MANUAL TECNICO DE ANALIZADOR LEXICO

by Juan Pablo Orizabal Gil
Carné: 201730318

MANUAL TÉCNICO

El software fue probado en Mozilla Firefox, pero es altamente probable que funcione a la perfeccion en otros interpretes tales como Google Chrome, o Safari. Sin embargo no nos hacemos responsables si no corre de forma correcta.

El software es un analizador lexico, esto quiere decir que podemos realizar distintas funciones, como analizar palabras dentro de un texto, reconocer patrones, así como exportarlo a un archivo de texto.

El software consta de tecnologia HTML y de Javascript para poder convertir el lenguaje etiquetado de HTML en un lenguaje dinamico.

Clase Codigo Javascript

Esta clase es la que contiene a todos los métodos y funciones que se utilizaron para cumplir con la logica que el analizador lexico requeria. Aqui esta desde la captura del archivo de texto, el analisis de los patrones, de los tokens, hasta la descarga del archivo de texto.

METODOS

introducirDatos(): Este método lo que hace es empezar el proceso para introducir datos al text area, llama al método procesar.

procesar(e): Este método se encarga de crear el objeto FileReader que nos servira para leer el archivo. El parametro e hace referencia al objeto que invoco al método es decir a la funcion que lo llamo.

mostrarInformacion(e): Este método introduce los datos al text area, ya tiene la informacion, unicamente es de introducirlos al text area.

numerar(): Este método se encarga de darle un numero a cada fila, es decir enumera según las filas que existan en el text area.

validarCadena(): Este método se encarga de validar la cadena, de aquí es donde parte todo el analisis de las cadenas que contenga el text area, se encarga de llamar a los métodos que son los que van validando carácter a carácter.

verificarTipo(cadena): este método se encarga de verificar de que tipo sera la cadena que se mando, esto quiere decir que devuelve el valor del primer carácter o a que tipo del alfabeto pertenece.

verificarIdentificador(numeroFila,cadena,numeroColumna): Verifica si es un token identificador valido

verificarNumero(numeroFila,cadena,numeroColuma): verifica si es un token numero valido.

verificarDecimal(numeroFila,cadena,numeroColumna): verifica si es un token decimal valido.

verificarTamañoCadena(numeroFila,cadena,numeroColumna): este método es un poco mas general ya que se encarga de verificar si es un token de signo de agrupacion, o un operador matematico o un signo de puntuacion.

cicloLetra(caracter): verifica si el carácter pertenece a ser una letra valida.

cicloDigitos(caracter): veirica si el carácter pertenece a ser un digito valido.

cicloSignosPuntuacion(caracter): verifica si el carácter pertenece a ser un signo de puntuacion valido.

cicloOperadoresAritmeticos(caracter): verifica si el carácter pertenece a ser un operador aritmetico valido.

cicloSignoAgrupacion(caracter): verifica si el carácter pertenece a ser un signo de agrupacion valido.

cicloEspacioSalto(caracter): verifica si es un salto de linea o un espacio.

mostrarTablaErrores(): Muestra la tabla de errores en una ventana nueva.

mostrarTablaTokens(): Muestra la tabla de tokens en una ventana nueva.

insertarFilaTablaErrores(filaError,palabra,columnaError): inserta una fila a la tabla errores.

insertarFilaTablaTokens(posicion,token,numeroColumna): inserta una fila a la tabla tokens.

recuentoDeLexemas(cadena): hace el conteo de cuantas veces se repite un lexema.

introducirDatosTablaLexema(): inserta datos a la tabla de recuento de lexemas.

insertarFilaTablaLexema(lexema,vecesRepetido): introduce los datos uno por uno a la tabla de lexemas.

mostrarTablaLexemas(): muestra la tabla de lexemas en una ventana nueva.

eliminarDatosLexema(): elimina los datos al arreglo que contiene a los lexemas y la cantidad de veces que se repite.

eliminarFilasTablaErrores(id): elimina los datos que puede contener una tabla según su id , esto para hacer cada analisis independiente.

buscarPatrones(): busca los patrones que se solicitan y los subraya.

validarString(): es el encargado de ver si las secuencias de caracteres son validas o no y lo retorna por medio de un booleano.

descargarArchivo(): es el encargado de descargar los datos que estan en el text area a un archivo de texto

Ademas estos son las constantes que se utilizaron para poder crear el alfabeto.

```
// son los caracteres que conforman el alfabeto
const LETRAS_MINUSCULAS=['a','b','c','d','e','f','g','h','i','j','k','l','m','n','o','p','q','r','s','t','u','v','w','x','y','z'];
const LETRAS_MAYUSCULAS=['A','B','C','D','E','F','G','H','I','J','K','L','M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z'];
const DIGITOS=[0,1,2,3,4,5,6,7,8,9];
const SIGNOS_PUNTUACION=['.',',',':',';'];
const OPERADORES_ARITMETICOS=['+','-','*','/','%'];
const SIGNOS_AGRUPACION=['(',')','[',']','{','}'];
const ESPACIO_SALTO_LINEA=['','\n'];
// son los nombres de los tokens
const IDENTIFICADOR="Identificador";//inicia con letra seguido de letras o digitos
const NUMERO="Numero";//solo digitos
const DECIMAL="Decimal";//n numeros seguidos de punto seguido de mas numeros
const PUNTUACION="Puntuacion";// solo los simbolos de puntuacion
const OPERADOR="Operador";// solo los simbolos de operador
const AGRUPACION="Agrupacion";// solo los simbolos de agrupacion
// son los distintos nombres que reciben los caracteres
const PALABRA_LETRA="letra";
const PALABRA_DIGITO="digito";
const PALABRA_SIGNO_PUNTUACION="signoPuntuacion";
const PALABRA_OPERADOR_ARITMETICO="operadorAritmetico";
const PALABRA_SIGNO_AGRUPACION="signoAgrupacion";
const PALABRA_ESPACIO_SALTO_LINEA="espacioSaltoLinea";
// variable que servira para llevar el control
var banderaReporteTokens=true;
// arreglos que tendran diferentes textos
var lexema = new Array();
var veces= new Array();
var zonaDatos;

// aqui es donde empezara  a leerse el archivo de entrada
```