Universidad de Guadalajara CUCEI

“Gramática del compilador”

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

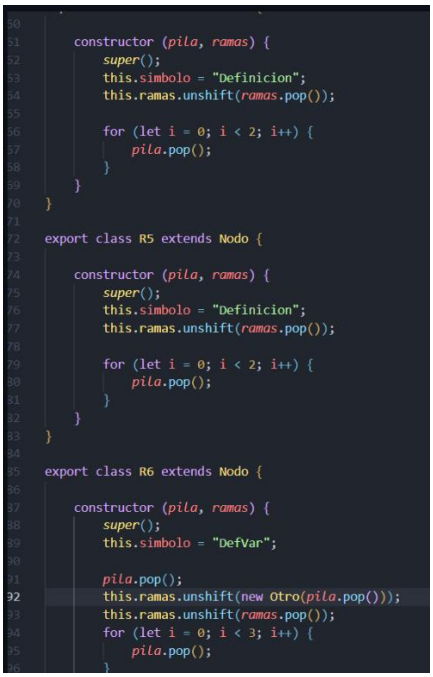
Alumno: Nudelstejer Gómez Iván

Código: 218130122

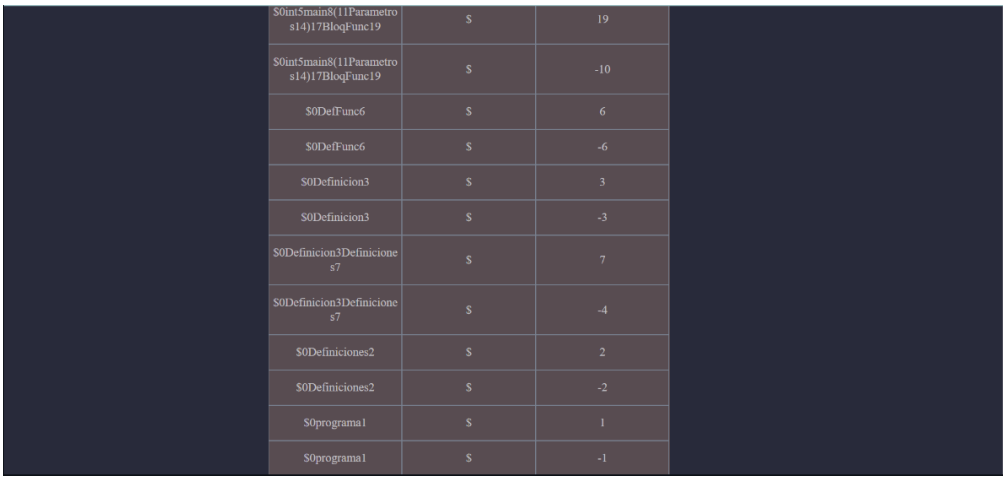
Profesor: Michel Emanuel López Franco

Capturas:

```
You, 16 minutes ago | 1 author (You)
import { Pila } from './pila.js';       You, 16 minutes ago • Analizador sint
You, 16 minutes ago | 1 author (You)
class Sintactico {

    constructor() {
        this.pila = new Pila();
        this.fila = 0;
        this.columna = 0;
        this.accion = 0;
        this.idReglas = [];
        this.lonReglas = [];
        this.simReglas = [];
        this.tablaLR = [];


        fetch('/GramaticaCompilador/compilador.lr').then(res => res.text())
        .then(content => {
            let lines = content.split(/\n/);
            let numReglas = parseInt(lines.shift());
            for (let i = 0; i < numReglas; i++) {
                lines[i] = lines[i].replace("\r", "");
                let regla = lines[i].split("\t");
                this.idReglas.push(regla[0]);
                this.lonReglas.push(regla[1]);
                this.simReglas.push(regla[2]);
            }

            let numFila = parseInt(lines[numReglas].split("\t").shift());
            numReglas++;
            for (let i = numReglas; i < numReglas + numFila; i++) {
                lines[i] = lines[i].replace("\r", "");
                let fila = lines[i].split("\t");
                this.tablaLR.push(fila);
            }
        });
    }
```



```
    constructor (pila, ramas) {
        super();
        this.simbolo = "Definicion";
        this.ramas.unshift(ramas.pop());

        for (let i = 0; i < 2; i++) {
            pila.pop();
        }
    }
}

export class R5 extends Nodo {

    constructor (pila, ramas) {
        super();
        this.simbolo = "Definicion";
        this.ramas.unshift(ramas.pop());

        for (let i = 0; i < 2; i++) {
            pila.pop();
        }
    }
}

export class R6 extends Nodo {

    constructor (pila, ramas) {
        super();
        this.simbolo = "DefVar";

        pila.pop();
        this.ramas.unshift(new Otro(pila.pop()));
        this.ramas.unshift(ramas.pop());
        for (let i = 0; i < 3; i++) {
            pila.pop();
        }
```



$0int5main8(11Parametros14)17BloqFunc19	$	19
$0int5main8(11Parametros14)17BloqFunc19	$	-10
$0DefFunc6	$	6
$0DefFunc6	$	-6
$0Definicion3	$	3
$0Definicion3	$	-3
$0Definicion3Definiciones7	$	7
$0Definicion3Definiciones7	$	-4
$0Definiciones2	$	2
$0Definiciones2	$	-2
$0programa1	$	1
$0programa1	$	-1

“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();
    }
}
```

"Pila.js"

```
import * as pilaElem from "./elementoPila.js";

class Pila {

    constructor() {
        this.pila = [];
    }

    push(element, tipo) {
        let elemento;
        switch(tipo) {
            case 1:
                elemento = new pilaElem.Terminal(element);
                break;

            case 2:
                elemento = new pilaElem.NoTerminal(element);
                break;
```

```
            case 3:
                elemento = new pilaElem.Estado(element);
                break;
        }
        this.pila.push(elemento);
    }

    front() {
        return this.pila[this.pila.length -1].value();
    }

    pop() {
        return this.pila.pop();
    }

    vaciar() {
        this.pila = [];
    }

    toString() {
        let string = "";
        this.pila.forEach(element => {
            string += element.string();
        });
        return string;
    }
}

export {Pila};
```

"GeneCode"

```
import { Otro, R21, R22, R23, R24, R25, R35, R36, R37, R38, R39, R43,
R44, R45, R46, R47, R48, R49, R50, R51, R52 } from
"./Tree/reglasClasses.js";

class GenCode {
    constructor(arbol) {
        this.arbol = arbol;
        this.codigoRes = [];
        this.auxTipo = "int";
        this.globales = [];
        this.contadorOP = 0;
        this.contadorARG = 0;
        this.contadorIf = 0;
        this.contadorWhile = 0;
        this.variables = [];
    }

    generarCodigo() {
        this.declaracionCodigo();
        return this.crearDocumento();
```

```
}

declaracionCodigo() {
    this.codigoRes.push(".MODEL SMALL\n.STACK 100H\n.DATA\n");

    this.codigoRes.push("\n\n");
    this.codigoRes.push("RESGEN DW ?\n");
    this.codigoRes.push("RESFUN DW ?\n");
    this.codigoRes.push("RESIMP DW ?\n");
    this.codigoRes.push("NUMIMP DW ?\n");
    this.codigoRes.push("RES DW 100 DUP(0)\n");
    this.codigoRes.push("ARG DW 100 DUP(0)\n");
    this.codigoRes.push("CONT DW 0\n");
    this.arbol.raiz.forEach(rama => {
        this.obtenerVariables(rama, "*");
    });

    this.codigoRes.push("\n\n");
    this.agregarMetodosBasicos();
    this.arbol.raiz.forEach(rama => {
        this.obtenerFunciones(rama, "*");
    });

    this.agregarCodigoBase();
}

obtenerVariables(rama, ambito) {
    if(rama.simbolo == "DefVar") {
        let tipo = rama.ramas[0].simbolo;
        let id = rama.ramas[1].simbolo;
        this.tipo = tipo;
        this.agregarVariable(tipo, id, ambito);
    }

    if(rama.simbolo == "ListaVar") {
        if(rama.ramas.length != 0) {
            let tipo = this.tipo;
            let id = rama.ramas[1].simbolo;
            this.agregarVariable(tipo, id, ambito);
        }
    }

    if(rama.simbolo == "DefFunc") {
        ambito = rama.ramas[1].simbolo;
    }

    rama.ramas.forEach(subrama => {
        this.obtenerVariables(subrama, ambito);
    });
}

agregarVariable(tipo, id, ambito) {
    if(ambito == "*") {
        this.codigoRes.push(id + "0");
```

```
            this.globales.push(id);
        } else {
            this.codigoRes.push(id + ambito);
        }

        if(tipo == "int") {
            this.codigoRes.push(" DW ?\n");
        } else if(tipo == "float") {
            this.codigoRes.push(" Dw ?\n");
        } else if(tipo == "void") {
            this.codigoRes.push(" DW 0\n");
        } else if(tipo == "char") {
            this.codigoRes.push(" DW ?\n");
        }

        let variable = {
            tipo: tipo,
            id: id,
            ambito: ambito
        }
        this.variables.push(variable);
    }

    agregarMetodosBasicos() {
        this.codigoRes.push("MENORVAR MACRO NUM1, NUM2" + "\n");
        this.codigoRes.push("LOCAL ESMAYOR1" + "\n");
        this.codigoRes.push("LOCAL ESMENOR1" + "\n");
        this.codigoRes.push("LOCAL SALIR1" + "\n");
        this.codigoRes.push("MOV AX, NUM1" + "\n");
        this.codigoRes.push("MOV BX, NUM2" + "\n");
        this.codigoRes.push("CMP AX, BX" + "\n");
        this.codigoRes.push("JA ESMAYOR1" + "\n");
        this.codigoRes.push("JB ESMENOR1" + "\n");
        this.codigoRes.push("JE ESMAYOR1" + "\n\n");
        this.codigoRes.push("ESMAYOR1:" + "\n");
        this.codigoRes.push("MOV RESGEN, 0" + "\n");
        this.codigoRes.push("JMP SALIR1" + "\n\n");
        this.codigoRes.push("ESMENOR1:" + "\n");
        this.codigoRes.push("MOV RESGEN, 1" + "\n");
        this.codigoRes.push("JMP SALIR1" + "\n");
        this.codigoRes.push("SALIR1:" + "\n");
        this.codigoRes.push("ENDM" + "\n");
        this.codigoRes.push("\n");

        this.codigoRes.push("MAYORVAR MACRO NUM1, NUM2" + "\n");
        this.codigoRes.push("LOCAL ESMAYOR2" + "\n");
        this.codigoRes.push("LOCAL ESMENOR2" + "\n");
        this.codigoRes.push("LOCAL SALIR2" + "\n");
        this.codigoRes.push("MOV AX, NUM1" + "\n");
        this.codigoRes.push("MOV BX, NUM2" + "\n");
        this.codigoRes.push("CMP AX, BX" + "\n");
        this.codigoRes.push("JA ESMAYOR2" + "\n");
        this.codigoRes.push("JB ESMENOR2" + "\n");
        this.codigoRes.push("JE ESMENOR2" + "\n\n");
```

```
this.codigoRes.push("ESMAYOR2:" + "\n");
this.codigoRes.push("MOV RESGEN, 1" + "\n");
this.codigoRes.push("JMP SALIR2" + "\n\n");
this.codigoRes.push("ESMENOR2:" + "\n");
this.codigoRes.push("MOV RESGEN, 0" + "\n");
this.codigoRes.push("JMP SALIR2" + "\n");
this.codigoRes.push("SALIR2:" + "\n");
this.codigoRes.push("ENDM" + "\n");
this.codigoRes.push("\n");

this.codigoRes.push("MAYORIGUALVAR MACRO NUM1, NUM2" + "\n");
this.codigoRes.push("LOCAL ESMAYOR3" + "\n");
this.codigoRes.push("LOCAL ESMENOR3" + "\n");
this.codigoRes.push("LOCAL SALIR3:" + "\n");
this.codigoRes.push("MOV AX, NUM1" + "\n");
this.codigoRes.push("MOV BX, NUM2" + "\n");
this.codigoRes.push("CMP AX, BX" + "\n");
this.codigoRes.push("JA ESMAYOR3" + "\n");
this.codigoRes.push("JB ESMENOR3" + "\n");
this.codigoRes.push("JE ESMAYOR3" + "\n\n");
this.codigoRes.push("ESMAYOR3:" + "\n");
this.codigoRes.push("MOV RESGEN, 1" + "\n");
this.codigoRes.push("JMP SALIR3" + "\n\n");
this.codigoRes.push("ESMENOR3:" + "\n");
this.codigoRes.push("MOV RESGEN, 0" + "\n");
this.codigoRes.push("JMP SALIR3" + "\n");
this.codigoRes.push("SALIR3:" + "\n");
this.codigoRes.push("ENDM" + "\n");
this.codigoRes.push("\n");

this.codigoRes.push("MENORIGUALVAR MACRO NUM1, NUM2" + "\n");
this.codigoRes.push("LOCAL ESMAYOR4" + "\n");
this.codigoRes.push("LOCAL ESMENOR4" + "\n");
this.codigoRes.push("LOCAL SALIR4" + "\n");
this.codigoRes.push("MOV AX, NUM1" + "\n");
this.codigoRes.push("MOV BX, NUM2" + "\n");
this.codigoRes.push("CMP AX, BX" + "\n");
this.codigoRes.push("JA ESMAYOR4" + "\n");
this.codigoRes.push("JB ESMENOR4" + "\n");
this.codigoRes.push("JE ESMENOR4" + "\n\n");
this.codigoRes.push("ESMAYOR4:" + "\n");
this.codigoRes.push("MOV RESGEN, 0" + "\n");
this.codigoRes.push("JMP SALIR4" + "\n\n");
this.codigoRes.push("ESMENOR4:" + "\n");
this.codigoRes.push("MOV RESGEN, 1" + "\n");
this.codigoRes.push("JMP SALIR4" + "\n");
this.codigoRes.push("SALIR4:" + "\n");
this.codigoRes.push("ENDM" + "\n");
this.codigoRes.push("\n");

this.codigoRes.push("IGUALVAR MACRO NUM1, NUM2" + "\n");
this.codigoRes.push("LOCAL ESIGUAL1" + "\n");
this.codigoRes.push("LOCAL SALIR5" + "\n");
this.codigoRes.push("MOV AX, NUM1" + "\n");
```

```
this.codigoRes.push("MOV BX, NUM2" + "\n");
this.codigoRes.push("CMP AX, BX" + "\n");
this.codigoRes.push("JE ESIGUAL1" + "\n");
this.codigoRes.push("MOV RESGEN, 0" + "\n");
this.codigoRes.push("JMP SALIR5" + "\n\n");
this.codigoRes.push("ESIGUAL1:" + "\n");
this.codigoRes.push("MOV RESGEN, 1" + "\n");
this.codigoRes.push("JMP SALIR5" + "\n");
this.codigoRes.push("SALIR5:" + "\n");
this.codigoRes.push("ENDM" + "\n");
this.codigoRes.push("\n");

this.codigoRes.push("NOIGUALVAR MACRO NUM1, NUM2" + "\n");
this.codigoRes.push("LOCAL ESIGUAL2" + "\n");
this.codigoRes.push("LOCAL SALIR6" + "\n");
this.codigoRes.push("MOV AX, NUM1" + "\n");
this.codigoRes.push("MOV BX, NUM2" + "\n");
this.codigoRes.push("CMP AX, BX" + "\n");
this.codigoRes.push("JE ESIGUAL2" + "\n");
this.codigoRes.push("MOV RESGEN, 1" + "\n");
this.codigoRes.push("JMP SALIR6" + "\n\n");
this.codigoRes.push("ESIGUAL2:" + "\n");
this.codigoRes.push("MOV RESGEN, 0" + "\n");
this.codigoRes.push("JMP SALIR6" + "\n");
this.codigoRes.push("SALIR6:" + "\n");
this.codigoRes.push("ENDM" + "\n");
this.codigoRes.push("\n");

this.codigoRes.push("IMPNUM MACRO NUM1" + "\n");
this.codigoRes.push("MOV DL, NUM1" + "\n");
this.codigoRes.push("MOV AH, 02H" + "\n");
this.codigoRes.push("ADD DL, 30H" + "\n");
this.codigoRes.push("INT 21H" + "\n");
this.codigoRes.push("ENDM" + "\n");
this.codigoRes.push("\n");

this.codigoRes.push("IMPCHAR MACRO CHAR" + "\n");
this.codigoRes.push("MOV DL, CHAR" + "\n");
this.codigoRes.push("MOV AH, 02H" + "\n");
this.codigoRes.push("INT 21H" + "\n");
this.codigoRes.push("ENDM" + "\n");
this.codigoRes.push("\n");

this.codigoRes.push("IMPRIMIRCHAR MACRO CHAR1" + "\n");
this.codigoRes.push("MOV AX, CHAR1" + "\n");
this.codigoRes.push("MOV DL, AH" + "\n");
this.codigoRes.push("MOV AH, 02H" + "\n");
this.codigoRes.push("INT 21H" + "\n");
this.codigoRes.push("MOV AX, CHAR1" + "\n");
this.codigoRes.push("MOV DL, AL" + "\n");
this.codigoRes.push("MOV AH, 02H" + "\n");
this.codigoRes.push("INT 21H" + "\n");
this.codigoRes.push("IMPCHAR 0AH" + "\n");
this.codigoRes.push("IMPCHAR 0DH" + "\n");
```

```
this.codigoRes.push("ENDM" + "\n");
this.codigoRes.push("\n");

this.codigoRes.push("IMPRIMIRINT MACRO NUM1" + "\n");
this.codigoRes.push("MOV AX, NUM1" + "\n");
this.codigoRes.push("MOV NUMIMP, AX" + "\n");
this.codigoRes.push("MOV DX, 0" + "\n");
this.codigoRes.push("MOV AX, NUMIMP" + "\n");
this.codigoRes.push("MOV BX, 10000" + "\n");
this.codigoRes.push("DIV BX" + "\n");
this.codigoRes.push("MOV RESIMP, AX" + "\n");
this.codigoRes.push("IMPNUM AL" + "\n");
this.codigoRes.push("MOV AX, RESIMP" + "\n");
this.codigoRes.push("MOV BX, 10000" + "\n");
this.codigoRes.push("MUL BX" + "\n");
this.codigoRes.push("MOV BX, AX" + "\n");
this.codigoRes.push("MOV AX, NUMIMP" + "\n");
this.codigoRes.push("SUB AX, BX" + "\n");
this.codigoRes.push("MOV NUMIMP, AX" + "\n");

this.codigoRes.push("MOV DX, 0" + "\n");
this.codigoRes.push("MOV AX, NUMIMP" + "\n");
this.codigoRes.push("MOV BX, 1000" + "\n");
this.codigoRes.push("DIV BX" + "\n");
this.codigoRes.push("MOV RESIMP, AX" + "\n");
this.codigoRes.push("IMPNUM AL" + "\n");
this.codigoRes.push("MOV AX, RESIMP" + "\n");
this.codigoRes.push("MOV BX, 1000" + "\n");
this.codigoRes.push("MUL BX" + "\n");
this.codigoRes.push("MOV BX, AX" + "\n");
this.codigoRes.push("MOV AX, NUMIMP" + "\n");
this.codigoRes.push("SUB AX, BX" + "\n");
this.codigoRes.push("MOV NUMIMP, AX" + "\n");

this.codigoRes.push("MOV DX, 0" + "\n");
this.codigoRes.push("MOV AX, NUMIMP" + "\n");
this.codigoRes.push("MOV BX, 100" + "\n");
this.codigoRes.push("DIV BX" + "\n");
this.codigoRes.push("MOV RESIMP, AX" + "\n");
this.codigoRes.push("IMPNUM AL" + "\n");
this.codigoRes.push("MOV AX, RESIMP" + "\n");
this.codigoRes.push("MOV BX, 100" + "\n");
this.codigoRes.push("MUL BX" + "\n");
this.codigoRes.push("MOV BX, AX" + "\n");
this.codigoRes.push("MOV AX, NUMIMP" + "\n");
this.codigoRes.push("SUB AX, BX" + "\n");
this.codigoRes.push("MOV NUMIMP, AX" + "\n");

this.codigoRes.push("MOV DX, 0" + "\n");
this.codigoRes.push("MOV AX, NUMIMP" + "\n");
this.codigoRes.push("MOV BX, 10" + "\n");
this.codigoRes.push("DIV BX" + "\n");
this.codigoRes.push("MOV RESIMP, AX" + "\n");
this.codigoRes.push("IMPNUM AL" + "\n");
```

```
    this.codigoRes.push("MOV AX, RESIMP" + "\n");
    this.codigoRes.push("MOV BX, 10" + "\n");
    this.codigoRes.push("MUL BX" + "\n");
    this.codigoRes.push("MOV BX, AX" + "\n");
    this.codigoRes.push("MOV AX, NUMIMP" + "\n");
    this.codigoRes.push("SUB AX, BX" + "\n");
    this.codigoRes.push("MOV NUMIMP, AX" + "\n");

    this.codigoRes.push("MOV DX, 0" + "\n");
    this.codigoRes.push("MOV AX, NUMIMP" + "\n");
    this.codigoRes.push("MOV BX, 1" + "\n");
    this.codigoRes.push("DIV BX" + "\n");
    this.codigoRes.push("MOV RESIMP, AX" + "\n");
    this.codigoRes.push("IMPNUM AL" + "\n");

    this.codigoRes.push("IMPCHAR 0AH" + "\n");
    this.codigoRes.push("IMPCHAR 0DH" + "\n");
    this.codigoRes.push("ENDM" + "\n");
    this.codigoRes.push("\n");
}

obtenerFunciones(rama, ambito) {
    if(rama.simbolo == "DefFunc") {
        ambito = rama.ramas[1].simbolo;
        let id = rama.ramas[1].simbolo;
        this.agregarFuncion(id);
        rama.ramas.forEach(subrama => {
            this.analizarFuncion(subrama, ambito);
        });
        this.codigoRes.push("\nENDM\n\n");
    }

    rama.ramas.forEach(subrama => {
        this.obtenerFunciones(subrama, ambito);
    });
}

analizarFuncion(rama, ambito) {
    if(rama.simbolo == "Parametros") {
        if(rama.ramas.length != 0) {
            let id = rama.ramas[1].simbolo;
            this.agregarParametro("", id, ambito);
        }
    }

    if(rama.simbolo == "ListaParam") {
        if(rama.ramas.length != 0) {
            let id = rama.ramas[2].simbolo;
            this.agregarParametro(", ", id, ambito);
        }
    }

    if(rama.simbolo == "Sentencia") {
        this.codigoRes.push("\n");
```

```
            this.obtenerSentencia(rama, ambito);
        }

        if(rama.simbolo != "Sentencia") {
            rama.ramas.forEach(subrama => {
                this.analizarFuncion(subrama, ambito);
            });
        }
    }

    agregarFuncion(id) {
        this.codigoRes.push(id + " MACRO ");
    }

    agregarParametro(coma, parametro, ambito) {
        this.codigoRes.push(coma + parametro + ambito);
    }

    obtenerSentencia(rama, ambito) {
        if(rama instanceof R21) {
            let id = rama.ramas[0].simbolo;
            if(this.globales.includes(id)) {
                id += "0";
            } else {
                id += ambito;
            }
            this.contadorOP = 0;
            let contenido = this.obtenerExpresion(rama.ramas[2],
ambito);
            if(contenido == "RESFUN" || contenido != "AX") {
                this.codigoRes.push("MOV BX, " + contenido + "\n");
                contenido = "BX";
            }
            this.codigoRes.push("MOV " + id + ", " + contenido +
"\n");
        }
        else if(rama instanceof R25) {
            this.contadorARG = 0;
            this.agregarLlamadaFunc(rama.ramas[0], ambito);
        }
        else if(rama instanceof R22) {
            let resultado = this.obtenerExpresion(rama.ramas[2],
ambito);
            let ifNum = this.contadorIf;
            this.contadorIf++;
            this.codigoRes.push("LOCAL FINIF" + ifNum + "\n");
            this.codigoRes.push("MOV AX, " + resultado + "\n");
            this.codigoRes.push("MOV BX, 0" + "\n");
            this.codigoRes.push("CMP AX, BX" + "\n");
            this.codigoRes.push("JE FINIF" + ifNum+ "\n");
            this.agregarSentenciaBloque(rama.ramas[4], ambito);

            this.codigoRes.push("FINIF" + ifNum+ ":\n");
            this.agregarOtro(rama.ramas[5], ambito);
```

```
        } else if(rama instanceof R23) {
            let whileNum = this.contadorWhile;
            this.contadorWhile++;
            this.codigoRes.push("LOCAL FINWHILE" + whileNum + "\n");
            this.codigoRes.push("LOCAL INICIOWHILE" + whileNum +
"\n");
            this.codigoRes.push("INICIOWHILE" + whileNum+ ":\n");
            let resultado = this.obtenerExpresion(rama.ramas[2],
ambito);

            this.codigoRes.push("MOV AX, " + resultado + "\n");
            this.codigoRes.push("MOV BX, 0" + "\n");
            this.codigoRes.push("CMP AX, BX" + "\n");
            this.codigoRes.push("JE FINWHILE" + whileNum+ "\n");
            this.obtenerSentencias(rama.ramas[4].ramas[1], ambito);
            this.codigoRes.push("JMP INICIOWHILE" + whileNum+ "\n");
            this.codigoRes.push("FINWHILE" + whileNum+ ":\n");

        } else if(rama instanceof R24) {
            this.codigoRes.push("MOV RESFUN, 0\n");
            if(rama.ramas[1].ramas.length != 0) {
                let contenido =
this.obtenerExpresion(rama.ramas[1].ramas[0], ambito);
                this.codigoRes.push("MOV AX, " + contenido + "\n");
                this.codigoRes.push("MOV RESFUN, AX");
            }
        } else if(rama instanceof R25) {
            this.contadorARG = 0;
            return this.agregarLlamadaFunc(rama.ramas[0], ambito);
        }
    }

    agregarSentenciaBloque(rama, ambito) {
        if(rama.ramas[0].simbolo == "Sentencia") {

            this.obtenerSentencia(rama.ramas[0], ambito);

        } else if(rama.ramas[0].simbolo == "Bloque"){
            this.obtenerSentencias(rama.ramas[0].ramas[1], ambito);
        }
    }

    obtenerSentencias(rama, ambito) {
        if(rama.ramas.length != 0) {
            this.obtenerSentencia(rama.ramas[0], ambito);
            this.obtenerSentencias(rama.ramas[1], ambito);
        }
    }

    agregarOtro(rama, ambito) {
        if(rama.ramas.length != 0) {
            this.agregarSentenciaBloque(rama.ramas[1], ambito);
        }
    }
```

```
    obtenerExpresion(rama, ambito) {
        if(rama instanceof R43)
            return this.obtenerExpresion(rama.ramas[1], ambito);

        else if(rama instanceof R44) {
            let contenido = this.obtenerExpresion(rama.ramas[1],
ambito);
            if(rama.ramas[0].simbolo == "+") {
                this.codigoRes.push("ADD AX, " + contenido + "\n");
            } else if(rama.ramas[0].simbolo == "-") {
                this.codigoRes.push("SUB AX, " + contenido + "\n");
            }
            this.codigoRes.push("MOV RES[" + this.contadorOP + "],
AX\n");
            contenido = "RES[" + this.contadorOP + "]";
            this.contadorOP += 2;
            return contenido;
        }

        else if(rama instanceof R45) {
            let contenido = this.obtenerExpresion(rama.ramas[1],
ambito);
            this.codigoRes.push("MOV AX, " + contenido + "\n");
            this.codigoRes.push("NEG AX" +"\n");
            this.codigoRes.push("MOV RES[" + this.contadorOP + "],
AX\n");
            contenido = "RES[" + this.contadorOP + "]";
            this.contadorOP += 2;
            return contenido;
        }


        else if(rama instanceof R46 || rama instanceof R47 || rama
instanceof R48
            || rama instanceof R49 || rama instanceof R50 || rama
instanceof R51) {
            let contenido1 = this.obtenerExpresion(rama.ramas[0],
ambito);
            if(contenido1 == "AX") {
                this.codigoRes.push("MOV RES[" + this.contadorOP + "],
AX\n");
                contenido1 = "RES[" + this.contadorOP + "]";
                this.contadorOP += 2;
            }
            let contenido2 = this.obtenerExpresion(rama.ramas[2],
ambito);
            if(contenido2 == "AX") {
                this.codigoRes.push("MOV RES[" + this.contadorOP + "],
AX\n");
                contenido2 = "RES[" + this.contadorOP + "]";
                this.contadorOP += 2;
            }
```

```
            let op = rama.ramas[1].simbolo;
            if(op == "+") {
                this.codigoRes.push("MOV AX, " + contenido1 + "\n");
                this.codigoRes.push("MOV BX, " + contenido2 + "\n");
                this.codigoRes.push("ADD AX, BX" + "\n");
                return "AX";

            } else if(op == "-") {
                this.codigoRes.push("MOV AX, " + contenido1 + "\n");
                this.codigoRes.push("MOV BX, " + contenido2 + "\n");
                this.codigoRes.push("SUB AX, BX" + "\n");
                return "AX";

            } else if(op == "*") {
                this.codigoRes.push("MOV AX, " + contenido1 + "\n");
                this.codigoRes.push("MOV BX, " + contenido2 + "\n");
                this.codigoRes.push("MUL BX" + "\n");
                return "AX";

            } else if(op == "/") {
                this.codigoRes.push("MOV DX, 0" + "\n");
                this.codigoRes.push("MOV AX, " + contenido1 + "\n");
                this.codigoRes.push("MOV BX, " + contenido2 + "\n");
                this.codigoRes.push("DIV BX" + "\n");
                return "AX";

            } else if(op == "<") {
                this.codigoRes.push("MENORVAR " + contenido1 + ", " +
contenido2 + "\n");
                this.codigoRes.push("MOV AX, RESGEN" + "\n");
                return "AX";

            } else if(op == ">") {
                this.codigoRes.push("MAYORVAR " + contenido1 + ", " +
contenido2 + "\n");
                this.codigoRes.push("MOV AX, RESGEN" + "\n");
                return "AX";

            } else if(op == "<=") {
                this.codigoRes.push("MENORIGUALVAR " + contenido1 + ",
" + contenido2 + "\n");
                this.codigoRes.push("MOV AX, RESGEN" + "\n");
                return "AX";

            } else if(op == ">=") {
                this.codigoRes.push("MAYORIGUALVAR " + contenido1 + ",
" + contenido2 + "\n");
                this.codigoRes.push("MOV AX, RESGEN" + "\n");
                return "AX";

            }else if(op == "==") {
                this.codigoRes.push("IGUALVAR " + contenido1 + ", " +
contenido2 + "\n");
                this.codigoRes.push("MOV AX, RESGEN" + "\n");
```

```
                return "AX";

            } else if(op == "!=") {
                this.codigoRes.push("NOIGUALVAR " + contenido1 + ", "
+ contenido2 + "\n");
                this.codigoRes.push("MOV AX, RESGEN" + "\n");
                return "AX";

            } else if(op == "||") {
                this.codigoRes.push("MOV AX, " + contenido1 + "\n");
                this.codigoRes.push("MOV BX, " + contenido2 + "\n");
                this.codigoRes.push("OR AX, BX" + "\n");
                return "AX";

            } else if(op == "&&") {
                this.codigoRes.push("MOV AX, " + contenido1 + "\n");
                this.codigoRes.push("MOV BX, " + contenido2 + "\n");
                this.codigoRes.push("AND AX, BX" + "\n");
                return "AX";

            }

        } else if(rama instanceof R52)
            return this.obtenerTermino(rama.ramas[0], ambito);
    }

    obtenerTermino(rama, ambito) {
        if(rama instanceof R35) {
            this.contadorARG = 0;
            return this.agregarLlamadaFunc(rama.ramas[0], ambito);
        }

        else if(rama instanceof R36) {
            if(this.globales.includes(rama.ramas[0].simbolo)) {
                return(rama.ramas[0].simbolo + "0");
            } else {
                return(rama.ramas[0].simbolo + ambito);
            }
        }

        else if(rama instanceof R37)
            return rama.ramas[0].simbolo;

        else if(rama instanceof R38)
            return Math.round(rama.ramas[0].simbolo);

        else if(rama instanceof R39) {
            return "'" + rama.ramas[0].simbolo.replace(new RegExp('"',
'g'), "") + "'";
        }
    }

    agregarLlamadaFunc(rama, ambito) {
        this.codigoRes.push("MOV RESFUN, 0\n");
```

```
        let id = rama.ramas[0].simbolo;
        let listaArgumentos = this.agregarArgumentos(rama.ramas[2],
ambito);
        this.codigoRes.push(id);
        for (let i = 0; i < listaArgumentos.length; i++) {
            if(i != 0) {
                this.codigoRes.push(", " + listaArgumentos[i]);
            } else {
                this.codigoRes.push(" " + listaArgumentos[i]);
            }
        }
        this.codigoRes.push("\n");
        return "RESFUN";
    }

    agregarArgumentos(argumentos, ambito) {
        let listaArgumentos = [];
        if(argumentos.ramas.length != 0) {
            let i = 0;
            if(argumentos.ramas[0] instanceof Otro)
                i++;

            this.contadorOP = 0;
            let expresion = this.obtenerExpresion(argumentos.ramas[i],
ambito);
            if(expresion == "AX") {
                this.codigoRes.push("MOV ARG[" + this.contadorARG +
"], AX\n");
                expresion = "ARG[" + this.contadorARG + "]";
                this.contadorARG += 2;
            }
            listaArgumentos.unshift(expresion);
            listaArgumentos = [...listaArgumentos,
...this.agregarArgumentos(argumentos.ramas[i + 1], ambito)]
        }
        return listaArgumentos;
    }

    agregarCodigoBase() {
        this.codigoRes.push(".CODE\n");
        this.codigoRes.push("CODEP PROC FAR\n");
        this.codigoRes.push("MOV AX,@DATA\nMOV DS,AX\n");
        this.codigoRes.push("main\n");

        this.imprimirVariables();

        this.codigoRes.push("CODEP ENDP\n");
    }

    imprimirVariables() {
        this.variables.forEach(variable => {
            let nombre = variable.id;
            if(variable.ambito == "*") {
                nombre += "0";
```

```
            } else {
                nombre += variable.ambito;
            }
            nombre.split("").forEach(caracter => {
                this.codigoRes.push("IMPCHAR '" + caracter + "'" +
"\n");
            });
            this.codigoRes.push("IMPCHAR ':'" + "\n");
            this.codigoRes.push("IMPCHAR ' '" + "\n");
            if(variable.tipo == "int" || variable.tipo == "float" ||
variable.tipo == "void") {
                this.codigoRes.push("IMPRIMIRINT " + nombre + "\n");
            }
            else if(variable.tipo == "char") {
                this.codigoRes.push("IMPRIMIRCHAR " + nombre + "\n");
            }

        });
    }

    analisis() {
        this.arbol.raiz.forEach(rama => {
            this.siguienteNodo(rama, "*");
        });
    }

    siguienteNodo(rama, ambito) {

        rama.ramas.forEach(subrama => {
            this.siguienteNodo(subrama, ambito);
        });
    }


    crearDocumento() {

        const blob = new Blob(this.codigoRes, { type: 'text/plain' });

        return blob;
    }
};

export {GenCode};
```

“Lexico.js”

```
class Lexico {

    constructor(fuente) {
        this.fuente = fuente || "$";
        this.indice = 0;
        this.continua = true;
        this.caracter = "";
        this.simbolo = "";
        this.tipo = 404;
        this.tipoS = "";
        this.estado = 0;
    }

    reiniciar(fuente) {
        this.indice = 0;
        this.fuente = fuente;
        this.estado = 0;
    }

    //Automata
    obtenerSimbolo() {
        this.simbolo = "";
        this.continua = true;
        this.tipo = 404;
        this.tipoS = "";
        this.estado = 0;

        while(this.continua) {
            this.caracter = this.sigCaracter();
            switch (this.estado) {
                case 0:
                    if(this.esLetra()) {
                        this.estado = 1;
                        this.simbolo += this.caracter;
                        break;
                    }
```

```
if(this.esDigito()) {
    this.estado = 2;
    this.simbolo += this.caracter;
    break;
}

if(this.esOpSum()) {
    this.estado = 5;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
    break;
}

if(this.esOpMul()) {
    this.estado = 6;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
    break;
}

if(this.esOpRel()) {
    this.estado = 7;
    this.simbolo += this.caracter;
    break;
}

if(this.caracter == "|") {
    this.estado = 8;
    this.simbolo += this.caracter;
    break;
}

if(this.caracter == "&") {
    this.estado = 9;
    this.simbolo += this.caracter;
    break;
}

if(this.caracter == "!") {
    this.estado = 10;
    this.simbolo += this.caracter;
    break;
}

if(this.caracter == "=") {
    this.estado = 18;
    this.simbolo += this.caracter;
    break;
}

if(this.caracter == ";") {
    this.estado = 12;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
```

```
    break;
}

if(this.caracter == ",") {
    this.estado = 13;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
    break;
}

if(this.caracter == "(") {
    this.estado = 14;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
    break;
}

if(this.caracter == ")") {
    this.estado = 15;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
    break;
}

if(this.caracter == "{") {
    this.estado = 16;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
    break;
}

if(this.caracter == "}") {
    this.estado = 17;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
    break;
}

if(this.caracter == '"') {
    this.estado = 19;
    this.simbolo += this.caracter;
    break;
}

if(this.caracter == "$") {
    this.estado = 23;
    this.simbolo += this.caracter;
    this.aceptacion(this.estado);
    break;
}

if(this.espacio()) {
    break;
}
```

```
    if(this.termina()) {
        this.aceptacion(this.estado);
        this.simbolo += this.caracter;
        break;
    }

    this.simbolo += this.caracter;
    this.estado = 404;
    this.aceptacion(this.estado);

    break;

case 1:
    if(this.esLetra() || this.esDigito()) {
        this.simbolo += this.caracter;
        break;
    } else {
        this.estado = this.esReservada();
        this.regresa();
        this.aceptacion(this.estado);
        break;
    }

case 2:
    if(this.esDigito()) {
        this.simbolo += this.caracter;
        break;
    } else if (this.caracter == "."){
        this.estado = 3;
        this.simbolo += this.caracter;
        break;
    } else {
        this.regresa();
        this.aceptacion(this.estado);
        break;
    }

case 3:
    let num;
    if(this.esDigito()) {
        this.simbolo += this.caracter;
        break;
    }
    else if(this.termina()) {
        num = this.indice - 1;
    }
    else {
        num = this.indice -2;
    }

    if(this.fuente[num] == ".") {
        this.estado = 404;
    }
```

```
    this.regresa();
    this.aceptacion(this.estado);
    break;

case 7:
    if(this.caracter == "=") {
        this.simbolo += this.caracter;
        this.aceptacion(this.estado);
        break;
    } else {
        this.regresa();
        this.aceptacion(this.estado);
    }
    break;

case 8:
    if(this.caracter == "|") {
        this.simbolo += this.caracter;
    } else {
        this.estado = 404;
        this.regresa();
    }
    this.aceptacion(this.estado);
    break;

case 9:
    if(this.caracter == "&") {
        this.simbolo += this.caracter;
    } else {
        this.estado = 404;
        this.regresa();
    }
    this.aceptacion(this.estado);
    break;

case 10:
    if(this.caracter == "=") {
        this.simbolo += this.caracter;
        this.estado = 11;
    } else {
        this.regresa();
    }
    this.aceptacion(this.estado);
    break;

case 18:
    if(this.caracter == "=") {
        this.simbolo += this.caracter;
        this.estado = 11;
    } else {
        this.regresa();
    }
    this.aceptacion(this.estado);
    break;
```

```
            case 19:
                if(this.esLetra() || this.esDigito()) {
                    this.simbolo += this.caracter;
                    break;
                } else if(this.caracter == '"') {
                    this.simbolo += this.caracter;
                    this.estado = 24;
                    this.aceptacion(this.estado);
                    break;
                }
                this.estado = 404;
                this.aceptacion(this.estado);
                break;

            default:
                break;
        }
    }

    switch (this.estado) {
        case 404:
            this.tipo = 404;
            this.tipoS = "Error"
            break;

        case 0:
            this.tipo = 100;
            this.tipoS = "Vacio";
            break;

        case 1:
            this.tipo = 0;
            this.tipoS = "Id";
            break;

        case 2:
            this.tipo = 1;
            this.tipoS = "Entero";
            break;

        case 3:
            this.tipo = 2;
            this.tipoS = "Real";
            break;

        case 4:
            this.tipo = 4;
            this.tipoS = "Tipo";
            break;

        case 5:
            this.tipo = 5;
            this.tipoS = "OpSum";
```

```
    break;

case 6:
    this.tipo = 6;
    this.tipoS = "OpMul";
    break;

case 7:
    this.tipo = 7;
    this.tipoS = "OpRelac";
    break;

case 8:
    this.tipo = 8;
    this.tipoS = "OpOr";
    break;

case 9:
    this.tipo = 9;
    this.tipoS = "OpAnd";
    break;

case 10:
    this.tipo = 10;
    this.tipoS = "OpNot";
    break;

case 11:
    this.tipo = 11;
    this.tipoS = "OpIgualdad";
    break;

case 18:
    this.tipo = 18;
    this.tipoS = "=";
    break;

case 12:
    this.tipo = 12;
    this.tipoS = ";";
    break;

case 13:
    this.tipo = 13;
    this.tipoS = ",";
    break;

case 14:
    this.tipo = 14;
    this.tipoS = "(";
    break;

case 15:
    this.tipo = 15;
```

```
            this.tipoS = ")";
            break;

        case 16:
            this.tipo = 16;
            this.tipoS = "{";
            break;

        case 17:
            this.tipo = 17;
            this.tipoS = "}";
            break;

        case 19:
            this.tipo = 19;
            this.tipoS = "if";
            break;

        case 20:
            this.tipo = 20;
            this.tipoS = "while";
            break;

        case 21:
            this.tipo = 21;
            this.tipoS = "return";
            break;

        case 22:
            this.tipo = 22;
            this.tipoS = "else";
            break;

        case 23:
            this.tipo = 23;
            this.tipoS = "$";
            break;

        case 24:
            this.tipo = 3;
            this.tipoS = "Cadena";
            break;

        default:
            break;
    }
}

sigCaracter () {
    if(this.indice == this.fuente.length) {
        return "$";
    }
    return this.fuente[this.indice++];
}
```

```
    regresa() {
        if (this.caracter != "$" && !this.espacio()) {
            this.indice--;
        }
        if(this.caracter == "$") {
            this.caracter = "";
        }
    }


    esLetra() {
        if (this.caracter.match(/[a-z]/i)) {
            return true;
        }
        return false;
    }

    esDigito() {
        if (this.caracter.match(/[0-9]/)) {
            return true;
        }
        return false;
    }

    esReservada() {
        if (this.simbolo == "int" || this.simbolo == "float" ||
this.simbolo == "void" || this.simbolo == "char") {
            return 4;
        } else if(this.simbolo == "if") {
            return 19;
        } else if(this.simbolo == "while") {
            return 20;
        } else if(this.simbolo == "return") {
            return 21;
        } else if(this.simbolo == "else") {
            return 22;
        }
        return 1;
    }

    esOpSum() {
        if (this.caracter.match(/[+-]/)) {
            return true;
        }
        return false;
    }

    esOpMul() {
    if (this.caracter.match(/[*/]/)) {
            return true;
        }
        return false;
    }
```

```
    esOpRel() {
    if (this.caracter.match(/[<>]/)) {
            return true;
        }
        return false;
    }

    espacio() {
        return this.caracter == " " || this.caracter == "\n";
    }

    termina() {
        return this.caracter == "$";
    }

    aceptacion(estado) {
        this.continua = false;
    }

}

export {Lexico};
```

"Pila.js"

```
import * as pilaElem from "./elementoPila.js";

class Pila {

    constructor() {
        this.pila = [];
    }

    push(element, tipo) {
        let elemento;
        switch(tipo) {
            case 1:
                elemento = new pilaElem.Terminal(element);
                break;

            case 2:
                elemento = new pilaElem.NoTerminal(element);
                break;

            case 3:
                elemento = new pilaElem.Estado(element);
                break;
        }
        this.pila.push(elemento);
    }

    front() {
```

```
        return this.pila[this.pila.length -1].value();
    }

    pop() {
        return this.pila.pop();
    }

    vaciar() {
        this.pila = [];
    }

    toString() {
        let string = "";
        this.pila.forEach(element => {
            string += element.string();
        });
        return string;
    }
}

export {Pila};
```

"ElementosPila.js"

```
export class ElementoPila {
    constructor() { }

    string() {}
    value() {}
}

export class Estado extends ElementoPila {

    constructor (elemento) {
        super();
        this.elemento = elemento || 0;
    }

    string() {
        return this.elemento.toString();
    }

    value() {
        return this.elemento;
    }
}

export class NoTerminal extends ElementoPila {

    constructor (elemento) {
```

```
        super();
        this.elemento = elemento || "";
    }

    string() {
        return this.elemento;
    }

    value() {
        return this.elemento;
    }
}

export class Terminal extends ElementoPila {

    constructor (elemento) {
        super();
        this.elemento = elemento || "";
    }

    string() {
        return this.elemento;
    }

    value() {
        return this.elemento;
    }
}
```