pantalla de computadora  Descripción generada automáticamente | Captura de pantalla de computadora  Descripción generada automáticamente |

**FUNCIONALIDADES**



**EXPRESIONES REGULARES**

Por el momento solo creamos una clase funcional llamada “GrammarFormat”, en una carpeta llamada “Classes”. En esta clase definimos las expresiones regulares para cada caso o línea del archivo leído. Evaluamos el archivo por bloques identificando los requerimientos mínimos explicados en el proyecto.

Diagrama

Descripción generada automáticamente

Las expresiones regulares utilizadas fueron:

//Regular expression to evaluate SETS

private static string SETS = @"^(\s\*([A-Z])+\s\*=\s\*((((\'([A-Z]|[a-z]|[0-9]|\_)\'\.\.\'([A-Z]|[a-z]|[0-9]|\_)\')\+)\*(\'([A-z]|[a-z]|[0-9]|\_)\'\.+\'([A-z]|[a-z]|[0-9]|\_)\')\*(\'([A-z]|[a-z]|[0-9]|\_)\')+)|(CHR\(+([0-9])+\)+\.\.CHR\(+([0-9])+\)+)+)\s\*)";

//Regular expression to evaluate TOKENS

private static string TOKENS = @"^(\s\*TOKEN\s\*[0-9]+\s\*=\s\*(([A-Z]+)|((\'\*)([a-z]|[A-Z]|[1-9]|(\<|\>|\=|\+|\-|\\*|\(|\)|\{|\}|\[|\]|\.|\,|\:|\;))(\'))+|((\||\'|\\*|\?|\[|\]|\{|\}|\(|\)|\\)\*\s\*([A-Z]|[a-z]|[0-9]|\')\*\s\*(\||\'|\\*|\?|\[|\]|\{|\}|\(|\)|\\)\*\s\*([A-Z]|[a-z]|[0-9])\*\s\*\)\*\s\*(\||\'|\\*|\?|\[|\]|\{|\}|\(|\)|\\)\*\s\*\{\*\s\*([A-Z]|[a-z]|[0-9])\*\s\*(\||\'|\\*|\?|\[|\]|\{|\}|\(|\)|\\)\*\s\*(\||\'|\\*|\?|\[|\]|\{|\}|\(|\)|\\)\*\s\*)+)+)";

//Regular expression to evaluate ACTIONS and ERRORS

private static string ACTIONSANDERRORS

= @"^((\s\*RESERVADAS\s\*\(\s\*\)\s\*)+|{+\s\*|(\s\*[0-9]+\s\*=\s\*'([A-Z]|[a-z]|[0-9])+'\s\*)+|}+\s\*|(\s\*([A-Z]|[a-z]|[0-9])\s\*\(\s\*\)\s\*)+|{+\s\*|(\s\*[0-9]+\s\*=\s\*'([A-Z]|[a-z]|[0-9])+'\s\*|}+\s)\*(\s\*ERROR\s\*=\s\*[0-9]+\s\*))$";

Para utilizar las expresiones regulares hicimos uso de una librería llamada System.Text.RegularExpressions, con el fin de ingresar como parámetro el texto a analizar y la expresión regular establecida.

La utilizamos de la siguiente manera:

Captura de pantalla de un celular

Descripción generada automáticamente

Fuente: código propio

Regex.Match permite ingresar el texto y la expresión regular, y la función verifica si el texto concuerda con lo que la expresión regular requiere.

**REFERENCIAS**

Microsoft. (2022). *Regex Clase.* Recuperado de <https://learn.microsoft.com/es-es/dotnet/api/system.text.regularexpressions.regex?view=net-7.0>