Universidad Rafael Landívar

Facultad de Ingeniería

Lenguajes Formales y Autómatas

Sección 01

Ingeniero Moises Alonso

Proyecto Fase I

Generador de Scanner

Pablo Muralles 1113818

Guatemala 4 de Marzo del 2020

Objetivo General

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.

Objetivo Específico

- Aplicar los conocimientos aprendidos en clase hasta el momento para realizar esta fase.
- Generar correctamente el analizador léxico.
- Crear las expresiones regulares de manera adecuada para validar el archivo.

Análisis

- Entradas: Archivos de texto a evaluar

- Salidas: Mensaje donde especifica si esta correcto el archivo o sino esta correcto mostrar línea del error
- Procesos:
    1. Leer archivo que suba el usuario
    2. Generar los árboles de expresiones
    3. Evaluar el texto subido con los árboles de expresiones
    4. Retornar mensaje al usuario

Expresiones Regulares

A continuación, se especifican las expresiones regulares utilizadas para poder generar los arboles de expresiones

SETS:

([a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s])+=((('([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'|'([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'\.\.'([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})')(\+('([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'|'([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'\.\.'([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'))*)|((CHR\([0-9]+\)\.\.CHR\([0-9]+\))(\+(CHR\([0-9]+\)\.\.CHR\([0-9]+\)))*))

TOKENS:

((TOKEN([\s])+[0-9]+([\s])*=([\s])*((('([a-zA-Z0-9<>="";:(){}\.\[\],'\+\-_*\.])')+)))|(TOKEN([\s])+[0-9]+([\s])*=([\s])*((('([a-zA-Z0-9<>="";:(){}\.\[\],'\+\-_*\.])')+)([\s])*([a-zA-ZñÑ]+)([\s])*((('([a-zA-Z0-9<>="";:(){}\.\[\],'\+\-_*\.])')+))(([\s])*\|([\s])*((('([a-zA-Z0-9<>="";:(){}\.\[\],'\+\-_*\.])')+)([\s])*([a-zA-ZñÑ]+)([\s])*((('([a-zA-Z0-9<>="";:(){}\.\[\],'\+\-_*\.])')+)))|(TOKEN([\s])+[0-9]+([\s])*=([\s])*([a-zA-ZÑñ]+([\s])*(\+|*|\?|\|)?)+)|(TOKEN([\s])+[0-9]+([\s])*=([\s])*([a-zA-ZÑñ]+([\s])*(\+|*|\?)?)*(([\s])*((\(([\s])*[a-zA-ZÑñ]+([\s])*(\+|*|\?|\|)?)(([\s])*[a-zA-ZÑñ]+([\s])*(\+|*|\?|\|)?([\s])*)*\)(\+|*|\?|\|)?)*)?(([\s])*{([\s])*[a-zA-ZÑñ]+([\s])*\(([\s])*\)([\s])*})))))

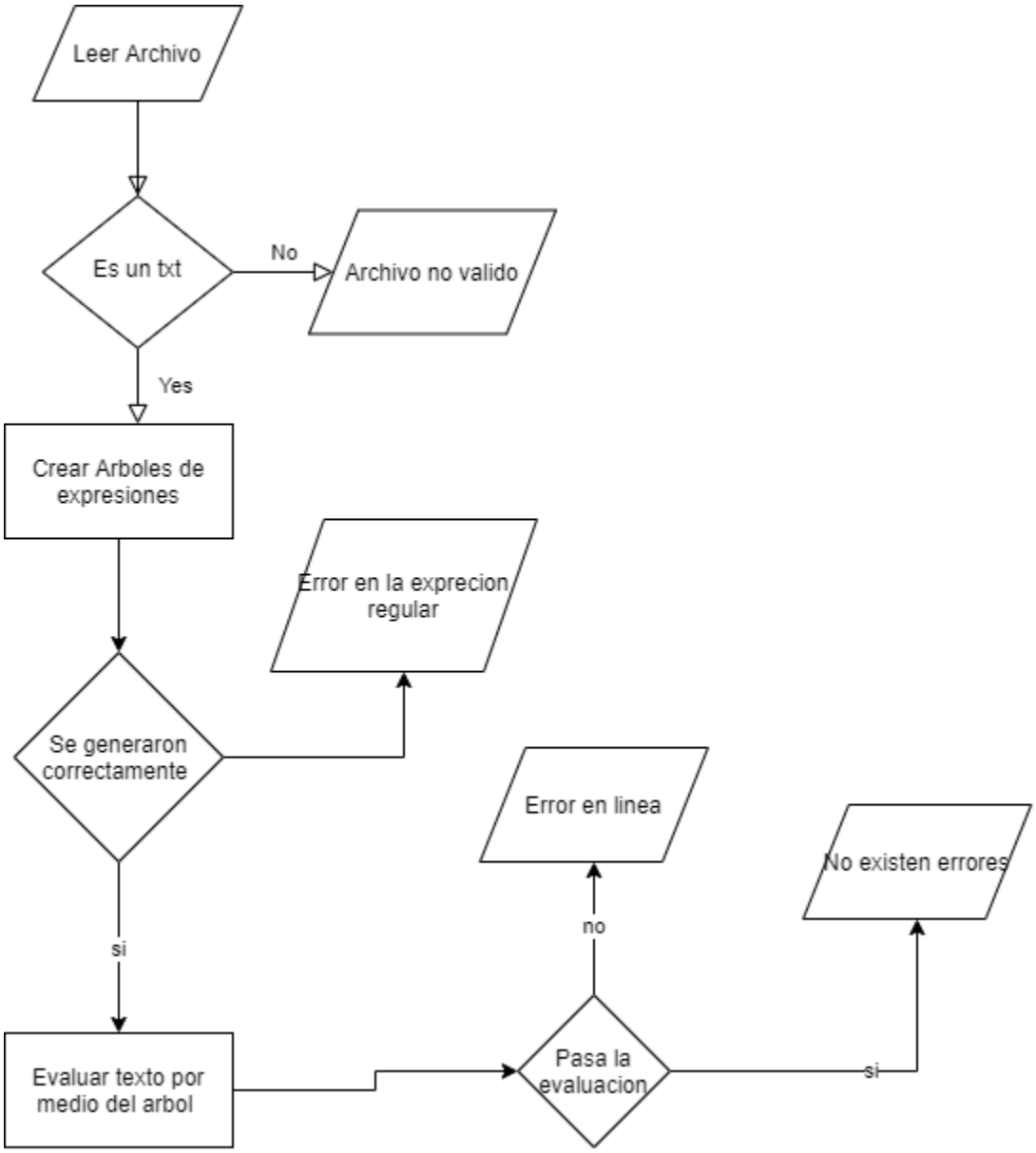
ACTION:

((([a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s])+=((('([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'|'([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'\.\.'([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})')(\+('([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'|'([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'\.\.'([0-9a-zA-ZñÑ<>="";:(){}\.\[\],'\+\-_*\.\s]{1})'))*)|((CHR\([0-9]+\)\.\.CHR\([0-9]+\))(\+(CHR\([0-9]+\)\.\.CHR\([0-9]+\)))*))**)|(** [0-9]+=('([a-zA-Z0-9<>="";:(){}\.\[\],'\+\-_*\.]+)')**))**

ERRORES:

([a-zA-zñÑ]+)*ERROR=[0-9]+

Diagrama de Flujo



Diagramas de Clases

Nodo
public string Padre public Nodo Izquierdo public Nodo Derecho public string Data
public Nodo(string data)

Crear arbol
string ExprecionRegularSets string ExprecionRegularActions string ExprecionRegularTokens string ExprecionRegularError List<string> SimbolosTerminales List<string> Operadores Queue<string> TokensExpresion Stack<Nodo> T Stack<Nodo> S
public ArbolExprecionesSets( ) public void ConvertirExprecionaTokens() public void Crear_st_op() public bool VerificarPrecedencia() public void RecorridoInorden()

Manipular Texto
private StreamReader TextoEvaluar private int contador private string Contenido
public ManipulacionTexto() public void BuscarInicios() public void Comparar()