[Date]

Universidad de Guadalajara
CUCEI

Centro Universitario de Ciencias Exactas e Ingenierías
Departamento de Ciencias Computacionales

Carrera: Ingeniera en sistemas computacionales

Materia: Seminario de Solución de problemas de traductores de lenguaje II

Sección: D02

Alumno: Nudelstejer Gómez Iván

Código: 218130122

Profesor: Michel Emanuel López Franco

Objetivo:

Desarrollar un componente específico de un traductor (puede ser un compilador o un intérprete) para un lenguaje de programación simplificado. Este proyecto se realizará en varias fases, cada una centrada en una parte diferente del proceso de traducción.

Fase Actual: Análisis Léxico y Sintáctico

Descripción:

En esta fase, se ha implementado el analizador léxico y sintáctico para el lenguaje de programación asignado. El lenguaje tiene una sintaxis y un conjunto de tokens definidos previamente. El analizador léxico convierte el código fuente en una secuencia de tokens, mientras que el analizador sintáctico verifica la estructura gramatical del programa.

Requerimientos:

1. Tokens Definidos: Se han definido los tokens que formarán parte del lenguaje, incluyendo palabras reservadas, identificadores, literales numéricos y operadores. Estos están reflejados en las clases `Lexico` y `Sintactico`.

2. Implementación del Analizador Léxico: Se ha creado la clase `Lexico` que lee el código fuente y produce una lista de tokens. Además, maneja y reporta errores léxicos como caracteres inválidos o formatos incorrectos de tokens.

3. Manejo de Errores: El código incluye manejo de errores léxicos y sintácticos. Por ejemplo, se reporta un "Error léxico" en caso de detectar un carácter inválido, y un "Error sintáctico" cuando la estructura gramatical no es correcta.

Entregables:

1. Código Fuente:

- `lexico.js`: Contiene la implementación del analizador léxico.

- `sintactico.js`: Contiene la implementación del analizador sintáctico.

2. **Documento de Diseño:

- Incluye una explicación del diseño, decisiones clave durante la implementación y cómo se manejan los errores léxicos y sintácticos.

3. Archivos de Muestra:

- Archivos de entrada de muestra (`input`) junto con la salida producida por los analizadores léxico y sintáctico.

- Incluye casos que demuestran el manejo correcto de los tokens y ejemplos que muestran cómo se manejan los errores léxicos y sintácticos.

En la fase actual, se ha logrado la integración de ambos analizadores, estableciendo las bases para las fases posteriores del proceso de traducción.

Evaluación:

- Criterio: ¿El analizador léxico identifica correctamente todos los tokens definidos en la especificación del lenguaje?

- Observación: Se evaluará la capacidad del analizador léxico para reconocer y clasificar adecuadamente los tokens según la definición del lenguaje.

- Criterio: ¿El analizador léxico proporciona mensajes de error útiles y precisos para entradas inválidas?

- Observación: Se revisará cómo el analizador léxico maneja y reporta errores léxicos, asegurando mensajes claros y descriptivos que ayuden a entender y corregir los problemas en el código fuente.

- Criterio: ¿El código está bien organizado, comentado y sigue las buenas prácticas de programación?

- Observación: Se evaluará la estructura general del código, la presencia de comentarios explicativos y el seguimiento de buenas prácticas de programación, como la legibilidad del código y el uso adecuado de nombres de variables y funciones.

- Criterio: ¿La documentación proporciona una buena visión general de tu implementación y decisiones?

- Observación: Se analizará la calidad y claridad de la documentación, incluyendo explicaciones sobre el diseño del analizador léxico, decisiones importantes tomadas durante la implementación y cómo se gestionan los errores léxicos.

“Script.js”

```
import {Lexico} from './classes/lexico.js';
import { Semantico } from './classes/semantico.js';
import { Sintactico } from './classes/sintactico.js';
import { GenCode } from './classes/genCode.js';
import {Nodo, Arbol} from './classes/Tree/arbol.js';

let sint = new Sintactico();

const botonlex1 = document.getElementById("boton");
botonlex1.addEventListener("click", () => {
    ejecutaranalizador(1);
});

const obtenerFuente = () => {
    const input = document.getElementById("input");
    let fuente = input.value;
    fuente = fuente.trim();
    return fuente;
}

const imprimirRubros = (columna1, columna2, columna3) => {
    for (let i = columna1.length; i < 30; i++) {
        columna1 += " ";
    }

    for (let i = columna2.length; i < 20; i++) {
        columna2 += " ";
    }

    console.log(columna1 + columna2 + columna3);
}

const generarFilasTabla = (columna1, columna2, columna3) => {
    let resultado = document.getElementById("resultado");
    let tabla = resultado.firstElementChild;

    let row = document.createElement("DIV");

    let column1 = document.createElement("DIV");
    let span1 = document.createElement("SPAN");
    column1.classList.add("column");
    span1.textContent = columna1;
    column1.appendChild(span1);
    row.appendChild(column1);

    let column2 = document.createElement("DIV");
    let span2 = document.createElement("SPAN");
    column2.classList.add("column");
    span2.textContent = columna2;
    column2.appendChild(span2);
    row.appendChild(column2);
```

```
    let column3 = document.createElement("DIV");
    let span3 = document.createElement("SPAN");
    column3.classList.add("column");
    span3.textContent = columna3;
    column3.appendChild(span3);
    row.appendChild(column3);

    row.classList.add("row", "row-content");

    tabla.appendChild(row);
}

const reiniciaTabla = () => {
    //let resultado = document.getElementById("resultado");
    //let tabla = resultado.firstElementChild;
    let vacio = document.getElementById("vacio");
    let mensaje = document.getElementById("mensaje-resultado");
    vacio.classList.add("oculto");
    mensaje.classList.add("oculto");


    /*resultado.classList.remove("oculto");

    [...tabla.children].forEach(element => {
        if(element != tabla.firstElementChild) {
            tabla.removeChild(element);
        }
    });*/
}

const reiniciarArbol = () => {
    let titulo = document.getElementById("titulo-arbol");
    let arbol = document.getElementById("arbol");
    let lista = arbol.firstElementChild;
    titulo.classList.add("oculto");
    arbol.classList.add("oculto");
    [...lista.children].forEach(element => {
        lista.removeChild(element);
    });
}

const entradaVaciaMsj = () => {
    /*let resultado = document.getElementById("resultado");
    resultado.classList.add("oculto");*/

    let aceptacion = document.getElementById("mensaje-resultado");
    aceptacion.classList.add("oculto");

    /*let titulo = document.getElementById("titulo-arbol");
    let arbol = document.getElementById("arbol");
    titulo.classList.add("oculto");
    arbol.classList.add("oculto");*/

    let vacio = document.getElementById("vacio");
```

```
    vacio.classList.remove("oculto");

    let botonDescarga = document.getElementById("archivo");
    botonDescarga.classList.add("oculto");
}

const resultadoMsj = (texto) => {
    let botonDescarga = document.getElementById("archivo");
    let aceptacion = document.getElementById("mensaje-resultado");
    aceptacion.classList.remove("oculto");
    botonDescarga.classList.add("oculto");
    aceptacion.firstElementChild.textContent = texto;
}

const imprimirArbol = (arbol) => {
    let tituloDom = document.getElementById("titulo-arbol");
    let arbolDom = document.getElementById("arbol");
    let listaDom = arbolDom.firstElementChild;
    arbolDom.classList.remove("oculto");
    tituloDom.classList.remove("oculto");

    let espacio = 0;
    let fragmento = new DocumentFragment();
    arbol.raiz.forEach(rama => {
        imprimirNodo(rama, espacio, fragmento);
    });
    listaDom.appendChild(fragmento);
}

const imprimirNodo = (rama, espacio, fragmento) => {
    let espaciado = "";
    for (let i = 0; i < espacio; i++) {
        espaciado += "__";
    }
    //console.log(espaciado + espacio + ". " + rama.simbolo);
    let ramaDom = document.createElement("li");
    ramaDom.textContent = espaciado + espacio + ". " + rama.simbolo;
    ramaDom.classList.add("parte-arbol");
    if(rama.ramas.length == 0) {
        ramaDom.classList.add("hoja");
    }
    fragmento.appendChild(ramaDom);
    rama.ramas.forEach(elemento => {
        imprimirNodo(elemento, espacio + 1, fragmento);
    });
}

const ejecutaranalizador = () => {
    let fuente = "";
    fuente = obtenerFuente();
    let lex = new Lexico(fuente);
    let seman;
    let genCode;
```

```
    //console.log("Resultado del analisis Sintactico");
    //console.log("Resultado del anlisis Lexico");
    //console.log("Arbol Sintactico");
    //console.log("Resultado del analisis Semantico")

    if(fuente.length != 0) {
        //console.log("Pila                        Entrada
Accion   ");
        reiniciaTabla();
        //reiniciarArbol();

        sint.inicializarPila();
        sint.inicializarArbol();
        let arbol = new Arbol();

        while(!lex.termina()) {
            lex.obtenerSimbolo();

            let opcion = 0;

            do {
                if(lex.tipo == 404) {
                    resultadoMsj("Error lexico");
                    console.log("Error lexico");
                    lex.caracter = "$";
                    break;
                }
                sint.sigEntrada(lex.tipo);

                //imprimirRubros(sint.pila.toString(), lex.simbolo,
sint.accion);
                //generarFilasTabla(sint.pila.toString(), lex.simbolo,
sint.accion);

                opcion = sint.sigAccion(lex.simbolo);

                if(opcion == 2) {
                    //imprimirRubros(sint.pila.toString(), lex.simbolo,
sint.accion);
                    //generarFilasTabla(sint.pila.toString(), lex.simbolo,
sint.accion);

                } else if(opcion == 3) {
                    resultadoMsj("Error sintactico");
                    console.log("Error sintactico");
                    lex.caracter = "$";
                    break;

                } else if(opcion == 4) {
                    resultadoMsj("Aceptacion sintactico");
                    console.log("Aceptacion sintactico");
                    arbol = sint.arbol;
                    //imprimirArbol(arbol);
```

```
                    seman = new Semantico(arbol);
                    seman.analisis();
                    if(seman.valido)
                    {
                        resultadoMsj("Aceptacion semantico");

                        genCode = new GenCode(arbol);
                        let blob = genCode.generarCodigo();

                        // Crear el enlace de descarga
                        let archivo = document.getElementById("archivo");
                        let enlace = archivo.firstElementChild;
                        let boton = enlace.firstElementChild;
                        boton.textContent = "Descargar Codigo";
                        enlace.download = 'codigo.asm';
                        enlace.href = URL.createObjectURL(blob);
                        archivo.classList.remove("oculto");
                    }
                    else
                        resultadoMsj("Error semantico\n" + seman.error);
                    break;
                }
            } while(opcion == 2);
        }
    } else {
        console.log("Entrada vacia");
        entradaVaciaMsj();
    }
}
```