Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Laboratorio Organización de Lenguajes y Compiladores 1

Proyecto 2: CompiScript+

Manual Técnico

Nombre: Josué Daniel Rojché García

Carné: 201901103

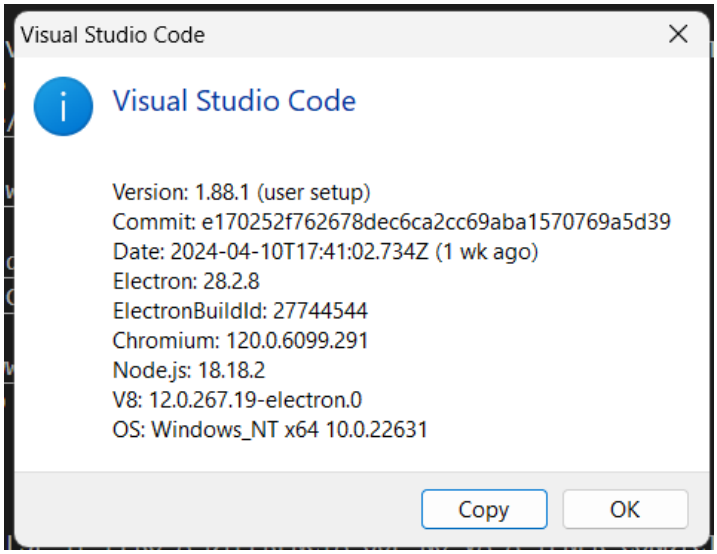
Fecha: 21/04/2024

Sección: N

Auxiliar: Walter Alexander Guerra Duque

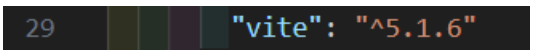
Requerimientos del Sistema

En la realización del software el Editor de código utilizado fue VisualStudioCode, con las especificaciones que se observan en la siguiente imagen.



Para la realización del analizador se utilizó jison como herramienta, para la realización del backend y el manejo de rutas se utilizaron express y cors.

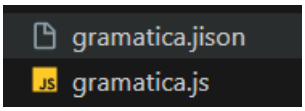
Para la realización del frontend, se utilizó react desde vite



La versión de nodejs utilizada es `v20.11.1`

Analizador Léxico y Sintáctico

Para conseguir la solución del programa se realizó analizador léxico, para el reconocimiento de archivos de tipo .sc, por lo que se utiliza las librerías mencionadas anteriormente. En la siguiente imagen podrá observar el paquete para el analizador, con sus respectivos archivos.



Dentro del archivo gramatica.jison se puede observar los caracteres y expresiones regulares y palabras reservadas que serán reconocidos por el lenguaje, así como el manejo de errores léxicos que puedan ocurrir.

```
%lex

%options case-insensitive

%%

\s+                 //espacios en blanco
"//".*          //comentario simple
[/][*][^*]*[*]+([^/*][^*]*[*]+)*[/]  //comentario vlineas


"++"                return 'sigincremento';
"--"                return 'sigdecremento';
"+"                 return 'mas';
"-"                 return 'menos';
"/"                 return 'dividir';
"*"                 return 'por';
"%"                 return 'modulo';
"!="                return 'negacionigual';
"!"                 return 'notlogico';
"("                 return 'parentesisabre';
")"                 return 'parentesiscierra';
"{"                 return 'llaveabre';
"}"                 return 'llavecierra';
"["                 return 'corcheteabre';
"]"                 return 'corchetecierra';
","                 return 'signocoma';
";"                 return 'sigpuntoycoma';
"=="                return 'igualigual';
"="                 return 'sigigual';
"."                 return 'sigpunto';
```

```
72    "."                   return 'sigpunto';
73    "?"                   return 'siginterrogacion';
74    ":"                   return 'dospuntos';
75    "<="                  return 'menorigual';
76    ">="                  return 'mayorigual';
77    "<<"                  return 'menormenor';
78    "<"                   return 'menorque';
79    ">"                   return 'mayorque';
80    "||"                  return 'orlogico';
81    "&&"                  return 'andlogico';
82    "int"                 return 'resint';
83    "double"              return 'resdouble';
84    "bool"                return 'resbool';
85    "char"                return 'reschar';
86    "std::string"         return 'resstring';
87    "pow"                 return 'respotencia';
88    "new"                 return 'resnew';
89    "if"                  return 'resif';
90    "else"                return 'reselse';
91    "switch"              return 'resswitch';
92    "case"                return 'rescase';
93    "default"             return 'resdefault';
94    "while"               return 'reswhile';
95    "for"                 return 'resfor';
96    "do"                  return 'resdo';
97    "break"               return 'resbreak';
98    "continue"            return 'rescontinue';
99    "return"              return 'resreturn';
100   "void"                return 'resvoid';
101   "cout"                return 'rescout';
102   "endl"                return 'resendl';
103   "tolower"             return 'restolower';
104   "toupper"             return 'restoupper';
```

```
104   "toupper"           return 'restoupper';
105   "round"             return 'resround';
106   "length"            return 'reslength';
107   "typeof"            return 'restypeof';
108   "std::tostring"     return 'restostring';
109   "c_str"             return 'rescstr';
110   "execute"           return 'resexecute';
111
112
113   ("true"|"false")\b      return 'bool';
114   [0-9]+[.][0-9]+\b       return 'decimal';
115   [0-9]+\b                return 'numero';
116   ([a-zA-Z])[a-zA-Z0-9_]* return 'id';
117   \'[^\']\'               { yytext = yytext.substr(1,yyleng-2); return 'caracter';}
118   \"[^\"]*\"              { yytext = yytext.substr(1,yyleng-2); return 'cadena'; }
119
120   <<EOF>>             return 'EOF';
121   //para agregar los errores a la consola de salida, agregarlo a la lista de impresion que se encuentra en print global.obimpresiones desde la cl
122   . {addError('Error léxico', 'Caracter no reconocido\" ' + yytext +' \" ', yylloc.first_line, yylloc.first_column); console.error('Error léxico:
123   /lex
```

Dentro del archivo se tiene el manejo de errores sintácticos, la gramática que servirá para verificar la correcta lectura del archivo y la declaración de la precedencia de operaciones, también se realiza el manejo de la lógica para almacenar lo reconocido, como declaraciones, impresiones, operaciones, etc.

```
%left 'orlogico'
%left 'andlogico'
%right 'notlogico'
%left 'negacionigual' 'igualigual'
%left 'menorigual' 'mayorigual' 'mayorque' 'menorque'
%left 'mas', 'menos'
%left 'dividir', 'por','modulo'
%left 'pow'
%right Umenos

%start INI

%%
```

```
%%

INI: CODIGO EOF                         {$$ = $1; return $$;}
;

CODIGO: CODIGO INSTRUCCION              { $$ = $1; $$.push($2);}
        |INSTRUCCION                    { $$ = []; $$.push($1);}
;

INSTRUCCION: DECLARACIONESARR        {console.log($1);$$=$1;}
        | DECLARACIONES              {console.log($1);$$=$1;}
        | SENTENCIAS                 {console.log($1); $$=$1;}
        | FUNCIONES                  {console.log($1); $$=$1;}
        | METODOS                    {console.log($1); $$=$1;}
        | LLAMADAS sigpuntoycoma     {console.log($1); $$=$1;}
        | FCOUT                      {console.log($1); $$=$1;}
        | FEXECUTE                   {console.log($1); $$=$1;}
;

DECLARACIONES: TIPODATO LISTANVARIABLES sigpuntoycoma                   { $$= new Asignacion($2, new Dato("sindato", $1, @1.first_line, @1.first_column), $1, @1.first_
        | TIPODATO LISTANVARIABLES sigigual ASIGNACIONES sigpuntoycoma   { $$= new Asignacion($2, $4, $1, @1.first_line, @1.first_column); limpiarlistVariables();}
        | LISTANVARIABLES sigigual ASIGNACIONES sigpuntoycoma            { $$= new Reasignacion($1, $3, @1.first_line, @1.first_column); limpiarlistVariables();}
        | id sigincremento sigpuntoycoma                                 { $$= new IncrementoDecremento($1,new Oid($1, "id", @1.first_line, @1.first_column, "id"),"++",
        | id sigdecremento sigpuntoycoma                                 { $$= new IncrementoDecremento($1,new Oid($1, "id", @1.first_line, @1.first_column, "id"),"--",
        | error sigpuntoycoma                                            { addError('Error sintáctico', 'No se reconoce' + $1, this._$.first_line, this._$.first_column)
;
DECLARACIONESARR: TIPODATO LISTANVARIABLES corcheteabre corchetecierra sigigual resnew TIPODATO corcheteabre EXPRESIONES corchetecierra sigpuntoycoma     { $$= new Asign
        | TIPODATO LISTANVARIABLES corcheteabre corchetecierra sigigual corcheteabre LISTANEXPR corchetecierra sigpuntoycoma                          { $$= new Asign
        | TIPODATO LISTANVARIABLES corcheteabre corchetecierra corcheteabre corchetecierra sigigual resnew TIPODATO corcheteabre EXPRESIONES corchetecierra corcheteabre
        | TIPODATO LISTANVARIABLES corcheteabre corchetecierra corcheteabre corchetecierra sigigual corcheteabre LISTASFILAS corchetecierra sigpuntoycoma { $$= new Asign
        | id corcheteabre EXPRESIONES corchetecierra sigigual EXPRESIONES sigpuntoycoma                                                             { $$= new Modif
        | id corcheteabre EXPRESIONES corchetecierra corcheteabre EXPRESIONES corchetecierra sigigual EXPRESIONES sigpuntoycoma                     { $$= new Modif
```

```
LISTASFILAS: corcheteabre LISTANEXP corchetecierra signocoma LISTASFILAS    { addLSMA($2); concatenarLSMA($5); $$=getLSMA(); }
        |  corcheteabre LISTANEXP corchetecierra                            { addLSMA($2); $$=getLSMA(); }
;


LISTANEXP:  EXPRESIONES signocoma LISTANEXP      {  addExp($1); concatenarlistaExp($3); $$=getExp(); limpiarlistExp();}
        |   EXPRESIONES                          {  addExp($1); $$=getExp(); }
;

LISTANEXPR:  EXPRESIONES signocoma LISTANEXPR    {  addExp($1); concatenarlistaExp($3); $$=getExp();}
        |   EXPRESIONES                          {  addExp($1); $$=getExp();}
;

TIPODATO: resint       {$$="int";}
        | resdouble    {$$="double";}
        | resbool      {$$="booleano";}
        | reschar      {$$= "char";}
        | resstring    {$$= "string";}
;

LISTANVARIABLES:  id signocoma LISTANVARIABLES    { addVariables($1); concatenarlista($3); $$=getLVariables();}
        |   id                                    { addVariables($1); $$=getLVariables();}
;

ASIGNACIONES: EXPRESIONES             {$$=$1;}
        | OTRASEXPRESIONES            {$$=$1;}
        | error                       { addError('Error sintáctico', 'No se reconoce' + $1, this._$.first_line, this._$.first_column);}
;
```

```
EXPRESIONES: OPERACIONES          {$$=$1;}
        | OPERACIONESRELACIONAL   {$$=$1;}
        | OPERADORESLOGICOS       {$$=$1;}
        | ACCESOVEC               {$$=$1;}
        | id                      {$$= new Oid($1, "id", @1.first_line, @1.first_column, "id"); }
        | caracter                {$$= new Dato($1, "char", @1.first_line, @1.first_column);}
        | cadena                  {$$= new Dato($1, "string", @1.first_line, @1.first_column);}
        | bool                    {$$= new Dato($1, "booleano", @1.first_line, @1.first_column);}
        | decimal                 {$$= new Dato($1, "double", @1.first_line, @1.first_column);}
        | numero                  {$$= new Dato($1, "int", @1.first_line, @1.first_column);}
;
OTRASEXPRESIONES: CASTEAR         {$$=$1;}
        | OPERADORTERNARIO        {$$=$1;}
        | INCREYDECRE             {$$=$1;}
        | LLAMADAS                {$$=$1;}
        | FTOLOWER                {$$=$1;}
        | FTOUPPER                {$$=$1;}
        | FROUND                  {$$=$1;}
        | FLENGTH                 {$$=$1;}
        | FTYPEOF                 {$$=$1;}
        | FTOSTRING               {$$=$1;}
        | FCSTR                   {$$=$1;}
;

ACCESOVEC: id corcheteabre EXPRESIONES  corchetecierra  corcheteabre EXPRESIONES  corchetecierra  {$$= new AccesoV2($1, $3, $6, @1.first_line, @1.first_column);}
        | id corcheteabre EXPRESIONES  corchetecierra                                              {$$= new AccesoV($1, $3, @1.first_line, @1.first_column) ;}
;
```

```
OPERACIONES: menos EXPRESIONES %prec Umenos                                    {$$= new Aritmetica($2, $2 , $1 + "unario", @1.first_line, @1.first_column );}
        | EXPRESIONES mas EXPRESIONES                                          {$$= new Aritmetica($1,$3,$2, @1.first_line, @1.first_column) ;}
        | EXPRESIONES menos EXPRESIONES                                        {$$= new Aritmetica($1,$3,$2, @1.first_line, @1.first_column) ;}
        | EXPRESIONES por EXPRESIONES                                          {$$= new Aritmetica($1,$3,$2, @1.first_line, @1.first_column) ;}
        | EXPRESIONES dividir EXPRESIONES                                      {$$= new Aritmetica($1,$3,$2, @1.first_line, @1.first_column) ;}
        | respotencia parentesisabre EXPRESIONES signocoma EXPRESIONES parentesiscierra  {$$= new Aritmetica($3,$5,$1, @1.first_line, @1.first_column) ;}
        | EXPRESIONES modulo EXPRESIONES                                       {$$= new Aritmetica($1,$3,$2, @1.first_line, @1.first_column) ;}
        | AGRUPACION                                                           {$$= $1;}
;

OPERACIONESRELACIONAL: EXPRESIONES igualigual EXPRESIONES   {$$= new Relacional($1,$3,$2, @1.first_line, @1.first_column);}
        | EXPRESIONES negacionigual EXPRESIONES             {$$= new Relacional($1,$3,$2, @1.first_line, @1.first_column);}
        | EXPRESIONES menorigual EXPRESIONES                {$$= new Relacional($1,$3,$2, @1.first_line, @1.first_column);}
        | EXPRESIONES menorque EXPRESIONES                  {$$= new Relacional($1,$3,$2, @1.first_line, @1.first_column);}
        | EXPRESIONES mayorigual EXPRESIONES                {$$= new Relacional($1,$3,$2, @1.first_line, @1.first_column);}
        | EXPRESIONES mayorque EXPRESIONES                  {$$= new Relacional($1,$3,$2, @1.first_line, @1.first_column);}
;

OPERADORESLOGICOS:  notlogico EXPRESIONES      {$$= new Logico($2, $2 ,$1, @1.first_line, @1.first_column);}
        | EXPRESIONES andlogico EXPRESIONES    {$$= new Logico($1, $3 ,$2, @1.first_line, @1.first_column);}
        | EXPRESIONES orlogico EXPRESIONES     {$$= new Logico($1, $3 ,$2, @1.first_line, @1.first_column);}
;

OPERADORTERNARIO: OPERACIONESRELACIONAL siginterrogacion ASIGNACIONES dospuntos ASIGNACIONES    {$$=$1 +" "+ $2 +" "+ $3 +$4 + " " + $5;}
;


AGRUPACION: parentesisabre EXPRESIONES parentesiscierra             {$$= $2;}
;
```

```
CASTEAR: parentesisabre TIPODATO parentesiscierra EXPRESIONES        {$$=new Castear($4,$2, @1.first_line, @1.first_column);}
;

INCREYDECRE: EXPRESIONES sigincremento                               { $$= new IncrementoDecremento2($1,"++", @1.first_line, @1.first_column, $1.tipoid, $1.id); }
        | EXPRESIONES sigdecremento                                  { $$= new IncrementoDecremento2($1,"--", @1.first_line, @1.first_column, $1.tipoid, $1.id); }
;

SENTENCIAS: SENTIF                                                    {$$=$1;}
        | SENTSWITCH                                                  {$$=$1;}
        | SENTDOWHILE                                                 {$$=$1;}
        | SENTWHILE                                                   {$$=$1;}
        | SENTFOR                                                     {$$=$1;}
        | error                      { addError('Error sintáctico', 'No se reconoce' + $1, this._$.first_line, this._$.first_column);}
;

SENTIF: resif parentesisabre EXPRESIONES parentesiscierra llaveabre CONTENIDOS FINIF    {$$= new If($3, $6, $7 ,@1.first_line, @1.first_column); limpiarElSEIF();}
;

CONTENIDOS: CONTENIDOS CONTEIF                          { $$ = $1; $$.push($2);}
        | CONTEIF                                       { $$ = []; $$.push($1);}
;
CONTEIF: resbreak sigpuntoycoma                                          {$$= new BBreak(@1.first_line, @1.first_column);}
        | rescontinue sigpuntoycoma                                      {$$= new Continu(@1.first_line, @1.first_column);}
        | RETORNOS                                                       {$$= $1;}
        | INSTRUCCION                                                    {$$= $1;}
;

FINIF: llavecierra                                                                                  {$$=[$1];}
        | llavecierra reselse resif parentesisabre EXPRESIONES parentesiscierra llaveabre CONTENIDOS FINIF    {addELSEif(new elseif($5, $8, $9 , @1.first_line, @1.first_c
        | llavecierra reselse llaveabre CONTENIDOS llavecierra                                      {$$= [new soloelse($4, @1.first_line, @1.first_column)];}
;
```

```
SENTSWITCH: resswitch parentesisabre EXPRESIONES parentesiscierra llaveabre SWCASOS llavecierra    {$$=new Switchh($3, $6, @1.first_line, @1.first_column); limpiarlistCas
;

SWCASOS: SWCASE                                                  { addCasos($1); $$=getCasos();}
        | SWCASE SWCASOS                                         { addCasos($1); concatenarlistaCasos($2); $$=getCasos();}
;

SWCASE: rescase ASIGNACIONES dospuntos CONTENIDOS                {$$= new Scasos($2, $4, @1.first_line, @1.first_column);}
        | resdefault dospuntos CONTENIDOS                        {$$= new Sdefault($3, @1.first_line, @1.first_column);}
;

SENTWHILE: reswhile parentesisabre EXPRESIONES parentesiscierra llaveabre CONTENIDOSCICLOS llavecierra                {$$= new Bwhile($3, $6,  @1.fir
;

SENTFOR: resfor parentesisabre DECLARACIONES EXPRESIONES sigpuntoycoma INCREYDECRE parentesiscierra llaveabre CONTENIDOSCICLOS llavecierra {$$= new BFor($3, $4, $6, $9,
;

SENTDOWHILE: resdo llaveabre CONTENIDOSCICLOS llavecierra reswhile parentesisabre EXPRESIONES parentesiscierra sigpuntoycoma    {$$= new Bdowhile($3, $7,  @1.f
;

CONTENIDOSCICLOS: CONTENIDOSCICLOS CONTENIDOCICL                 { $$ = $1; $$.push($2);}
        | CONTENIDOCICL                                          { $$ = []; $$.push($1);}
;
CONTENIDOCICL:resbreak sigpuntoycoma                             {$$= new BBreak(@1.first_line, @1.first_column);}
        | rescontinue sigpuntoycoma                              {$$= new Continu(@1.first_line, @1.first_column);}
        | RETORNOS                                               {$$=$1;}
        | INSTRUCCION                                            {$$=$1;}
;

RETORNOS: resreturn sigpuntoycoma                                {$$=$1 + $2;}
        | resreturn ASIGNACIONES sigpuntoycoma                   {$$=$1 + " " + $2 + " "+ $3;}
;
```

```
FUNCIONES: TIPODATO id SNPARAMETROS llaveabre CONTENIDOSCICLOS llavecierra    {$$=$1 + " "+ $2 + " " + $3 +$4 + " " + $5 + $6;}
;

SNPARAMETROS: parentesisabre PARAMETROS parentesiscierra                     {$$=$1 + " " + $2 + " " + $3;}
        | parentesisabre  parentesiscierra                                   {$$=$1 + $2;}
;

PARAMETROS: TIPODATO EXPRESIONES                                             {$$=$1+ $2;}
        | TIPODATO EXPRESIONES signocoma PARAMETROS                          {$$=$1 + " " +  $2 +" " + $3 + " "  +  $4;}
;

METODOS: resvoid id SNPARAMETROS llaveabre CONTENIDOSMETOD llavecierra       {$$=$1 + " " + $2 + " "+$3 + $4 + " " + $5+ " " + $6;}
;

CONTENIDOSMETOD: CONTENIDOSMETOD CONTMETOD                                   { $$ = $1; $$.push($2);}
        | CONTMETOD                                                          { $$ = []; $$.push($1);}
;
CONTMETOD: resbreak sigpuntoycoma                                            {$$= new BBreak(@1.first_line, @1.first_column);}
        | rescontinue sigpuntoycoma                                          {$$= new Continu(@1.first_line, @1.first_column);}
        | INSTRUCCION                                                        {$$=$1;}
;

LLAMADAS: id SNPARAMETROS                                                    {$$=$1 + " " + $2;}
;

FCOUT:  rescout menormenor ASIGNACIONES menormenor resendl sigpuntoycoma    {$$= new Print($3, "salto", @1.first_line, @1.first_column) ;}
        | rescout menormenor ASIGNACIONES sigpuntoycoma                          {$$= new Print($3, "sinsalto", @1.first_line, @1.first_column) ;}
;

FTOLOWER: restolower parentesisabre ASIGNACIONES parentesiscierra            {$$= new Ftolower($3, @1.first_line, @1.first_column);}
;

FTOUPPER: restoupper parentesisabre ASIGNACIONES parentesiscierra            {$$= new Ftoupper($3, @1.first_line, @1.first_column);}
```

```
FTOUPPER: restoupper parentesisabre ASIGNACIONES parentesiscierra            {$$= new Ftoupper($3, @1.first_line, @1.first_column);}
;

FROUND: resround parentesisabre ASIGNACIONES parentesiscierra                {$$= new Fround($3, @1.first_line, @1.first_column);}
;

FLENGTH: EXPRESIONES sigpunto reslength parentesisabre parentesiscierra      {$$=new Flength($1, @1.first_line, @1.first_column);}
;

FTYPEOF: restypeof parentesisabre ASIGNACIONES parentesiscierra              {$$=new Ftypeof($3, @1.first_line, @1.first_column);}
;

FTOSTRING: restostring parentesisabre ASIGNACIONES parentesiscierra          {$$=new Ftostring($3, @1.first_line, @1.first_column);}
;

FCSTR: EXPRESIONES sigpunto rescstr parentesisabre parentesiscierra          {$$=$1 + " " + $2 + " " + $3 + " " + $4 + $5;}
;

FEXECUTE: resexecute id SNPARAMETROS sigpuntoycoma                           {$$=$1 + " " + $2 + " " + $3 + " " + $4;}
;
```

Manejo de Errores

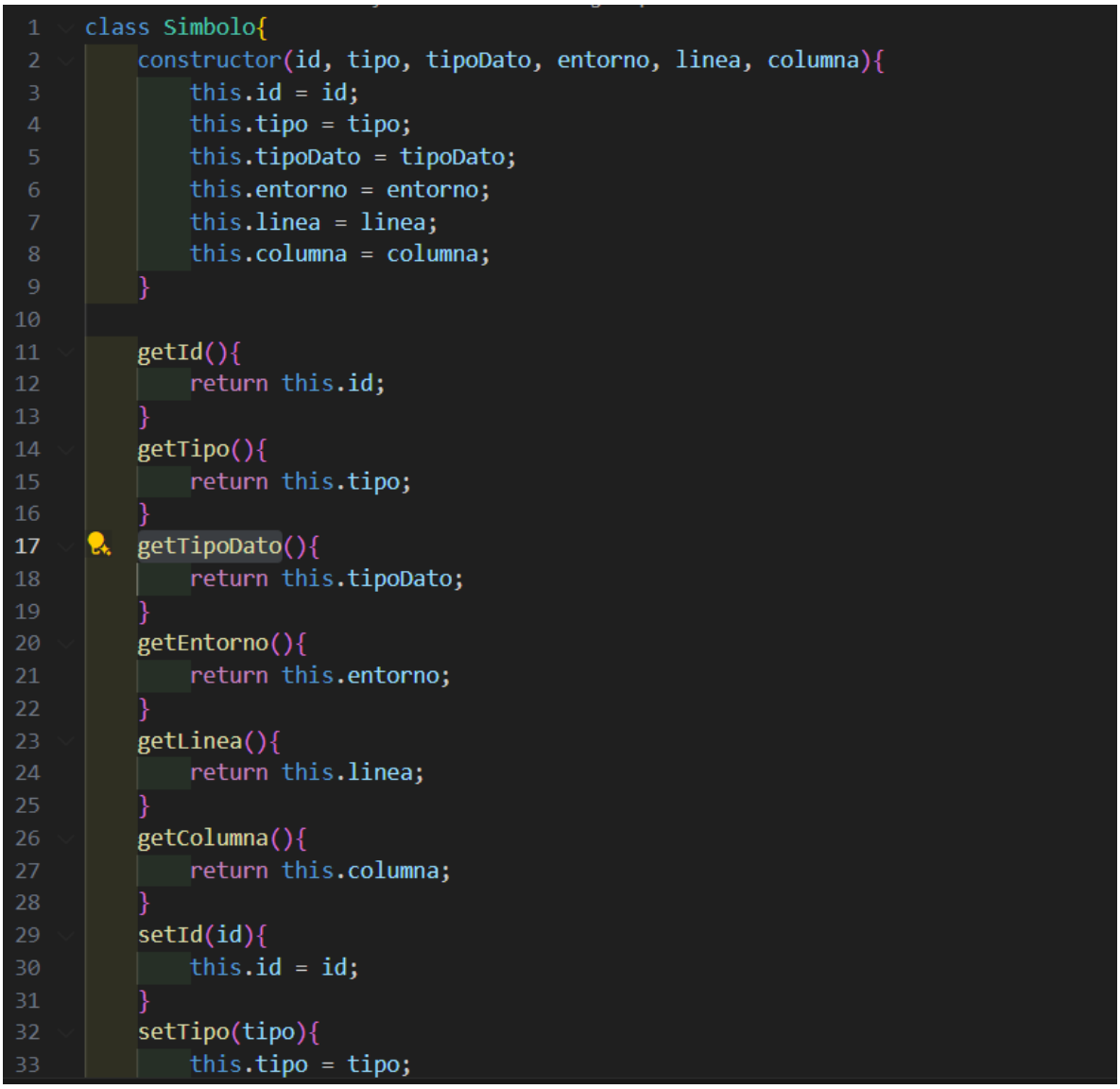
Para el manejo de los errores, se utilizó un objeto que contendría los parámetros que se observan en la imagen y estas a su vez se almacenaron dentro de una lista y así poder recorrerla para los reportes.

```
class MyError{
    constructor(tipo, descripcion, linea, columna){
        this.tipo = tipo;
        this.descripcion = descripcion;
        this.linea = linea;
        this.columna = columna;
    }
    getTipo(){
        return this.tipo;
    }
    getDescripcion(){
        return this.descripcion;
    }
    getLinea(){
        return this.linea;
    }
    getColumna(){
        return this.columna;
    }
    setTipo(tipo){
        this.tipo = tipo;
    }
    setDescripcion(descripcion){
        this.descripcion = descripcion;
    }
    setLinea(linea){
        this.linea = linea;
    }
    setColumna(columna){
        this.columna = columna;
    }

}
```

Manejo de Simbolos

Para este se manejó un objeto y varias listas, el primero para almacenar datos con valores simples, y el segundo para manejar los que incluyen arreglos de valores y arreglos de 2 dimensiones, y todos estos objetos se almacenaron a su vez en una lista para posteriormente poder ser utilizado para obtener los datos tanto

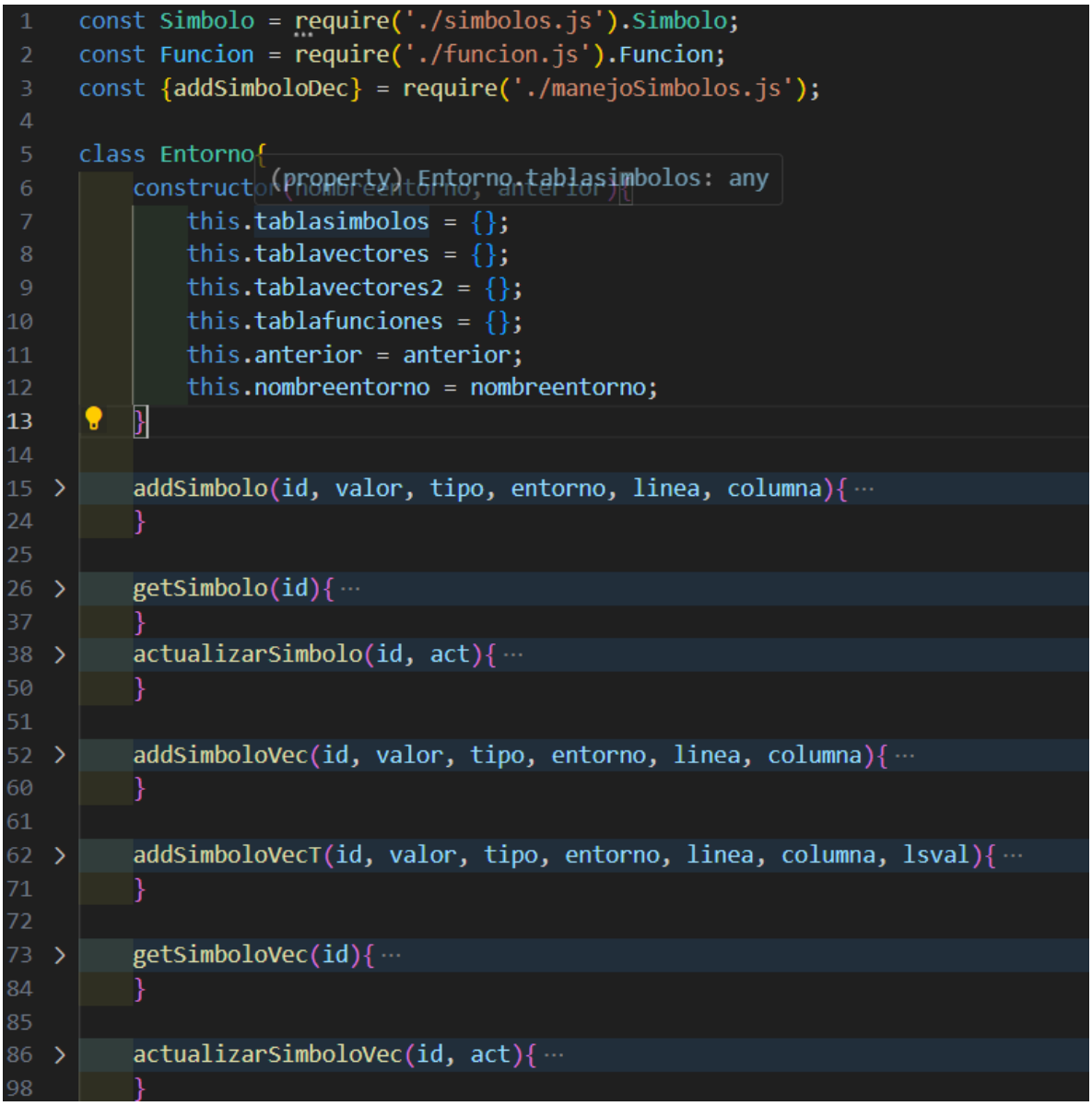
para manejo de funcionalidades y el reporte. Para ello se muestran los dos objetos en las imágenes siguientes:



```
class Simbolo{
    constructor(id, tipo, tipoDato, entorno, linea, columna){
        this.id = id;
        this.tipo = tipo;
        this.tipoDato = tipoDato;
        this.entorno = entorno;
        this.linea = linea;
        this.columna = columna;
    }

    getId(){
        return this.id;
    }
    getTipo(){
        return this.tipo;
    }
    getTipoDato(){
        return this.tipoDato;
    }
    getEntorno(){
        return this.entorno;
    }
    getLinea(){
        return this.linea;
    }
    getColumna(){
        return this.columna;
    }
    setId(id){
        this.id = id;
    }
    setTipo(tipo){
        this.tipo = tipo;
```

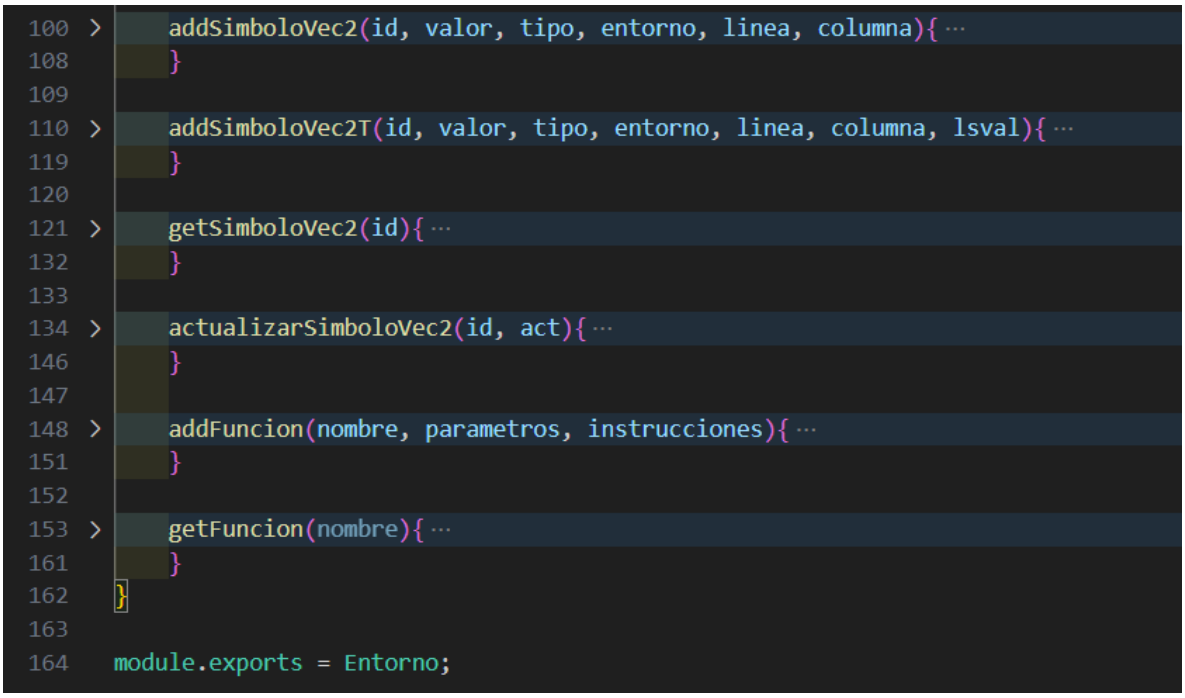
```
const Simbolos = require('./simbolos');
const fs = require('fs');
let tablaSimbolos = [];
const urlReporteS = '../Backend/analisisSem/ReportesArchivos/ReporteSimbolos.html';

function addSimboloDec(vsimbolo) {
    tablaSimbolos.push(vsimbolo);
}

function generarTablaS() {···
    }

    function escribirArchivo(contenido) {···
    }

    function limpiarTablaS() {···
    }

    function openReporteS(){···
    }

    module.exports = {
        addSimboloDec,
        generarTablaS,
        limpiarTablaS,
        openReporteS
    };
```

Para tener un manejo correcto de los símbolos debido a los Entornos, se manejaron com diversas funciones, en las cuales se agrega el símbolo, se obtiene, se actualiza, esa misma lógica se utilizó para todas las declaraciones, declaraciones de arreglos, y para las funciones y métodos del lenguaje.



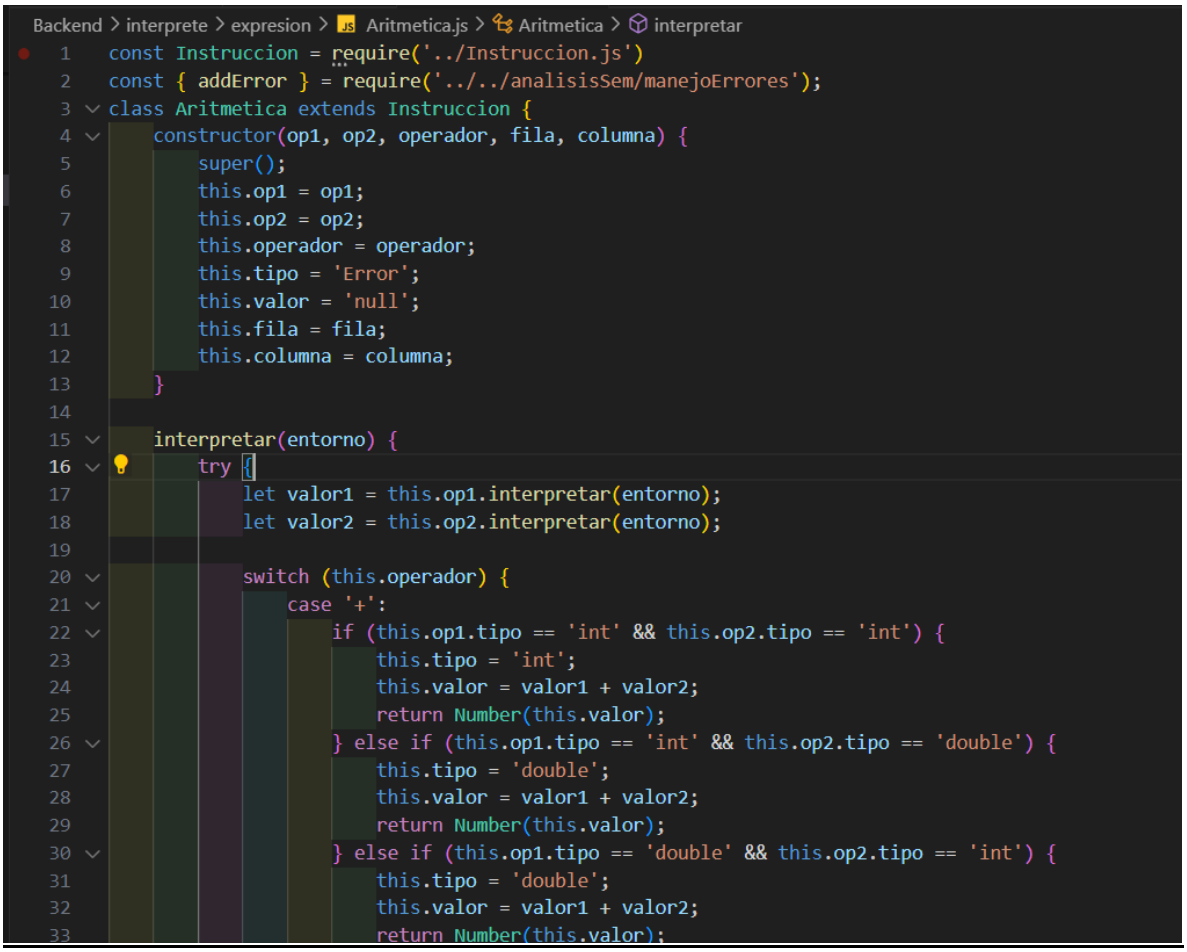
```
const Simbolo = require('./simbolos.js').Simbolo;
const Funcion = require('./funcion.js').Funcion;
const {addSimboloDec} = require('./manejoSimbolos.js');

class Entorno{
    constructor(nombreentorno, anterior){
        this.tablasimbolos = {};
        this.tablavectores = {};
        this.tablavectores2 = {};
        this.tablafunciones = {};
        this.anterior = anterior;
        this.nombreentorno = nombreentorno;
    }

    addSimbolo(id, valor, tipo, entorno, linea, columna){ ···
    }

    getSimbolo(id){ ···
    }
    actualizarSimbolo(id, act){ ···
    }

    addSimboloVec(id, valor, tipo, entorno, linea, columna){ ···
    }

    addSimboloVecT(id, valor, tipo, entorno, linea, columna, lsval){ ···
    }

    getSimboloVec(id){ ···
    }

    actualizarSimboloVec(id, act){ ···
    }
```

(property) Entorno.tablasimbolos: any

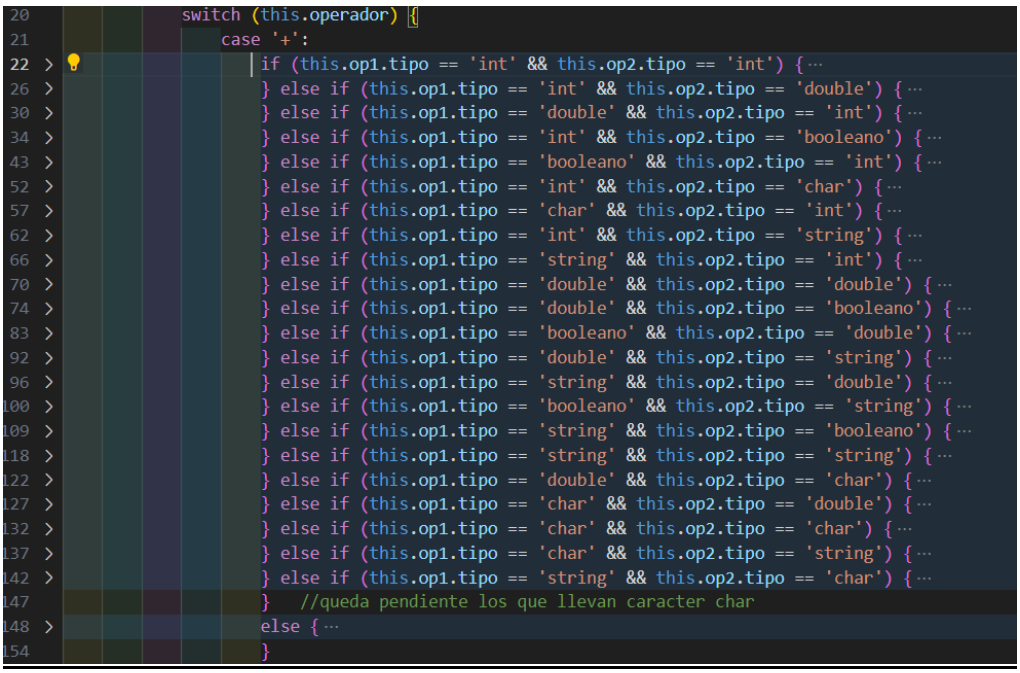


Manejo de Operaciones

Para las operaciones aritméticas se realizó una clase donde se recibiera como parámetro en el constructor los valores de la operación, el tipo de operador a utilizar, el valor como resultado, la fila y la columna de dicha operación, primero se evalúa el tipo de operación a recibir, luego los tipos de datos de cada uno para así verificar si la operación se realizará como int, string, carácter, bool, double, y retorna el resultado. En la imagen solo se muestra el análisis de la suma, pero de la misma forma se analiza el resto de operaciones en el resto del archivo



```
Backend > interprete > expresion > Aritmetica.js > Aritmetica > interpretar
const Instruccion = require('../Instruccion.js')
const { addError } = require('../../analisisSem/manejoErrores');
class Aritmetica extends Instruccion {
    constructor(op1, op2, operador, fila, columna) {
        super();
        this.op1 = op1;
        this.op2 = op2;
        this.operador = operador;
        this.tipo = 'Error';
        this.valor = 'null';
        this.fila = fila;
        this.columna = columna;
    }

    interpretar(entorno) {
        try {
            let valor1 = this.op1.interpretar(entorno);
            let valor2 = this.op2.interpretar(entorno);

            switch (this.operador) {
                case '+':
                    if (this.op1.tipo == 'int' && this.op2.tipo == 'int') {
                        this.tipo = 'int';
                        this.valor = valor1 + valor2;
                        return Number(this.valor);
                    } else if (this.op1.tipo == 'int' && this.op2.tipo == 'double') {
                        this.tipo = 'double';
                        this.valor = valor1 + valor2;
                        return Number(this.valor);
                    } else if (this.op1.tipo == 'double' && this.op2.tipo == 'int') {
                        this.tipo = 'double';
                        this.valor = valor1 + valor2;
                        return Number(this.valor);
```



```
            switch (this.operador) {
                case '+':
                    if (this.op1.tipo == 'int' && this.op2.tipo == 'int') {…
                    } else if (this.op1.tipo == 'int' && this.op2.tipo == 'double') {…
                    } else if (this.op1.tipo == 'double' && this.op2.tipo == 'int') {…
                    } else if (this.op1.tipo == 'int' && this.op2.tipo == 'booleano') {…
                    } else if (this.op1.tipo == 'booleano' && this.op2.tipo == 'int') {…
                    } else if (this.op1.tipo == 'int' && this.op2.tipo == 'char') {…
                    } else if (this.op1.tipo == 'char' && this.op2.tipo == 'int') {…
                    } else if (this.op1.tipo == 'int' && this.op2.tipo == 'string') {…
                    } else if (this.op1.tipo == 'string' && this.op2.tipo == 'int') {…
                    } else if (this.op1.tipo == 'double' && this.op2.tipo == 'double') {…
                    } else if (this.op1.tipo == 'double' && this.op2.tipo == 'booleano') {…
                    } else if (this.op1.tipo == 'booleano' && this.op2.tipo == 'double') {…
                    } else if (this.op1.tipo == 'double' && this.op2.tipo == 'string') {…
                    } else if (this.op1.tipo == 'string' && this.op2.tipo == 'double') {…
                    } else if (this.op1.tipo == 'booleano' && this.op2.tipo == 'string') {…
                    } else if (this.op1.tipo == 'string' && this.op2.tipo == 'booleano') {…
                    } else if (this.op1.tipo == 'string' && this.op2.tipo == 'string') {…
                    } else if (this.op1.tipo == 'double' && this.op2.tipo == 'char') {…
                    } else if (this.op1.tipo == 'char' && this.op2.tipo == 'double') {…
                    } else if (this.op1.tipo == 'char' && this.op2.tipo == 'char') {…
                    } else if (this.op1.tipo == 'char' && this.op2.tipo == 'string') {…
                    } else if (this.op1.tipo == 'string' && this.op2.tipo == 'char') {…
                    }   //queda pendiente los que llevan caracter char
                    else {…
                    }
```

Manejo de Datos

Dentro de la clase Dato, se retornan los datos encontrados en el análisis en jison, los cuales se convierten al valor correspondiente.

```
class Dato extends Instruccion {
    constructor(valor,tipo, fila, columna) {
        this.tipo = tipo;
        this.fila = fila;
        this.columna = columna;
    }

    interpretar(entorno) {
        try {
                switch (this.tipo) {
                    case 'int':
                        return Number(this.valor);
                    case 'string':
                        //this.valor = this.valor.replace(/\\"/g, '"');
                        //console.log('valor: ', this.valor); //    no  me sustituye el de las comillas
                        //.replace("\\\"","\"")
                        this.valor = this.valor.replace(/\\n/g, '\n').replace(/\\t/g, '\t').replace(/\\'/g, "'").replace(/\\\\/g, '\\');
                        return this.valor;
                    case 'booleano':
                        if (this.valor == 'true') {
                            this.valor = true;
                            return this.valor;
                        } else {
                            this.valor = false;
                            return this.valor;
                        }
                    case 'char':
                        return this.valor;
                    case 'double':
                        return Number(this.valor);
                    case 'Error':
                        addError('Error Semantico', 'Dato de tipo incorrecto ' + this.tipo + ' - ' + this.valor, this.fila, this.columna);
                        return;
```

Cuando se tienen id, se busca en la tabla para obtener el valor y el tipo de dato del mismo y se retorna.

```
class Oid extends Instruccion {
    constructor(id, tipo, fila, columna, tipoid) {
        super();
        this.valor = "";
        this.id = id;
        this.tipo = tipo;
        this.tipoid = tipoid;
        this.fila = fila;
        this.columna = columna;
    }

    interpretar(entorno) {
        try {
                //validar en obtener del entorno
                this.valor = entorno.getSimbolo(this.id);
                let data = this.valor.getTipo();
                this.valor = data.valor;
                this.tipo = data.tipo;
                if (this.tipo == 'int') {
                    this.valor = Number(this.valor);
                } else if (this.tipo == 'double') {
                    this.valor = Number(this.valor);
                } else if (this.tipo == 'booleano') {
                    this.valor = Boolean(this.valor);
                } else if (this.tipo == 'string') {
                    this.valor = this.valor.replace(/\\n/g, '\n').replace(/\\t/g, '\t').replace(/\\'/g, "'").replace(/\\\\/g, '\\');
                } else if (this.tipo == 'char') {
                    this.valor = this.valor;
                }
                return this.valor;

        } catch (error) {
            addError('Error', 'Error al interpretar el dato' + error, this.fila, this.columna);
```