Universidad Rafael Landívar
Facultad de Ingeniería.
Ingeniería en Informática y Sistemas.
Lenguajes Formales y Automatas - Sección: 01.
Catedrático: Ing. Moises Alonso.

DOCUMENTACIÓN

“GENERADOR DE SCANNER”

Estudiante: Herbert Emilio Alfaro Montes.
Carné: 1181320
Estudiante: César Daniel Bocel Morales.
Carné: 1094921

Guatemala, 3 de abril del 2024

EXPRESIONES REGULARES UTILIZADAS

SETS

```
public static bool TabOrEspaceOrEnter(string linea)
{
    return Regex.IsMatch(linea, @"^\s*$");
}
```

Esta expresión regular es utilizada para determinar si hay un tab, un espacio o un salto de línea en la línea del archivo, lo cual permite coincidir con cualquier carácter de espacio en blanco.

```
public static bool IniciaEnSets(string linea)
{
    return Regex.IsMatch(linea, @"^SETS\s*$");
}
```

Esta expresión regular es utilizada para determinar los SETS siguientes; es decir reconoce la palabra “SETS” e indica que inicia la parte de los SETS.

```
public static bool ContenidoValidoSets(string linea)
{
    return Regex.IsMatch(linea, @"^(\s*[A-Z]+ *=
*)(('\w+'((\+|\.{2})?'\w+')*)\s*|(CHR\(\d+\)((\+|\.{2})?CHR\(\d+\))*\s*))$");
}
```

Esta expresión regular es un conjunto de expresiones regulares, lo cual permite validar cualquier contenido valido para los SETS, donde permite coincidir la línea del archivo con espacios en blanco, dígitos, letras y CHR.

```
public static string GetContenedidoSet(string linea)
{
    Regex rg = new
Regex("((\\s*'\\w'\\s*((\\+|\\.{2})\\s*'\\w'\\s*)*)|(\\s*CHR\\(\\d+\\)\\s*((\\+|\\.{
2})\\s*CHR\\(\\d+\\)\\s*)*))");
    MatchCollection matchCollection = rg.Matches(linea);
    if (matchCollection.Count < 0)
        return string.Empty;
    else
        return matchCollection[0].Value;
}
```

Esta expresión regular obtiene el contenido obtenido en la sección “SETS”, retornando lo que se encuentre en la misma.

TOKENS

```
public static bool IniciaEnToken(string linea)
{
    return Regex.IsMatch(linea, @"^TOKENS\s*$");
}
```

Esta expresión regular es utilizada para determinar los TOKENS siguientes; es decir reconoce la palabra “TOKENS” e indica que inicia la parte de los TOKENS.

```
public static bool ContenidoValidoTokens(string linea)
{
    try
    {
        return Regex.IsMatch(linea, @"^(\s*(TOKEN\s*\d+\s*=\s*)(('.{1}')+|(\w+
(\|\|)?)+|('.{1}' \w+
*'.{1}'(\|\|)?)+|(\w*\s*(\(|{)(\s*\w*\s*(\(\))?(\|\|)?)+(\)|})(\|\|)?)+)\s*)$",
RegexOptions.IgnoreCase, timeout);
    }
    catch (RegexMatchTimeoutException e)
    {
        // Increase the timeout interval and retry.
        return false;
    }
}
```

Esta expresión regular es un conjunto de expresiones regulares, lo cual permite validar cualquier contenido valido para los TOKENS, donde permite coincidir la línea del archivo con espacios en blanco, dígitos y letras.

```
public static string TokenCorregido(string linea)
{
    Regex rg = new Regex(@"({)(\s*|\t*)(RESERVADAS)(\s*|\t*)(\()(\))(\s*|\t*)(})",
RegexOptions.Compiled | RegexOptions.IgnoreCase);
    MatchCollection matches = rg.Matches(linea);
    if (matches.Count > 0)
    {
        try
        {
            string nuevaLinea = Regex.Replace(linea,
@"({)(\s*|\t*)(RESERVADAS)(\s*|\t*)(\()(\))(\s*|\t*)(})", "", RegexOptions.None,
TimeSpan.FromSeconds(1.5));
            return nuevaLinea;
        }
        catch (RegexMatchTimeoutException)
        {
```

```
            return null;
        }
    }
    else
    {
        return linea;
    }
}
```

Esta expresión regular verifica los “TOKENS” con las expresiones regulares previamente mencionadas y corrige la parte afectada por una correcta utilización en la línea mencionada.

ACTION

```
public static bool IniciaEnAction(string linea)
{
    return Regex.IsMatch(linea, @"^ACTIONS\s*$");
}
```

Esta expresión regular indica el inicio de ACTION; es decir reconoce ACTIONS y todo lo siguiente con la palabra reservada ACTIONS.

```
public static bool ContenidoValidoActions(string linea)
{
    var matchs = Regex.Matches(linea, @"^\s*([A-Z0-
9_]+\(\))|({\s*)|(}\s*)|(\s*\d*\s*=\s*'[A-Z]*'\s*)|\s*$");
    if (matchs.Count < 0 && matchs == null)
        return false;

    return !string.IsNullOrEmpty(matchs[0].Value);
}
```

ERROR

```
public static bool IniciaEnError(string linea)
{
    return Regex.IsMatch(linea, @"^ERROR\w*\s* =\s* \d+$");
}
```

Esta expresión regular indica el inicio del ERROR; donde posteriormente se indicara en que parte del archivo se obtuvo un error para un formato no valido en el contenido del mismo.

```
public static bool ContenidoValidoError(string linea)
{
    return Regex.IsMatch(linea, @"^((\s*\w*ERROR\w*\s*=)(\s*\d*\s*)|\s*)$");
}
```

Esta expresión regular es utilizada para indicar el ERROR obtenido al momento de realizar la lectura del archivo, donde se le indica al usuario la línea exacta donde el formato no es válido para el mismo.

VIDEO DE EXPLICACIÓN

LINK DEL VIDEO