Universidad Rafael Landívar
Facultad de Ingeniería
Lenguajes formales y autómatas, Sección: 01
Catedrático: Ing. Moisés Alonso

Fase I
"Generador de scanner"

Roberto Solares (1173318)

Guatemala, 4 de Marzo del 2020

Objetivo

Dentro del análisis de los lenguajes Formales es importante conocer las fases del proceso de compilación, la finalidad del proyecto es que el estudiante comprenda la función de los analizadores léxico y sintáctico de un compilador a través de la generación de un programa que sea capaz de reconocer un lenguaje y finalmente evaluar si las palabras utilizadas están bien formadas de acuerdo con una gramática.

El proyecto consta de 3 fases, las cuales son dependientes, es decir, que para poder terminar la fase III, las anteriores deben estar completas, de lo contrario no se podrá entregar el funcionamiento completo.

Análisis

- **Entradas:** Un archivo de texto.
- **Salidas:** Mensaje informativo, que indica si está bien o mal.
- **Procesos:**
    - Leer el archivo.
    - Generar los árboles de expresión.
    - Evaluar el texto utilizando los árboles.
    - Mostrar algún mensaje al usuario.
- **Restricciones:** Ninguna.

Expresiones Regulares

SETS

```
((S·E·T·S)·((\t|\s)*·(\n))+·((\t|\s)*·[A-Z]+·(\t|\s)*·=·(\t|\s)*·((('·.·')|(C·H·R·\(·[0-9]+·\)))·((\.·\.)·(('·.·')|(C·H·R·\(·[0-9]+·\))))?·(\+·(('·.·')|(C·H·R·\(·[0-9]+·\)))·((\.·\.)·(('·.·')|(C·H·R·\(·[0-9]+·\))))?)*)·((\t|\s)*·(\n))+)+)?
```

TOKENS

```
(\n)*·(T·O·K·E·N·S)·((\t|\s)*·(\n))+·((\t|\s)*·(T·O·K·E·N)·(\t|\s)+·[0-9]+·(\t|\s)*·=·(\t|\s)*·(\s|\*|\+|\?|\(|\)|\||'·.·'|[A-Z]+|{|})+·((\t|\s)*·(\n))+)+
```

ACTIONS

```
(\n)*·(A·C·T·I·O·N·S)·((\t|\s)*·(\n))+·((\t|\s)*·(R·E·S·E·R·V·A·D·A·S·\(·\))·((\t|\s)*·(\n))+·{·((\t|\s)*·(\n))+·((\t|\s)*·[0-9]+·(\t|\s)*·=·(\t|\s)*·'·[A-Z]+·'·((\t|\s)*·(\n))+)+·}·((\t|\s)*·(\n))+)·((\t|\s)*·[A-Z]+·\(·\)·((\t|\s)*·(\n))+·{·((\t|\s)*·(\n))+·((\t|\s)*·[0-9]+·(\t|\s)*·=·(\t|\s)*·'·[A-Z]+·'·((\t|\s)*·(\n))+)+·}·((\t|\s)*·(\n))+)*
```

ERRORS

((\t|\s)*·(\n))+·([A-Z]+·(E·R·R·O·R)·(\t|\s)*·'·(\t|\s)*·=·(\t|\s)*·[0-9]·((\t|\s)*·(\n))+)+

Expresión final

^·(" + SETS + ")·(" + TOKENS + ")·(" + ACTIONS + ")·(" + ERRORS + ")·$

Diagramas de clases

Regex
public string Expression public Node Tree
private void CreateTree(string regex)

Node
public string Value public Node LeftChild public Node RightChild public Node Parent
public Node(string value)

Helpers
public bool CheckPrecedence(string token, string lastOperator, List<string> operators) public List<string> TokenizeExpression(string regex) private bool DigitInterval(string token) private bool UpperInterval(string token) private bool LowerInterval(string token)

Element
public char Value public int Column public int Row
public Element(char value, int column, int row)

Diagramas de flujo