UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS

ESTRUCTURAS DE DATOS

MANUAL TECNICO

Nombre: Joseph Jeferson Marroquín Monroy
Carné: 202010316

Manual Técnico

Lugar, fecha y responsables de la elaboración

El Proyecto 2 fue desarrollado entre el 6 y el 29 de abril del 2022, elaborado por Joseph Jeferson Marroquin Monroy.

Objetivos

El presente manual tiene como finalidad describir la funcionalidad del algoritmo aplicado para realizar el análisis léxico, sintáctico y semántico de un interprete.

Configuración del sistema

El proyecto 2 está desarrollado en una Laptop Lenovo Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz con 8GB de RAM, Sistema operativo de 64 bits, procesador x64.

```
1   class Simbolo{
2       constructor(nombre,tipo,valor,parametros,linea,columna){
3           this.nombre=nombre;
4           this.tipo=tipo;
5           this.valor=valor;
6           this.parametros=parametros;
7           this.linea=linea;
8           this.columna=columna;
9       }
10  }
```

Creamos un constructor donde vamos a guardar todos los datos de nuestra tabla de símbolos.

```
//Insertar un nuevo simbolo
insertar(simbolos){
    this.simbolos.push(simbolos)
}
```

```
obtener(nombre){
    let res=null;
    this.simbolos.forEach(simbolo=>{
        if(simbolo.nombre==nombre){
```

```
modificar(simbol){
    this.simbolos.forEach(simbolo=>{
        if(simbolo.nombre==simbol.nombre){
```

Creamos las funciones insertar, obtener y modificar para acceder a los datos de la tabla de símbolos.

```
class N_Error{
    constructor(tipo,descripcion,fila,columna){
        this.tipo=tipo;
        this.descripcion=descripcion;
        this.fila=fila
        this.columna=columna
        this.siguiente=null;
        this.anterior=null;
    }
}
```

Creamos un constructor donde vamos a guardar los errores producidos en la ejecución.

```
insertar(Error){
    if(this.principio==null){
        this.principio=Error;
        this.fin=Error;
        return;
    }
    this.fin.siguiente=Error;
    Error.anterior=this.fin;
    this.fin=Error;

}
```

Creamos insertar para agregar nuevos errores encontrados.

```
// VAMOS A INTERPRETAR
function final(id,consola){
    consola.value='';
    L_Error.getInstance().reiniciar();
    TS.getInstance().reiniciar();
    try{
      var resultado= gramatica.parse(id.getValue().toLowerCase());

      console.log(imprimir(resultado));
      UpdateGraphviz(imprimir(resultado));
      var interprete = new Interprete();
      var metodos = new Metodos();
      var texto= metodos.analizar(resultado);
      texto+= interprete.analizar(resultado);
     //download(texto, 'Analisis.txt', 'text/plain')
      consola.value=texto;
      return;
    }catch(error){
       consola.value=error+"\n"+L_Error.getInstance().getErrores2();
       console.log(error);
       //download(error,'Archivo con errores','text/plain')
       return;
    }
}
```

Creamos una función donde mandamos a interpretar todo lo ingresado en la consola y así ejecutarlo.

```
case "DECLARACION":
      raiz.childs[0].childs.forEach(hijo=>{

            if(raiz.childs[1]=="int"){
                simbolo= new Simbolo(hijo.value,"integer",0,"",raiz.childs[0].fila,raiz.childs[0].columna);
            }
            else if(raiz.childs[1]=="double"){
                simbolo= new Simbolo(hijo.value,"double","0.0","",raiz.childs[0].fila,raiz.childs[0].columna);
            }
            else if(raiz.childs[1]=="boolean"){
                simbolo= new Simbolo(hijo.value,"boolean",true,"",raiz.childs[0].fila,raiz.childs[0].columna);
            }
            else if(raiz.childs[1]=="string"){
                simbolo= new Simbolo(hijo.value,"string","","",raiz.childs[0].fila,raiz.childs[0].columna);
            }
            else if(raiz.childs[1]=="char"){
                simbolo= new Simbolo(hijo.value,"char",'\u0000',"",raiz.childs[0].fila,raiz.childs[0].columna);
            }
            TS.getInstance().insertar(simbolo)

      })
      break;
```

Realizamos la función de declarar verificando que tipo de dato se ingreso, agregándolo a la tabla de símbolos.

```
case "ASIGNACION":
        raiz.childs[0].childs.forEach(hijo=>{
            simbolo=TS.getInstance().obtener(hijo.value);
            if(simbolo.tipo=="integer"){
                op = new Operador()
                res = op.ejecutar(raiz.childs[1])
                if(res.tipo=="integer"){
                    simbolo.tipo=res.tipo;
                    simbolo.valor=res.valor;
                    TS.getInstance().modificar(simbolo)
                }else{
                    L_Error.getInstance().insertar(new N_Error("Semantico","El valor asignado no corresponde a
                    codigo="Error Semantico"+" El valor asignado no corresponde a un entero "+" fila: "+raiz.c
                }
            }
```

Asignamos un valor a la variable que se mandó, verificando que corresponda a su tipo de dato.

```
case "METODO_SIN_PA":
    interprete=new Interprete();
    let arregloInstrucciones=[];

    //DECLARANDO
    if(TS.getInstance().obtener(raiz.childs[0])==null){
        simbolo= new Simbolo(raiz.childs[0],"metodo","","",raiz.childs[0].fila,raiz.childs[0].columna);
        TS.getInstance().insertar(simbolo)
        raiz.childs[1].childs[0].childs.forEach(nodito => {
            //interprete.analizaMetodo("si");
            arregloInstrucciones.push(nodito)
            //simbolo.valor+=this.interpretar(nodito);
            //simbolo.valor+=interprete.interpretar(nodito);
            //interprete.analizaMetodo(null);
        });
        simbolo.valor=arregloInstrucciones;
        TS.getInstance().modificar(simbolo)
```

Declaramos nuestro método verificando que no exista otro igual, guardando sus instrucciones para ejecutarlas cuando llamen al método.

```
case "entero":
    Resultado= new ResultadoOp();
    Resultado.tipo="integer";
    Resultado.valor=parseInt(raiz.value);
    return Resultado

case "decimal":
    Resultado= new ResultadoOp();
    Resultado.tipo="double";
    Resultado.valor=parseFloat(raiz.value)
    return Resultado

case "true":
        Resultado= new ResultadoOp();
        Resultado.tipo="boolean";
        Resultado.valor=true;
        return Resultado;


case "false":
        Resultado= new ResultadoOp();
        Resultado.tipo="boolean";
        Resultado.valor=false;
        return Resultado;
```

Asignamos el valor y tipo de una expresión.

```
case "PRINT":
    op = new Operador();
    res = op.ejecutar(raiz.childs[0]);
    codigo+=res.valor
    return codigo;

case "PRINTLN":
    op = new Operador();
    res = op.ejecutar(raiz.childs[0]);
    codigo+="\n"+res.valor+"\n"
    return codigo;
```

Ejecutamos la expresión a evaluar y así imprimir su valor.

```
case "IF":
    metodos=new Metodos();
    op = new Operador();
    res=op.ejecutar(raiz.childs[0])

    if(res.tipo=="boolean"){
        if(res.valor){
            raiz.childs[1].childs[0].childs.forEach(nodito => {
              codigo+=metodos.interpretar(nodito);
              codigo+=this.interpretar(nodito);
            });
            return codigo;
        }else{
```

Para el caso de if verificamos que se cumpla la condición para ejecutar las instrucciones que contenga.

```
case "SWITCH":
    metodos=new Metodos();
    op = new Operador();
    res=op.ejecutar(raiz.childs[0])
    var BreakException = {};

    if(raiz.childs.length==2){
        raiz.childs[1].childs.forEach(nodito => {
            switchcase=op.ejecutar(nodito.childs[0])
            switch(res.valor){
                case switchcase.valor:
                    if(almacenaBreak=="Si"){
                        almacenaBreak=null;
                        break;
                    }
                    try{
                        nodito.childs[1].childs.forEach(hh => {
                            codigo+=metodos.interpretar(hh);
                            codigo+=this.interpretar(hh);
                            if(almacenaBreak=="Si"){
                                throw BreakException;
                            }
                        });
                    }catch (e) {
                        if (e !== BreakException) throw e;
                      }
                    return codigo;
            }
```

Para el caso de switch recorremos los case que contenga para verificar si se cumple, si no se deja el valor default.

```
case "LLAMADA_MSIN_PA":
    metodos=new Metodos();
    var BreakException = {};

    //DECLARANDO
    if(TS.getInstance().obtener(raiz.childs[0])==null){
        simbolo= new Simbolo(raiz.childs[0],"metodo","","",raiz.childs[0].fila,raiz.childs[0].columna);
        TS.getInstance().insertar(simbolo)
    }else{
        simbolo=TS.getInstance().obtener(raiz.childs[0]);
        if(typeof simbolo.valor=='object'){
            try{
            simbolo.valor.forEach(ins =>{
                codigo+=metodos.interpretar(ins);
                codigo+=this.interpretar(ins);
                if(almacenaReturn=="Si"){
                    throw BreakException;
                }
            });
        }catch (e) {
```

En el caso de llamadas, buscamos el método con sus instrucciones para poder ejecutarlas.