MANUAL TECNICO

Para utilizar el servidor es necesario tener instalado nodejs (para el comando npm), se debe instalar de la página oficial nodejs.org/es/ :

El servidor se creó con los siguientes comandos:

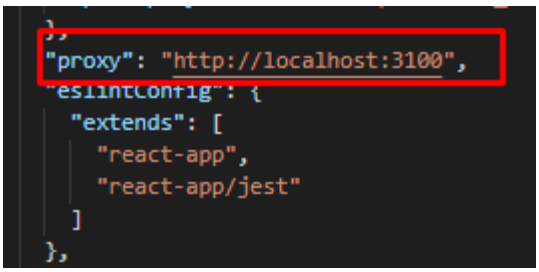
npm install

npm start

Se utilizó el framework Reactjs, se utilizaron los siguientes comandos:

npm create-react-app servidor

En servidor/client/package.json se hizo la conexión al backend agregando el script.



La clase routes.ts es donde indicamos las peticiones que tendrá nuestro servidor:

```
router.post('/ejecutar',function(req, res){
    try{
        const {input} = req.body;

        let ast : Ast = interprete.parse(input);

        let respuesta = "";

        let controlador = new Controlador();
        let ts_global = new TablaSimbolos(null);

        ast.ejecutar(controlador,ts_global);
        let ts_html = controlador.graficar_ts(controlador,ts_global,"1");

        for(let tablitas of controlador.tablas){
            ts_html += controlador.graficar_ts(controlador,tablitas,"2")
        }

        /*for(let evaluar of arreglo){
            let valor = evaluar.expresion.getValor(controlador,ts_global);

            if(valor != null){
                console.log(`El valor de la expresion es : ${valor}`);
                respuesta += `El valor de la expresion es : ${valor} \n` ;
            }else{
                console.log(`El valor de la expresion es : ERROR`);
                respuesta += `El valor de la expresion es : ERROR \n` ;
            }
        }*/



        res.status(200).json({consola : controlador.consola, ts:ts_html});

    }catch(error){


        res.status(500).json({resultado : "Se ha producido un error"})
    }
})
```

```
router.post('/recorrer',function(req, res){
    try{        const input: any
        const {input} = req.body;
        console.log(input);
        let ast:Ast = interprete.parse(input);
        let nodo_ast : Nodo = ast.recorrer();
        let grafo = nodo_ast.GraficarSintactico();
        res.status(200).json({ast :grafo})

    }catch(error){
        console.log(error);
        res.status(500).json({resultado : "Se ha producido un error"})
    }
})

module.exports = router;
```

Se creó una clase AST para manipular el flujo de nuestro programa que tiene 3 pasadas, en la primera se guardan las funciones y métodos, la segunda ejecuta las declaraciones y por último se ejecutan todas las otras

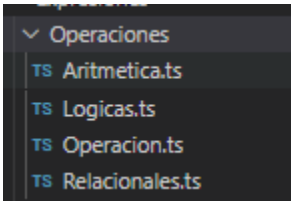
```
export default class Ast implements Instruccion{
    public lista_instrucciones : Array<Instruccion>;

    constructor(lista_instrucciones : Array<Instruccion>){
        this.lista_instrucciones = lista_instrucciones;
    }

    ejecutar(controlador: Controlador, ts: TablaSimbolos){
        let bandera_start = false;
        //1era pasada vamos a guardar las funciones y métodos del programa

        for(let instruccion of this.lista_instrucciones){
            if(instruccion instanceof Funcion){
                let funcion = instruccion as Funcion;
                funcion.agregarFuncionTS(ts);
            }
        }


        //Vamos a recorrer las instrucciones que vienen desde la gramática



        //2da pasada. Se ejecuta las declaraciones de variables
        for(let instruccion of this.lista_instrucciones){
            if(instruccion instanceof Declaracion){
                instruccion.ejecutar(controlador,ts);
            }
        }
        //era pasada ejecutamos las demás instrucciones
        for(let instruccion of this.lista_instrucciones){
            if(instruccion instanceof StartWith && !bandera_start){
                instruccion.ejecutar(controlador,ts);
                bandera_start = true;
            }else if(bandera_start){
                let error = new Errores("Semantico",`Solo se puede colocar un startwith.`,0,0);
                controlador.errores.push(error);
                controlador.append(`ERROR: Semántico, Solo se puede colocar un startwith.`);
                console.log("no se puede");
            }



            if(!(instruccion instanceof Declaracion) && !(instruccion instanceof Funcion) && !(instruccion instanceof StartWith)){
                instruccion.ejecutar(controlador,ts);
            }


        }
```

SE creó la clase expresión la cual contiene dos métodos, getTipo y getValor

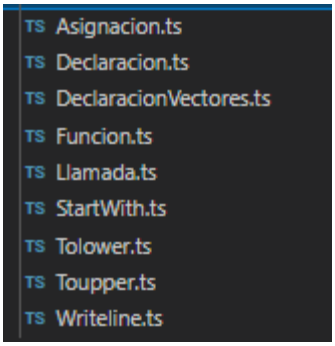
```typescript
export interface Expresion{

    /**
     * @function getTipo nos devuelve el tipo del valor de la expresion
     * @param controlador llevamos todo el control del programa
     * @param ts accede a la tabla de símbolos
     */

    getTipo(controlador : Controlador,ts: TablaSimbolos): tipo;
    /**
     * @function getValor nos devuelve el valor de la expresion
     * @param controlador llevamos todo el control del programa
     * @param ts accede a la tabla de símbolos
     */


    getValor(controlador : Controlador,ts: TablaSimbolos): any;

    recorrer() : Nodo;
}
```

SE creó la clase Instrucción la cual contiene un método. Ejecutar, la clase expresión y la clase instrucción indican que hacer con cada clase que extienda de estas.

Se crearon varias clases por cada operación que admite el programa, todas estas extienden de Expresión

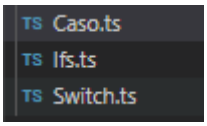


Se crearon varias clases para cada instrucción, estas extienden de instrucción

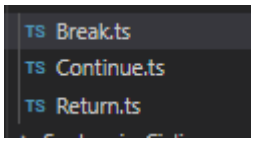


Se crearon varias clases para las sentencias cíclicas, extienden de instrucción

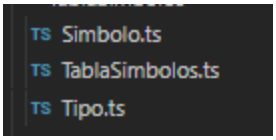
Se crearon varias clases para las sentencias de control, extienden de instrucción



Se crearon varias clases para las sentencias de control, extienden de instrucción



Para manejar la tabla de símbolo se creo una clase símbolo. Y en la clase tabla de símbolos se crearon métodos (agregar,exits,getsimbolo,existenactual) para manipular los datos:




```
export default class TablaSimbolos{
    public ant: TablaSimbolos;
    public tabla: Map<string,Simbolo>;

    //en la tabla vamos a ir guardando el nombre y todo lo que tiene
    //x , (x,0,entero)
    //y , (y,0,entero)
    //z , (z,0,entero)

    /**
     * @constructor creamos una nueva tabla.
     * @param ant indica cual es la tabla de simbolos anterior de la nueva tabla que nos servirá para le manejo de ambitos
     * Le mandamos una tabla global y otra local
     */

    constructor(ant : TablaSimbolos | any){
        this.ant = ant;
        this.tabla = new Map<string,Simbolo>();
    }

    agregar(id: string, simbolo: Simbolo){
        this.tabla.set(id.toLowerCase(),simbolo); //usamos todo minúscula porque nuestro lenguaje es caseinsensitive
    }

    existe(id: string): boolean{  // Con esto buscamos si existe la variable
        let ts: TablaSimbolos = this;

        while(ts != null){
            let existe = ts.tabla.get(id.toLowerCase());
            if(existe != null){
                return true;
            }
            ts = ts.ant
        }
        return false;
    }
```

```
getSimbolo(id: string){
    let ts: TablaSimbolos = this;

    while(ts != null){
        let existe = ts.tabla.get(id.toLowerCase());
        if(existe != null){
            return existe;
        }
        ts = ts.ant
    }
    return null;
}

existeEnActual(id:string):boolean{
    let ts: TablaSimbolos = this;
    let existe = ts.tabla.get(id.toLowerCase());
    if(existe != null){
        return true;
    }
    return false;

}
```