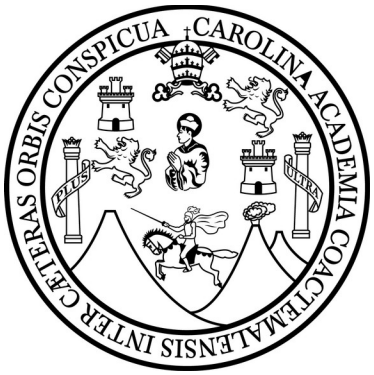
Universidad de San Carlos de Guatemala
Centro Universitario de Occidente
División de Ciencias de la Ingeniería
Lenguajes Formales y de Programación
Docente: Ing. Bryan Monzón



Manual Técnico Practica #1

Estudiante: Juan Fernando García Natareno
Registro Académico: 202230077

El presente documento describe el funcionamiento, diseño e implementación de un analizador léxico, una herramienta fundamental en el proceso de compilación e interpretación de lenguajes de programación. Su propósito es recibir como entrada un código fuente y descomponerlo en una secuencia de tokens, que representan las unidades léxicas del lenguaje, como palabras clave, identificadores, operadores y símbolos de puntuación.

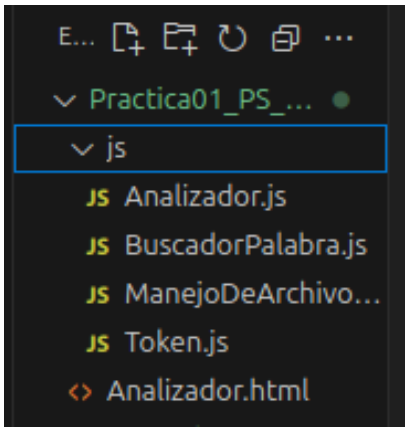
A lo largo del documento, se presentarán los fundamentos teóricos del análisis léxico, la estructura del código, los algoritmos utilizados y las instrucciones para la instalación, configuración y uso del sistema. También se incluyen ejemplos prácticos, posibles errores y soluciones, así como recomendaciones para extender o modificar el analizador según los requerimientos específicos del usuario.

Este analizador léxico ha sido desarrollado con el objetivo de ser eficiente, modular y fácil de mantener, permitiendo su adaptación a distintos lenguajes de programación o dominios específicos.

Descripción general del sistema

Este proyecto ha sido desarrollado utilizando el lenguaje de programación JavaScript para gestionar la lógica del sistema, permitiendo el procesamiento y análisis de datos de manera eficiente. Además, se apoya en el lenguaje de marcado HTML para la construcción de la interfaz de usuario, facilitando la interacción con el sistema a través del Front-End.

El objetivo principal de este desarrollo es proporcionar una herramienta funcional y modular que permita la identificación y análisis de patrones en un determinado conjunto de datos. Para ello, el sistema está estructurado en diferentes archivos y módulos que organizan las funcionalidades de manera clara y escalable.
A continuación, se presenta la estructura del proyecto:



En la carpeta js se guardaron todos los archivos .js que guardan la lógica del proyecto, a continuación se mostrará el archivo Analizador.html:

```
<!DOCTYPE html>
<html lang="en">

<head>

    <style>
        table{
            width: 50%;
            border-collapse: collapse;
            background-color: rgb(43, 43, 43);
        }

        #tableTokens th, #tableTokens td{
            text-align: center;
            color: rgb(5, 168, 168);
        }

        #tableErrores th, #tableErrores td{
            text-align: center;
            color: red;
        }

        #tableRecuento th, #tableRecuento td{
            text-align: center;
            color:  rgb(6, 243, 6);
        }
    </style>

    <Title>Analizador Lexico</Title>
</head>
```

```html
<body style="background-color: rgb(43, 43, 43);">
    <center>
        <h1 style="color: aqua;">Analizador Lexico</h1>
    </center>

    <center>
        <h3 style="color: aliceblue; display: inline-block; margin-right: 10px;">Buscar Palabra: </h3>
        <input type="text" name="campoTexto" id="textField">
        <button id="btnSearch" style="color: darkturquoise; font-size: 20px;">Buscar</button>
    </center>

    <center>
        <textarea name="textarea" id="textarea" rows="20" cols="75"  placeholder="INGRESE TEXTO PARA ANALIZAR" oninput="copyText()"></textarea>
        <textarea name="textarea2" id="textarea2" rows="20" cols="75"  readonly  style="background-color: rgb(58, 58, 58); color: aliceblue;"></textarea>
    </center>
    </br>
    <center> <p id="resultado" style="color: aliceblue;">No se ha buscado ninguna palabra o simbolo</p> </center>
    <center>
        <button id="btnAnalizar" style="background-color: rgb(61, 255, 61); font-size: 20px;">Analiza Archivo</button>
        <button id="btnSubmit" style="background-color: rgb(10, 243, 243); font-size: 20px;" onclick="copyText()">Subir Archivo</button>
        <input type="file" id="fileInput" style="display: none;">
        <p id="nombreArchivo" style="color: aliceblue;">Ningún archivo seleccionado</p>
    </center>
    </br>
    <center>
        <button id="btnChangeTxt" style="background-color: rgb(250, 194, 41); font-size: 20px;">Descargar Archivo con Cambios</button>
    </center>
    </br>
```

```html
<center>
    <div class="tabla-contenedor">
        <h3 style="color: aqua; font-size: 20px;">REPORTE DE TOKENS</h3>
        <table border="1" id="tableTokens">
            <tr>
                <th>Token</th>
                <th>Lexema</th>
                <th>Fila</th>
                <th>Columna</th>
            </tr>
            <tbody id="tabla-lexemas">
                <tr id="mensaje-lexemas">
                    <th colspan="4">AUN NO SE HA ANALIZADO EL TEXTO</th>
                </tr>
            </tbody>
        </table>
    </br>
        <h3 style="color: rgb(252, 43, 28); font-size: 20px;">REPORTE DE ERRORES</h3>
        <table border="1" id="tableErrores">
            <tr>
                <th>Cadena</th>
                <th>Columna</th>
                <th>Fila</th>
            </tr>
            <tbody id="tabla-errores">
                <tr id="mensaje-errores">
                    <th colspan="3">AUN NO SE HA ANALIZADO EL TEXTO</th>
                </tr>
            </tbody>
        </table>
    </br>
        <h3 style="color: rgb(6, 243, 6); font-size: 20px;">RECUENTO DE LEXEMAS</h3>
        <table border="1" id="tableRecuento">
            <tr>
                <th>Lexema</th>
                <th>Cantidad</th>
            </tr>
            <tbody id="tabla-recuento">
                <tr id="mensaje-recuento">
                    <th colspan="2">AUN NO SE HA ANALIZADO EL TEXTO</th>
                </tr>
            </tbody>
        </table>
    </br>
    </div>
</center>
```

A continuación se mostrará los archivos .js

Archivo Token.js: Este archivo es un objeto, con sus atributos:

```
export class Token {
    constructor(simbolo, tipo, fila, columna) {
        this.simbolo = simbolo;
        this.tipo = tipo;
        this.fila = fila;
        this.columna = columna;
    }


    getSimbolo() {
        return this.simbolo;
    }

    getTipo() {
        return this.tipo;
    }

    getFila() {
        return this.fila;
    }

    getColumna(){
        return this.columna;
    }
}
```

Archivo analizador.js: Este archivo es el encargado de la identificación de Tokens y lexemas o los posibles errores.

- En esta parte se declara las cadenas con las cuales haremos las comparaciones y le colocamos un eventListener al botón analizar:

```
import { Token } from "./Token.js";

const token = new Token();

var letras = "ABCDEFGHIJKLMNOPQRSTUVWXYZ".split("");
var signosPuntuacion = [".", ",", ";", ":"];
var operadoresAritmeticos = ["^", "*", "/", "+", "-"];
var operadoresRacionales = ["<", ">", "<=", ">="];
var operadoresLogicos = ["AND", "&&", "OR", "||"];
var signosDeAgrupacion = ["(", ")", "[", "]", "{", "}"];
var asignacion = ["="];

/*
SE AGRERA UN ACTION LISTENER AL BOTON btnAnalizar Y ESTE OBTENDRA EL
TEXTO QUE ESTE EN textarea Y LO COLOCA EN LA VARIABLE TEXTO
*/
document.getElementById("btnAnalizar").addEventListener("click", () => {
    let texto = document.getElementById("textarea").value;
    let { lexemas, errores } = tokenYlexemas(texto);

    llenarTablaLexemas(lexemas, errores);
    llenarTablaErrores(errores, texto);
    contadorDeLexemas(lexemas, errores);

});
```

- En esta parte, primeramente se separa la entrada cuando exista un salto de linea, seguidamente para poder analizar símbolo por símbolo se separa cuando exista un espacio y luego se hace un las comparaciones. Y también cada que se analiza un símbolo se le suma 1 a la columna, y cuando hay un salto de línea se le suma 1 a las filas.

```
function tokenYlexemas(entrada) {
    // Dividir el texto por saltos de línea primero
    let lineas = entrada.split("\n");
    let lexemas = [];
    let errores = [];
    let fila = 1, columna = 1;

    // Procesar cada línea
    for (let linea of lineas) {
        // Dividir cada línea por espacios
        let partes = linea.split(" ").filter(palabra => palabra !== "");

        for (let parte of partes) {
            // Procesar cada palabra (parte)
            let num = parseInt(parte, 10);
            if (letras.some(letra => parte.toUpperCase().startsWith(letra)) && (parte !== "AND" && parte !== "OR" && parte !== "_")) {
                //Si entra a la condicion entonces se creara un nuevo Token
                partes = parte.split("");
                for(let i=0; i < partes.length; i++){
                    if(partes[i] === "_"){
                        errores.push({ cadena: parte, fila, columna });
                        break;
                    }
                }
                lexemas.push(new Token(parte, "Identificador", fila, columna));
            } else if (num < 10 || num > 0) {
                lexemas.push(new Token(parte, "Número", fila, columna));
            } else if (signosPuntuacion.includes(parte)) {
                lexemas.push(new Token(parte, "Signo de Puntuación", fila, columna));
            } else if (operadoresAritmeticos.includes(parte)) {
                lexemas.push(new Token(parte, "Operador Aritmético", fila, columna));
            } else if (operadoresRacionales.includes(parte)) {
                lexemas.push(new Token(parte, "Operador Racional", fila, columna));
            } else if (operadoresLogicos.includes(parte)) {
                lexemas.push(new Token(parte, "Operador Lógico", fila, columna));
            } else if (signosDeAgrupacion.includes(parte)) {
                lexemas.push(new Token(parte, "Signo de Agrupación", fila, columna));
            } else if (asignacion.includes(parte)) {
                lexemas.push(new Token(parte, "Asignación", fila, columna));
            } else {
                errores.push({ cadena: parte, fila, columna });
            }
            // Actualizamos la columna en función del tamaño de la palabra
            columna = columna + parte.length + 1; // +1 por el espacio
        }
        // Al final de cada línea, aumentamos la fila y reiniciamos la columna
        fila++;
        columna = 1;
    }
```

-Esta parte se hará el conteo de lexemas.

```
/**
 * Este metodo contara la cantidad de lexemas que haya en el codigo
 * @param {*} arrayLexemas
 * @param {*} errores
 */
function contadorDeLexemas(arrayLexemas, errores) {
    let simbolos = [];
    let cantidadLexemas = [];

    for (let i = 0; i < arrayLexemas.length; i++) {
        let simbolo = arrayLexemas[i].getSimbolo(); // OBTENEMOS LOS SIMBOLOS DE LOS LEXEMAS

        let indice = simbolos.indexOf(simbolo); // BUSCA SI YA EXISTE

        if (indice === -1) {
            //Si el simbolo no esta en el arreglo de simbolos, le suma uno al contador de lexemas
            simbolos.push(simbolo);
            cantidadLexemas.push(1);
        } else {
            //Si esta el simbolo, significa que esta repetido entonces le suma 1 a la cantidad que ya tenia
            cantidadLexemas[indice]++;
        }
    }

    llenarTablaRecuento(simbolos, cantidadLexemas, errores);
}
```

-En esta parte se llena las tablas, hay un método para cada tabla, es decir tabla de Reporte de Tokens, Reporte de Errores y Conteo de Lexemas.

```
/**
 * Esta funcion llenara la tabla pra mostrar los lexemas, así mismo
 * si encuentra un error no mostrara el reporte de Lexemas
 * @param {*} lexemas
 * @param {*} errores
 * @returns
 */
function llenarTablaLexemas(lexemas, errores) {
    let tbody = document.getElementById("tabla-lexemas");
    tbody.innerHTML = "";

    if (lexemas.length === 0) {
        tbody.innerHTML = `<tr><th colspan="4">No se encontraron tokens</th></tr>`;
        return;
    } else if(errores.length > 0){
        tbody.innerHTML = `<tr><th colspan="4">Se han dectectado errores, Arregle los errores para ver la tabla de Tokens y Lexemas</th></tr>`;
        return;
    }


    lexemas.forEach(token => {
        let fila = document.createElement("tr");
        fila.innerHTML = `
            <td>${token.getTipo()}</td>
            <td>${token.getSimbolo()}</td>
            <td>${token.fila}</td>
            <td>${token.columna}</td>
        `;
        tbody.appendChild(fila);
    });
}
```

Archivo ManejoDeArchivos.js: En este archivo esta la lógica para poder leer y copiar el contenido de un archivo de texto y copiarlo en un textarea:

```
function copyText() {
    document.getElementById("textarea2").value = document.getElementById("textarea").value;
}
/*
ESTA PARTE DEL CODIGO LE AGREGA EL BOTON "btnSubmit" UN ACTION LISTENER.
AL DARLE CLIC AL BOTON ESTO HACE QUE SE DESPLIGUE UN EXPLORADOR DE ARCHIVOS
*/
document.getElementById("btnSubmit").addEventListener("click", ()=>{
    document.getElementById("fileInput").click();
});

/*
AGREGA UN ACTION LISTENER AL INPUT, ESTE DETECTA EN CUANDO EL USUARIO SELECCIONA
UN ARCHIVO Y LO LEERA
*/                                                          (parameter) event: Event
document.getElementById("fileInput").addEventListener("change", function(event){
    const archivo = event.target.files[0]; // OBTIENE EL PRIMERO ARCHIVO SELECCIONADO
    if(!archivo) return; // SI NO ES SELECCIONADO NINGUN ARCHIVO SE DETIENE LE EJECUCION

    //EN LA ETIQUETA <p> SE MOSTRARA EL NOMBRE DEL ARCHIVO QUE SELECCIONO EL USUARIO
    nombreArchivo.textContent = `Archivo cargado: ${archivo.name}`;

    const entrada = new FileReader(); // SE CREA UN OBJETO FileReader
    entrada.onload = function(e){
        //e.targer.result CONTIENE EL TEXTO QUE CONTIENE EL ARCHIVO Y SE INSERTA EN EL textarea
        //ESTO SOLO PASA CUANDO LA FUNCION onload INICIE ESTO OCURRE CUANDO SE TERMINA DE LEER EL ARCHIVO
        document.querySelector("textarea").value = e.target.result;
        copyText();
    };
    entrada.readAsText(archivo); //ESTO ES LO QUE LEE EL TEXTO QUE ESTA EN EL ARCHIVO
});

//METODO QUE ESCRIBIRA LO QUE ESTA EN EL TEXT AREA Y LO LUEGO LO DESCARGARA
document.getElementById("btnChangeTxt").addEventListener("click", ()=>{
    //Obtiene el texto del textarea
    let texto = document.getElementById("textarea").value;

    //Se crea un objeto blob, que almacena el contenido y se especifica que sera texto plano
    let blob = new Blob([texto], { type: 'text/plain' });
    let enlace = document.createElement("a");

    //Se genera un URL temporal a partir del Blob,
    enlace.href = URL.createObjectURL(blob); //Permite que el navegador trate al blob como archivo descargable
    enlace.download = "Texto Modificado - Analizador.txt"; //Es el nombre de como se descargara el archivo
    //FORZA LA DESCARGA DEL ARCHIVO
    document.body.appendChild(enlace); //Se añade temporalmente el enlace al body
    enlace.click();
    document.body.removeChild(enlace); // Se elimina el enlace despues de la descarga
});
```

Archivo BuscadorPalabra.js: Este método busca la palabra en el contenido del textarea

```javascript
/**
 * Añade un listener al boton "btnSearch", se obtiene el texto en el textarea
 * y la palabra a buscar que esta en el input
*/
document.getElementById("btnSearch").addEventListener("click",()=>{
    let texto = document.getElementById("textarea2").value;
    let palabra = document.getElementById("textField").value;

    buscar(texto, palabra);
});

function buscar(entrada, palabraAbuscar){
    //Se para la entrada cada salto de pagina, y elimina los espacios en blanco
    let partes = entrada.split(/[\n ]/).filter(palabra => palabra !== "");
    let contadorPalabra = 0;

    //Recorremos la entrada, y si la parte de la entrada coincide con la palabraAbuscar
    //entonces le sumamos 1 al contadorPalabra
    for(let parte of partes){
        if(parte === palabraAbuscar){
            contadorPalabra++;
        } else if (parte !== palabraAbuscar || parte === " "){
            continue;
        }
    }

    //Lo que nos indica si la palabra fue encontrada es el contadorPalabra, y colocamos un
    //mensaje segun la cantidad de contadorPalabra
    if(contadorPalabra === 0){
        alert("NO SE HA ENCONTRADO EL SIMBOLO O LEXEMA")
        document.getElementById("resultado").textContent = "NO SE HA ENCONTRADO EL SIMBOLO O LEXEMA";
    } else {
        resaltarPalabraEnTextarea(palabraAbuscar);
        document.getElementById("resultado").textContent = "La palabra/símbolo: (" + palabraAbuscar + ") se ha encontrado un total de " + contadorPalabra + " veces";
    }

    resaltarPalabraEnTextarea(palabraAbuscar);
}

function resaltarPalabraEnTextarea(palabra) {
    //Se obtiene el texto que esta en el textarea2
    let textarea = document.getElementById("textarea2");
    let texto = textarea.value;
    //Busca la primera aparicion de palabraAbusca y devolvera el indice
    let indice = texto.indexOf(palabra);

    //Si si encuentra el indice entonces reslatara la palabra
    if (indice !== -1) {
        textarea.focus(); // ENFOCA EL textarea
        textarea.setSelectionRange(indice, indice + palabra.length); // SELECCIONA LA PALABRA s
    }
}
```

El desarrollo de este analizador léxico ha permitido la implementación de un sistema capaz de descomponer y clasificar los distintos elementos de un lenguaje en tokens de manera eficiente. A través del uso de JavaScript para la lógica del programa y HTML para la interfaz de usuario, se logró una herramienta funcional y modular que facilita el análisis de cadenas de entrada.

Durante la implementación, se estableció una estructura clara del proyecto, organizando los archivos de manera que permitan su mantenimiento y escalabilidad. Además, se aseguraron mecanismos de validación para la correcta identificación de lexemas, minimizando errores en la segmentación y clasificación de los componentes léxicos.

Este sistema puede servir como base para futuras mejoras e integraciones con otros módulos, como un analizador sintáctico o un intérprete. Su flexibilidad permite adaptaciones para otros lenguajes o necesidades específicas, convirtiéndolo en una herramienta versátil dentro del procesamiento de lenguajes.

En conclusión, este proyecto no solo cumple con su objetivo principal de realizar análisis léxico, sino que también sienta las bases para desarrollos más avanzados en la interpretación y compilación de código.