

Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Curso: Laboratorio Organización de Lenguajes & Compiladores 1
Auxiliar: José Diego Pérez Toralla
Auxiliar: Maynor Octavio Piló Tuy



Manual Técnico

Juan F. Urbina S. **2019060651**
Sección: C

Guatemala, Mayo de 2023

Índice

Introducción	3
Requisitos del Sistema	4
Explicación del Código	5
Funciones del Programa	25
Conclusiones.....	33

Introducción

Dentro del segundo proyecto del laboratorio de Organización de Lenguajes & Compiladores 1, la aplicación realizada fue en base al uso de Jison para el lenguaje de programación de JavaScript, abordando la culminación de conceptos de gramáticas y expresiones regulares para crear un compilador y analizar por consiguiente los datos ingresados, la declaración de variables y cadenas, entre otras, mediante el archivo de prueba respectivo, la utilización de las listas para el almacenamiento de algunas variables dentro del archivo Jison y la utilización de memoria dinámica para almacenar las variables y tokens para la utilización posterior en lo que son las tablas de errores, tablas de símbolos y la generación de un árbol ast que contiene las concatenaciones de las respectivas variables y funciones que contienen valores e instrucciones aceptando mediante las expresiones regulares las variables y funciones correspondientes.

Se empleo el lenguaje de JavaScript como herramienta en la función de las estructuras para el análisis, almacenamiento de la información y la utilización del fronted y backend para el envío y recibimiento de información (en este caso no se utilizaron métodos de POST, UPDATE & GET) todo fue desarrollado para el uso dentro del fronted respecto a las funciones, para el análisis de archivos de información, envío del AST, obtención de tabla de errores, obtención de tabla de símbolos, etc.

Se pudo determinar que la implementación de las estructuras y el almacenamiento y/o arreglo de la información obtenida en cada una de ellas no mostro ningún problema durante su ejecución.

Requisitos del Sistema

- ***Sistema Operativo:*** Windows 7 o superior
- ***CPU:*** Intel Pentium D o AMD Athlon 64 (K8) 2.6GHz. (Requisitos Mínimo)
- ***RAM:*** 600MB
- ***Lenguaje Utilizado:*** JavaScript
- ***IDE:*** Visual Studio Code
- ***USO de Framework:*** React

Explicación del Código

Análisis Léxico

Utiliza las siguientes expresiones regulares para obtener los parámetros a la hora de leer el archivo, obteniendo los errores léxicos que tenga el archivo.

```
([a-zA-Z])([a-zA-Z0-9_])* return 'id'
['']\|['']\|['']\|['']\n|['']\t|['']\r|[''].?[''] return 'caracter'
[0-9]+(".[0-9]+\b) return 'doble'
[0-9]+ return 'entero'
```

```
[ " ] { cadena = ''; this.begin("string"); }
<string>[^\\]+ { cadena += yytext; }
<string>"\\" { cadena += "\""; }
<string>"\\n" { cadena += "\n"; }
<string>\\s { cadena += " "; }
<string>"\\t" { cadena += "\t"; }
<string>"\\\\" { cadena += "\\"; }
<string>"\\'" { cadena += "'"; }
<string>"\\r" { cadena += "\r"; }
<string>["] { yytext = cadena; this.popState(); return 'cadena'; }
```

```

"//" .*                                // comentario
[/][*][^]*[*]+([^/*][^]*[*])+[/] // comentario multilinea

"clase"          return 'prclase'
"double"         return 'prdouble'
"int"            return 'printeger'
"boolean"        return 'prboolean'
"char"           return 'prchar'
"string"         return 'prstring'
"list"           return 'prlist'
"new"            return 'prnew'
"add"            return 'pradd'

"if"             return 'prif'
"else"           return 'prelse'
"switch"         return 'prswitch'
"case"           return 'prcase'
"break"          return 'prbreak'
"while"          return 'prwhile'
"for"            return 'prfor'
"do"             return 'prdo'
"default"        return 'prdefault'
"continue"       return 'prcontinue'
"return"         return 'prreturn'
"void"           return 'prvoid'
"++"             return 'incremento'
"--"            return 'decremento'

"print"          return 'prprint'
"toLowerCase"    return 'prtoLowerCase'
"toUpperCase"    return 'prtoUpperCase'
"length"         return 'prlength'
"truncate"       return 'prtruncate'
"round"          return 'prround'
"typeof"         return 'prtypeof'
"toString"       return 'prtoString'
"toArray"        return 'prtoArray'
"exec"           return 'prexec'
"main"           return 'premain'

"true"           return 'true'
>false"          return 'false'

```

```
"||"      return 'or'
"&&"      return 'and'
"!="      return 'diferente'
"=="      return 'igualigual'
"!"       return 'not'
"="       return 'igual'
"<="      return 'menorigual'
">="      return 'mayorigual'
">"      return 'mayor'
"<"      return 'menor'
","      return 'coma'
";"      return 'ptcoma'
"."      return 'punto'
":"      return 'dospuntos'
"{"      return 'labre'
"}"      return 'lcierra'
"*"      return 'multi'
"/"      return 'div'
"-"      return 'menos'
"+"      return 'suma'
"^"      return 'exponente'
"%"      return 'modulo'
"("      return 'pabre'
")"      return 'pcierra'
"?"      return 'interrogacion'
"["      return 'cabre'
"]"      return 'ccierra'
```

Precedencia de Operadores

Sirve para obtener un orden a la hora de hacer las operaciones de valores, cuales tienen mayor “relevancia” o cuales operar antes de otros dentro de una función u orden.

```
%left 'interrogacion'
%left 'or'
%left 'and'
%right 'not'
%left 'igualigual' 'diferente' 'menor' 'menorigual' 'mayor' 'mayorigual'
%left 'suma' 'menos'
%left 'multi' 'div' 'modulo'
%left 'exponente'
%left 'incremento','decremento'
%left umenos
%left 'pabre'
```


Análisis Sintáctico

Se encarga de obtener los errores sintácticos y mediante una gramática regular cada en cada producción obtener los datos requeridos.

Se analizarán las cadenas y separarán los tokens para verificarlos posteriormente.

```
%% /*Inicio gramatica*/

ini
: ENTRADA EOF { retorno = { parse: $1, errores: errores }; errores = []; return retorno; }
| error EOF { retorno = { parse: null, errores: errores }; errores = []; return retorno; }
;

ENTRADA
: ENTRADA ENTZERO { if($2!="") $1.push($2); $$=$1; }
| ENTZERO { if($1!="") $$=[$1]; else $$=[]; }
;

ENTZERO
: FUNCIONBODY { $$=$1 }
| METODOBODY { $$=$1 }
| EXECBODY { $$=$1 }
| DEC_VAR { $$=$1 }
| DEC_VECT { $$=$1 }
| DEC_LIST { $$=$1 }
;

FUNCIONBODY
: TIPO id pabre pcierra labre INSTRUCCION lcierra { $$ = INSTRUCCION.nuevaFuncion($2, null, $6, $1, this._$.first_line, this._$.first_li }
| TIPO id pabre pcierra labre lcierra { $$ = INSTRUCCION.nuevaFuncion($2, null, [], $1, this._$.first_line, this._$.first_li }
| TIPO id pabre LISTAPARAMETROS pcierra labre INSTRUCCION lcierra { $$ = INSTRUCCION.nuevaFuncion($2, $4, $7, $1, this._$.first_line, this._$.first_li }
| TIPO id pabre LISTAPARAMETROS pcierra labre lcierra { $$ = INSTRUCCION.nuevaFuncion($2, $4, [], $1, this._$.first_line, this._$.first_li }
| TIPO_VECT id pabre pcierra labre INSTRUCCION lcierra { $$ = INSTRUCCION.nuevaFuncion($2, null, $6, {vector: $1}, this._$.first_line, this._$.first_li }
| TIPO_VECT id pabre pcierra labre lcierra { $$ = INSTRUCCION.nuevaFuncion($2, null, [], {vector: $1}, this._$.first_line, this._$.first_li }
| TIPO_VECT id pabre LISTAPARAMETROS pcierra labre INSTRUCCION lcierra { $$ = INSTRUCCION.nuevaFuncion($2, $4, $7, {vector: $1}, this._$.first_line, this._$.first_li }
| TIPO_VECT id pabre LISTAPARAMETROS pcierra labre lcierra { $$ = INSTRUCCION.nuevaFuncion($2, $4, [], {vector: $1}, this._$.first_line, this._$.first_li }
| TIPO_LIST id pabre pcierra labre INSTRUCCION lcierra { $$ = INSTRUCCION.nuevaFuncion($2, null, $6, {lista: $1}, this._$.first_line, this._$.first_li }
| TIPO_LIST id pabre pcierra labre lcierra { $$ = INSTRUCCION.nuevaFuncion($2, null, [], {lista: $1}, this._$.first_line, this._$.first_li }
| TIPO_LIST id pabre LISTAPARAMETROS pcierra labre INSTRUCCION lcierra { $$ = INSTRUCCION.nuevaFuncion($2, $4, $7, {lista: $1}, this._$.first_line, this._$.first_li }
| TIPO_LIST id pabre LISTAPARAMETROS pcierra labre lcierra { $$ = INSTRUCCION.nuevaFuncion($2, $4, [], {lista: $1}, this._$.first_line, this._$.first_li }
| TIPO error lcierra { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de func }
| TIPO_VECT error lcierra { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de func }
| TIPO_LIST error lcierra { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de func }
;

METODOBODY
: prvoid id pabre pcierra labre INSTRUCCION lcierra { $$ = INSTRUCCION.nuevoMetodo($2, [], $6, this._$.first_line, this._$.first_li }
| prvoid id pabre pcierra labre lcierra { $$ = INSTRUCCION.nuevoMetodo($2, [], [], this._$.first_line, this._$.first_li }
| prvoid id pabre LISTAPARAMETROS pcierra labre INSTRUCCION lcierra { $$ = INSTRUCCION.nuevoMetodo($2, $4, $7, this._$.first_line, this._$.first_li }
| prvoid id pabre LISTAPARAMETROS pcierra labre lcierra { $$ = INSTRUCCION.nuevoMetodo($2, $4, [], this._$.first_line, this._$.first_li }
| prvoid error lcierra { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de méto }
;
```

```

    EXECBODY
      : premain id pabre pcierra ptcoma      { $$ = INSTRUCCION.nuevoExec($2, null, this._$.first_line, this._$.first_c
      | premain id pabre LISTAVALORES pcierra ptcoma      { $$ = INSTRUCCION.nuevoExec($2, $4, this._$.first_line, this._$.first_co
      | premain error ptcoma      { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Llamada de main no
      ;

    LISTAPARAMETROS
      : LISTAPARAMETROS coma PARAMETROS      {$1.push($3); $$=$1;}
      | PARAMETROS      {$$=[$1];}
      ;

    PARAMETROS
      : TIPO_VECT id      { $$ = INSTRUCCION.nuevoParametro($2, {vector: $1}, this._$.first_line, this._$.first_column+1)}
      | TIPO_LIST id      { $$ = INSTRUCCION.nuevoParametro($2, {lista: $1}, this._$.first_line, this._$.first_column+1)}
      | TIPO id      { $$ = INSTRUCCION.nuevoParametro($2, $1, this._$.first_line, this._$.first_column+1)}
      ;

    INSTRUCCION
      : INSTRUCCION INSCERO      { if($2!="") $1.push($2); $$=$1; }
      | INSCERO      { if($1!="") $$=[$1]; else $$=[]; }
      ;

    INSCERO
      : DEC_VAR      { $$=$1 }
      | SENTENCIACONTROL      { $$=$1 }
      | SENTENCIACICLO      { $$=$1 }
      | DEC_VECT      { $$=$1 }
      | DEC_LIST      { $$=$1 }
      | SENTENCIATRANSFERENCIA      { $$=$1 }
      | LLAMADA ptcoma      { $$=$1 }
      | FPRINT      { $$=$1 }
      | error ptcoma      { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de instrucción no válida.", línea: this._$.first_line, columna: this._$.first_column+1 }); }
      | error lcierra      { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de instrucción no válida.", línea: this._$.first_line, columna: this._$.first_column+1 }); }
      ;

    SENTENCIATRANSFERENCIA
      : prbreak ptcoma      { $$ = new INSTRUCCION.nuevoBreak(this._$.first_line, this._$.first_column+1) }
      | prreturn EXPRESION ptcoma      { $$ = new INSTRUCCION.nuevoReturn($2, this._$.first_line, this._$.first_column+1) }
      | prcontinue ptcoma      { $$ = new INSTRUCCION.nuevoContinue(this._$.first_line, this._$.first_column+1) }
      | prreturn ptcoma      { $$ = new INSTRUCCION.nuevoReturn(null, this._$.first_line, this._$.first_column+1) }
      ;

    SENTENCIACICLO
      : WHILE      { $$=$1 }
      | FOR      { $$=$1 }
      | DOWHILE      { $$=$1 }
      ;

```

```

WHILE
  : prwhile pabre EXPRESION pcierra labre INSTRUCCION lcierra      {$$ = new INSTRUCCION.nuevoWhile($3, $6, this._$.first_line, this._$.first_column+1)}
  | prwhile pabre EXPRESION pcierra labre lcierra                  {$$ = new INSTRUCCION.nuevoWhile($3, [], this._$.first_line, this._$.first_column+1)}
  | prwhile error lcierra                                           {$$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de sentencia While no válida.", linea: this._$.first_line, columna: this._$.first_column+1});}
;

FOR
  : prfor pabre DEC_VAR EXPRESION ptcoma ACTUALIZACION pcierra labre INSTRUCCION lcierra      {$9.push($6); $$ = new INSTRUCCION.nuevoFor($3, $6, this._$.first_line, this._$.first_column+1)}
  | prfor pabre DEC_VAR EXPRESION ptcoma ACTUALIZACION pcierra labre lcierra                  {$$ = new INSTRUCCION.nuevoFor($3, [], this._$.first_line, this._$.first_column+1)}
  | prfor error lcierra                                           {$$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de sentencia For no válida.", linea: this._$.first_line, columna: this._$.first_column+1});}
;

ACTUALIZACION
  : id igual EXPRESION {$$ = INSTRUCCION.nuevaAsignacion($1, $3, this._$.first_line, this._$.first_column+1)}
  | id incremento {$$ = INSTRUCCION.nuevaAsignacion($1, { opIzq: { tipo: 'VAL_IDENTIFICADOR', valor: $1, linea: this._$.first_line, columna: this._$.first_column+1}, opDcha: '+' }, this._$.first_line, this._$.first_column+1)}
  | id decremento {$$ = INSTRUCCION.nuevaAsignacion($1, { opIzq: { tipo: 'VAL_IDENTIFICADOR', valor: $1, linea: this._$.first_line, columna: this._$.first_column+1}, opDcha: '-' }, this._$.first_line, this._$.first_column+1)}
;

DOWHILE
  : prdo labre INSTRUCCION lcierra prwhile pabre EXPRESION pcierra ptcoma      {$$ = new INSTRUCCION.nuevoDoWhile($7, $3, this._$.first_line, this._$.first_column+1)}
  | prdo labre lcierra prwhile pabre EXPRESION pcierra ptcoma                  {$$ = new INSTRUCCION.nuevoDoWhile($7, [], this._$.first_line, this._$.first_column+1)}
  | prdo error ptcoma                                                           {$$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de sentencia DoWhile no válida.", linea: this._$.first_line, columna: this._$.first_column+1});}
;

SENTENCIACONTROL
  : CONTROLIF      {$$=$1}
  | SWITCH          {$$=$1}
;

CONTROLIF
  : IF              {$$=$1}
  | IFELSE          {$$=$1}
  | ELSEIF          {$$=$1}
  | prif error lcierra {$$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de sentencia If no válida.", linea: this._$.first_line, columna: this._$.first_column+1});}
;

IF
  : prif pabre EXPRESION pcierra labre INSTRUCCION lcierra      { $$ = new INSTRUCCION.nuevoIf($3, $6, this._$.first_line, this._$.first_column+1)}
  | prif pabre EXPRESION pcierra labre lcierra                  { $$ = new INSTRUCCION.nuevoIf($3, [], this._$.first_line, this._$.first_column+1)}
;

```

```

ELSEIF
    : prif pabre EXPRESION pcierra labre INSTRUCCION lcierra prelse CONTROLIF      { $$ = new INSTRUCCION.nuevoElseIf($3, $6, this._$.first_line, this._$.first_col);
    | prif pabre EXPRESION pcierra labre lcierra prelse CONTROLIF                  { $$ = new INSTRUCCION.nuevoElseIf($3, $6, this._$.first_line, this._$.first_col);
;

SWITCH
    : prswitch pabre EXPRESION pcierra labre CASESLIST DEFAULT lcierra            { $$ = new INSTRUCCION.nuevoSwitch($3, $6, this._$.first_line, this._$.first_col);
    | prswitch pabre EXPRESION pcierra labre CASESLIST lcierra                    { $$ = new INSTRUCCION.nuevoSwitch($3, $6, this._$.first_line, this._$.first_col);
    | prswitch pabre EXPRESION pcierra labre DEFAULT lcierra                      { $$ = new INSTRUCCION.nuevoSwitch($3, null, this._$.first_line, this._$.first_col);
    | prswitch error lcierra                                                       { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Error de sintaxis en switch" });
;

CASESLIST
    : CASESLIST prcase EXPRESION dospuntos INSTRUCCION                          { $1.push(new INSTRUCCION.nuevoCaso($3, $5, this._$.first_line, this._$.first_col));
    | CASESLIST prcase EXPRESION dospuntos                                       { $1.push(new INSTRUCCION.nuevoCaso($3, [], this._$.first_line, this._$.first_col));
    | prcase EXPRESION dospuntos INSTRUCCION                                     { $$ = [new INSTRUCCION.nuevoCaso($2, $4, this._$.first_line, this._$.first_col)];
    | prcase EXPRESION dospuntos                                                 { $$ = [new INSTRUCCION.nuevoCaso($2, [], this._$.first_line, this._$.first_col)];
    | prcase error dospuntos                                                      { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de caso incorrecta" });
;

DEFAULT
    : prdefault dospuntos INSTRUCCION                                           { $$ = new INSTRUCCION.nuevoCaso(null, $3, this._$.first_line, this._$.first_col);
    | prdefault dospuntos                                                         { $$ = new INSTRUCCION.nuevoCaso(null, [], this._$.first_line, this._$.first_col);
;

DEC_VAR
    : TIPO id igual EXPRESION ptcoma                                             { $$ = INSTRUCCION.nuevaDeclaracion($2, $4, $1, this._$.first_line, this._$.first_col);
    | TIPO id ptcoma                                                             { $$ = INSTRUCCION.nuevaDeclaracion($2, null, $1, this._$.first_line, this._$.first_col);
    | id igual EXPRESION ptcoma                                                  { $$ = INSTRUCCION.nuevaAsignacion($1, $3, this._$.first_line, this._$.first_col);
    | id incremento ptcoma                                                       { $$ = INSTRUCCION.nuevaAsignacion($1, { opIzq: { tipo: 'VAL_IDENTIFICADOR', valor: $3, opDcha: '+' }, opDcha: $4 }, this._$.first_line, this._$.first_col);
    | id decremento ptcoma                                                        { $$ = INSTRUCCION.nuevaAsignacion($1, { opIzq: { tipo: 'VAL_IDENTIFICADOR', valor: $3, opDcha: '-' }, opDcha: $4 }, this._$.first_line, this._$.first_col);
    | TIPO error ptcoma                                                          { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Declaración de variable no válida" });
;

DEC_VECT
    : TIPO_VECT id igual prnew TIPO cabre EXPRESION ccierra ptcoma              { $$ = INSTRUCCION.nuevoVector($1, $5, $2, this._$.first_line, this._$.first_col);
    | TIPO_VECT id igual labre LISTAVALORES lcierra ptcoma                      { $$ = INSTRUCCION.nuevoVector($1, null, $2, this._$.first_line, this._$.first_col);
    | id cabre EXPRESION ccierra igual EXPRESION ptcoma                         { $$ = INSTRUCCION.modificacionVector($1, $4, this._$.first_line, this._$.first_col);
    | TIPO_VECT id igual EXPRESION ptcoma                                        { $$ = INSTRUCCION.nuevoVector($1, null, $2, this._$.first_line, this._$.first_col);
    | TIPO_VECT error ptcoma                                                     { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Error de sintaxis en declaración de vector" });
;

DEC_LIST
    : TIPO_LIST id igual prnew prlist menor TIPO mayor ptcoma                  { $$ = INSTRUCCION.nuevaLista($1, $7, $2, this._$.first_line, this._$.first_col);
    | id punto pradd pabre EXPRESION pcierra ptcoma                           { $$ = INSTRUCCION.modificacionLista($1, null, $4, this._$.first_line, this._$.first_col);
    | id cabre cabre EXPRESION ccierra ccierra igual EXPRESION ptcoma          { $$ = INSTRUCCION.modificacionLista($1, $4, this._$.first_line, this._$.first_col);
    | TIPO_LIST id igual EXPRESION ptcoma                                        { $$ = INSTRUCCION.nuevaLista($1, null, $2, this._$.first_line, this._$.first_col);
    | TIPO_LIST error ptcoma                                                     { $$ = ""; errores.push({ tipo: "Sintáctico", error: "Error de sintaxis en declaración de lista" });
;

```

```

TIPO_VECT
|   : TIPO cabre ccierra      {$$ = $1}
;

TIPO_LIST
|   : prlist menor TIPO mayor {$$ = $3}
;

TIPO
|   : TIPODATO                {$$ = $1}
;

TIPODATO
|   : prstring                {$$ = TIPO_DATO.CADENA}
|   : printeger               {$$ = TIPO_DATO.ENTERO}
|   : prdouble                {$$ = TIPO_DATO.DOUBLE}
|   : prchar                  {$$ = TIPO_DATO.CARACTER}
|   : prboolean               {$$ = TIPO_DATO.BOOLEANO}
;

```

EXPRESION

```

:   EXPRESION suma EXPRESION          {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION menos EXPRESION           {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION multi EXPRESION           {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION div EXPRESION             {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION exponente EXPRESION       {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION modulo EXPRESION          {$$= INSTRUCCION.nuevaOperacionBinaria($
| menos EXPRESION %prec umenos        {$$= INSTRUCCION.nuevaOperacionBinaria($
| pabre EXPRESION pcierra             {$$=$2}
| EXPRESION igualigual EXPRESION      {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION diferente EXPRESION      {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION menor EXPRESION           {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION menorigual EXPRESION      {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION mayor EXPRESION           {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION mayorigual EXPRESION      {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION or EXPRESION              {$$= INSTRUCCION.nuevaOperacionBinaria($
| EXPRESION and EXPRESION             {$$= INSTRUCCION.nuevaOperacionBinaria($
| not EXPRESION                      {$$= INSTRUCCION.nuevaOperacionBinaria($
| cadena                             {$$ = INSTRUCCION.nuevoValor($1, TIPO_VA
| caracter                           {$$ = INSTRUCCION.nuevoValor($1.trim()).s
| true                               {$$ = INSTRUCCION.nuevoValor($1.trim()),
| false                             {$$ = INSTRUCCION.nuevoValor($1.trim()),
| entero                             {$$ = INSTRUCCION.nuevoValor(Number($1.t
| doble                             {$$ = INSTRUCCION.nuevoValor(Number($1.t
| id cabre cabre EXPRESION ccierra ccierra {$$ = INSTRUCCION.accesoLista($1, $4, th
| id cabre EXPRESION ccierra         {$$ = INSTRUCCION.accesoVector($1, $3, t
| id                                 {$$ = INSTRUCCION.nuevoValor($1.trim()),
| CASTEO                            {$$=$1}
| TERNARIO                          {$$=$1}
| LLAMADA                           {$$=$1}
| FUNCIONESRESERVADAS               {$$=$1}

```

;

```

✓ CASTEO
  : pabre TIPO pcierra EXPRESION { $$ = new INSTRUCCION.nuevoCasteo($2, $
;

✓ TERNARIO
  : EXPRESION interrogacion EXPRESION dospuntos EXPRESION { $$ = new INST
;

✓ FUNCIONESRESERVADAS
  : FTOLOWER      { $$=$1 }
  | FToupper      { $$=$1 }
  | FLENGTH       { $$=$1 }
  | FTRUNCATE     { $$=$1 }
  | FROUND        { $$=$1 }
  | Ftypeof       { $$=$1 }
  | FTOSTRING     { $$=$1 }
  | FTCHARARRAY   { $$=$1 }
;

✓ FPRINT
  : prprint pabre EXPRESION pcierra ptcoma { $$ = new INSTRUCCION.nuevo
  | prprint pabre pcierra ptcoma           { $$ = new INSTRUCCION.nuevo
  | prprint error ptcoma                   { $$ = ""; errores.push({ tí
;

```

```

FTOLOWER
| : prtoLower pabre EXPRESION pcierra      {$$ = new INSTRUCCION.toLower($3, this._$.first_line,this._$.first_line); }
;

FTOUPPER
| : prtoUpper pabre EXPRESION pcierra      {$$ = new INSTRUCCION.toUpper($3, this._$.first_line,this._$.first_line); }
;

FLENGTH
| : prlength pabre EXPRESION pcierra      {$$ = new INSTRUCCION.nuevoLength($3, this._$.first_line,this._$.first_line); }
;

FTRUNCATE
| : prtruncate pabre EXPRESION pcierra     {$$ = new INSTRUCCION.nuevoTruncate($3, this._$.first_line,this._$.first_line); }
;

FROUND
| : prround pabre EXPRESION pcierra        {$$ = new INSTRUCCION.nuevoRound($3, this._$.first_line,this._$.first_line); }
;

FTYPEOF
| : prtypeof pabre EXPRESION pcierra       {$$ = new INSTRUCCION.nuevoTypeOf($3, this._$.first_line,this._$.first_line); }
;

FTOSTRING
| : prtoString pabre EXPRESION pcierra     {$$ = new INSTRUCCION.nuevoToString($3, this._$.first_line,this._$.first_line); }
;

FTOCHARARRAY
| : prtoCharArray pabre EXPRESION pcierra  {$$ = new INSTRUCCION.nuevoToCharArray($3, this._$.first_line,this._$.first_line); }
;

LLAMADA
| : id pabre LISTAVALORES pcierra          { $$ = INSTRUCCION.nuevaLlamada($1, $3, this._$.first_line, this._$.first_line); }
| id pabre pcierra                        { $$ = INSTRUCCION.nuevaLlamada($1, [], this._$.first_line, this._$.first_line); }
;

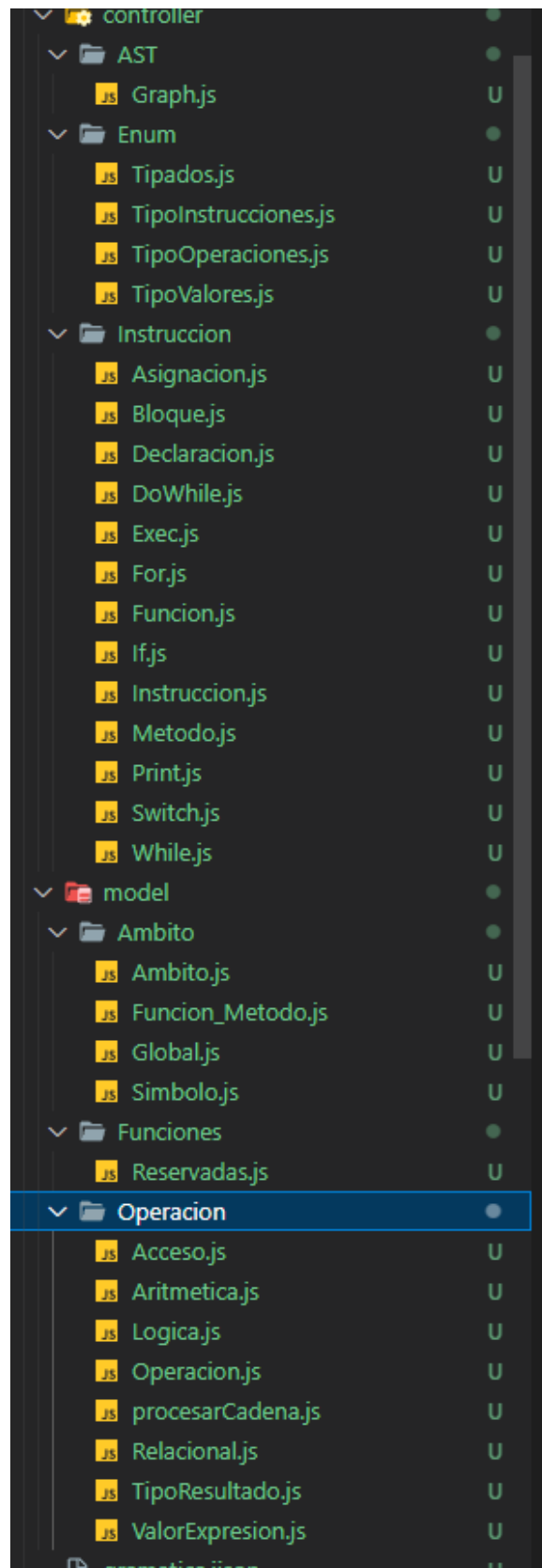
LISTAVALORES
| : LISTAVALORES coma VALORES { $1.push($3); $$=$1; }
| VALORES                      { $$=[$1]; }
;

VALORES
| : EXPRESION { $$=$1 }
;

```


Código Java

Obtendrá los tokens ingresados, formará los árboles y tablas y analizará o mostrará si las expresiones y funciones son aceptadas (para posteriormente correr el código).



Se exportarán clases las cuales servirán para su posterior uso:

- Las clase en la que se pasan los parámetros (en general) son la TipoOperaciones, TipoValores, Tipados e Instrucciones dentro del archivo jison.
- La clase index.js es la que corre la aplicación.
- Al abrir la carpeta del proyecto usar el comando “npm install” o “npm i” para instalar todas las librerías y módulos que requiere para usar el programa.
- Con “npm start” corre el programa abriendo en el “localhost:3000” corra el proyecto.

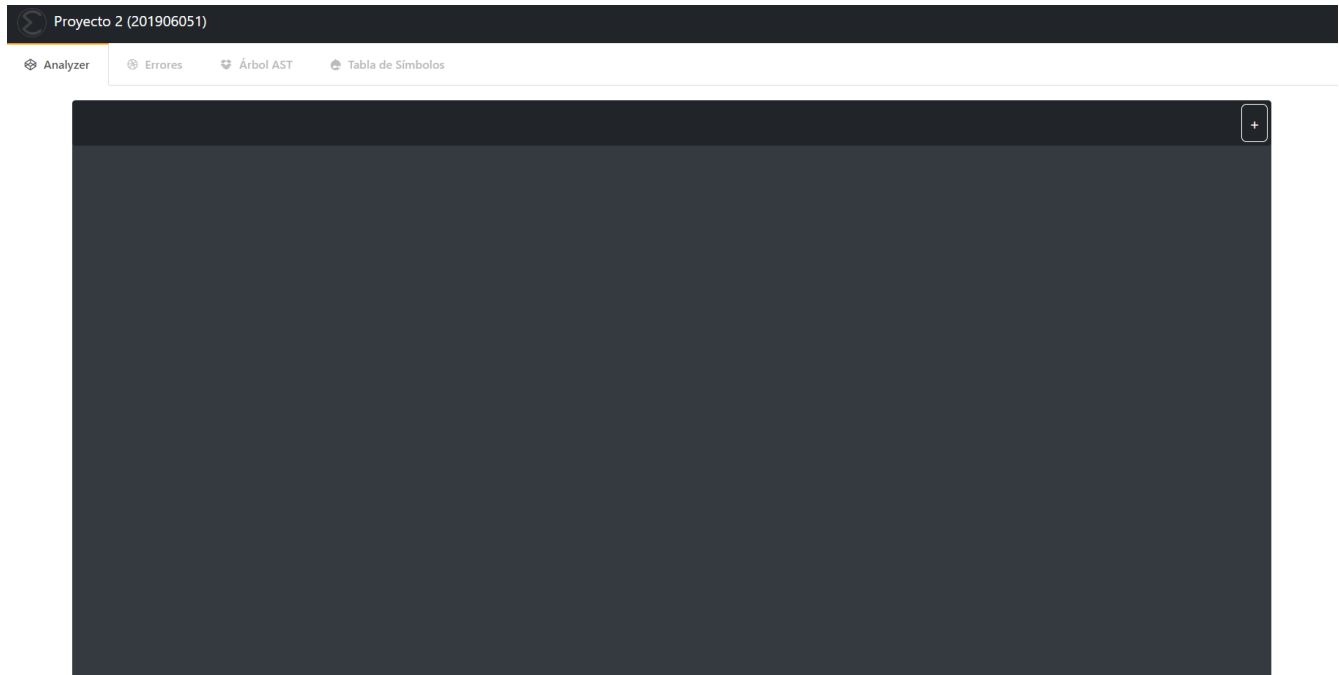
```
Debug
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test",
  "eject": "react-scripts eject",
  "jison": "jison ./src/Controllers/gramatica.jison && mv gramatica.js ./src/Controllers/"
},
```

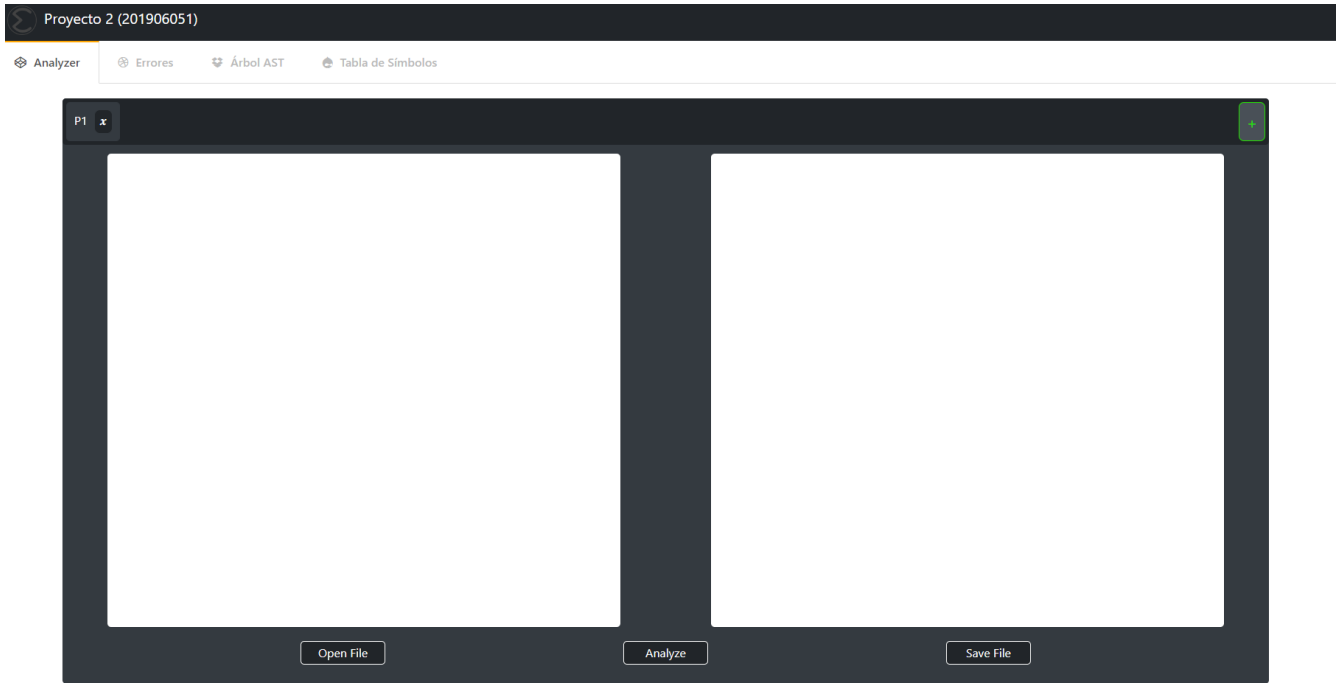
```
PS C:\Users\Pacos\Desktop\Proyectos\React + JScript\proyecto2> npm start

> proyecto2@0.1.0 start
> react-scripts start

[]
```

Fronted del proyecto





Función al Abrir el programa

- Al correr el servidor nos abre en el servidor local el puerto en donde uno va a usar la aplicación.

```
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import './index.css';
4  import App from './App';
5  import reportWebVitals from './reportWebVitals';
6
7  const root = ReactDOM.createRoot(document.getElementById('sb-nav-fixed'));
8  root.render(
9    <React.StrictMode>
10     | <App />
11     </React.StrictMode>
12  );
13
14  // If you want to start measuring performance in your app, pass a function
15  // to log results (for example: reportWebVitals(console.log))
16  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
17  reportWebVitals();
18
```

```

private void formWindowOpened(java.awt.event.WindowEvent evt) {
    // TODO add your handling code here:
    boolean tiene_archivos = false;

    String path = "src\\Reportes";
    //String per = "hola.png";
    //String[] fs = per.split("[\\.\\"]");
    //System.out.println(fs[0]);
    //String[] as = path.split("\\\\");
    //System.out.println(as[0]);
    String pathAFD = path + "\\AFD_201906051\\";
    File carpeta = new File( pathname:pathAFD);
    File[] lista = carpeta.listFiles();
    int cuenta = 0;
    //System.out.println(lista.length);
    String [] its;
    List<String> ist = new ArrayList<>();
    for(int i = 0; i < lista.length; i++){
        if(lista[i].getName().endsWith( suffix:".png")) {
            String [] temp = lista[i].getName().split( regex:"[\\.\\"]");
            //System.out.println(lista[i].getName().split(".png"));
            ist.add(temp[0]);
            this.tiene_archivos = true;
            cuenta++;
        }
    }
    for(String sd : ist){ /*System.out.println(sd);*/ this.ER_Main_List.add( e:sd); }
    this.ER_Menu.setModel(new DefaultComboBoxModel( items:this.ER_Main_List.toArray()));
    if(this.tiene_archivos){
        this.AutomataCreado = true;
    }
    //System.out.println("Archivos: " + cuenta);
}

```

Funciones Dentro de File

- En el ítem "nuevo", limpia la ruta, la consola, y el texto para que uno ingrese un nuevo archivo para analizar, no se puede analizar el archivo sin antes usar la opción de "Guardar Como".

```

private void newMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.Consola.setText( t: "");
    this.analizado = false;
    this.AutomataCreado = false;
    this.generado = false;
    this.ruta.setText( text: "");
    this.Texto_Ing.setText( t: "");
}

```

- En el ítem "Open" abre la carpeta de "Archivos" para buscar archivos con extensión ".olc" mostrando en la pantalla de texto todo lo que tiene el mismo archivo.

```

private void openMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.fc.setFileFilter(filter);
    this.fc.setCurrentDirectory(new File( pathname: "src\\Archivos\\"));
    this.fc.addChoosableFileFilter( filter: this.filter);
    if(this.fc.showOpenDialog( parent: this) == JFileChooser.APPROVE_OPTION){
        try{
            Texto_Ing.setText( t: "");
            List<String> ErTemp = new ArrayList<>();
            String text = readUnicodeClassic( fileName: fc.getSelectedFile().toString());
            Texto_Ing.setText( t: text);
            ruta.setText( text: fc.getSelectedFile().toString());
            this.analizado = false;
            this.generado = false;
            this.AutomataCreado = false;
        }catch(Exception e){}
    }
}

```

- En el ítem "Save" En el caso de abrir un archivo o haberlo guardado como, busca el archivo y le guarda los nuevos cambios que este tendrá.

```

private void saveMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if(this.ruta.getText() == ""){
        System.out.println( x: "Error: Debe guardar el archivo");
        this.Consola.setText( t: "Error: Debe guardar el archivo");
    }else{
        this.Consola.setText( t: " ");
        try(FileWriter fw = new FileWriter( fileName: this.ruta.getText())){
            this.analizado = false;
            this.AutomataCreado = false;
            this.generado = false;
            fw.write( str: this.Texto_Ing.getText());
            this.Consola.setText( t: "Guardado");
        }catch(Exception e) { }
    }
}

```

- En el ítem "Save As" Guarda el archivo del área de texto como nuevo archivo con extensión ".olc".

- En el Item "Exit" Sale del programa.

Funciones Dentro Generate & Analyze

- ```
private void analizeMenuEnteMenuActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO add your handling code here:
 if(this.ruta.getText() != ""){
 try{
 Reportes reportes = new Reportes();
 Boolean Errores = false;
 Lexico = new Lexico(new BufferedReader(new FileReader(fileName:this.ruta.getText())));
 sintactico = new Sintactico(s:lexico);
 sintactico.parse();
 try{
 System.out.println(x:"<--->---<---> ERRORES <--->---<--->");
 if(lexico.errores.size() > 0 || sintactico.errores.size() > 0){
 if(lexico.errores.size() > 0){
 System.out.println(x:"-----> ERRORES LEXICOS <-----");
 this.Console.setText(t:"Error Lexico encontrado... agregando a la tabla de errores");
 for(Error_LS erTab : lexico.errores){ System.out.println(x:erTab.show()); }
 }
 if(sintactico.errores.size() > 0){
 System.out.println(x:"-----> ERRORES SINTACTICOS <-----");
 this.Console.setText(t:"Error Sintactico encontrado... agregando a la tabla de errores");
 for(Error_LS erTab : sintactico.errores) { System.out.println(x:erTab.show()); }
 }

 Errores = true;
 this.analizado = false;
 this.AutomataCreado = false;
 }else{ Errores = false; }
 //generar reportes
 System.out.println(x:"<--->---<---> | ---<--->---<--->");
 reportes.GenerarReporte(errores:lexico.errores, errores2:sintactico.errores);
 if(Errores == true){ reportes.GenerarReporte(errores:lexico.errores, errores2:sintactico.errores); }
 if(Errores == false){
 System.out.println(x:"Expresiones Regulares: {"");
 for(ER er : sintactico.Lista_ER){ System.out.println("\t" + er.name); er.GestionArbol(); er.GenerarHermano(); }
 System.out.println(x:"}");

 System.out.println(x:"-----> Analidís completado <-----");
 this.analizado = true;
 this.Agregar_ER_Main(er:sintactico.Lista_ER);
 }
 }catch(Exception e){ this.Console.setText(t:"Error con el Archivo de entrada: Agregando errores en tabla"); }
```

- En el ítem "Generar" una vez analizado el archivo genera todos los grafos de los árboles y tablas, también generando el archivo de salida JSON.

```
private void generarMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO add your handling code here:
 if(analizado){
 if(generado == false){
 System.out.println("Generando Automatas {}");
 for(ER er : sintactico.Lista_ER){ System.out.println("\t\t:- " + er.name + " :-"); er.GenerarAFND(); }
 System.out.println("}");
 this.AutomataCreado = true;
 this.generado = true;
 this.Consola.setText(t:"Automatas Creados Correctamente");
 }else{ this.Consola.setText(t:"Automatas Existentes"); }
 }
 try{
 if(this.AutomataCreado){
 System.out.println("-----> Validando Cadenas <-----");
 if(sintactico.Lista_Cadena.size() > 0){
 boolean encontrado = false;
 for(Cadenas valu : sintactico.Lista_Cadena){
 encontrado = false;
 for(ER vela : sintactico.Lista_ER){
 if(valu.name.equals(anObject.vela.name)){
 System.out.println("Cadena: " + valu.string + ", En la Expresion Regular: " + vela.name);
 valu.Encontrado = true;
 vela.ValidarCadena(cadena:valu.string, conjList:sintactico.Lista_Conjunto, i:valu);
 encontrado = true;
 break;
 }
 }
 }
 if(encontrado == false){ System.out.println("Expresion Regular no Encontrada: [" + valu.name + "] para la cadena [" + valu.string + "]"); }
 }
 String texto = "";
 JSONArray Main = new JSONArray();
 JSONObject sec;
 for(Cadenas i : sintactico.Lista_Cadena) {
 sec = new JSONObject();
 if(i.validacion){
 texto += "La cadena: [" + i.string + "] es VALIDA para la Expresion Regular: [" + i.name + "]\n";
 System.out.println("La cadena: [" + i.string + "] es VALIDA para la Expresion Regular: [" + i.name + "]);
 //this.Consola.setText("La cadena: [" + i.string + "] es VALIDA para la Expresion Regular: [" + i.name + "]);
 }
 }
 }
 }
}
```

- En el ítem "Borrar archivos Directorios" Borra todos los archivos de imagen de los directorios de reportes.

```
private void deleteArchivesMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO add your handling code here:
 String direcA = "src\\Reportes\\ARBOLES_201906051\\";
 String direcS = "src\\Reportes\\SIGUIENTES_201906051\\";
 String direcT = "src\\Reportes\\TRANSICIONES_201906051\\";
 String direcAFD = "src\\Reportes\\AFD_201906051\\";
 String direcAFND = "src\\Reportes\\AFND_201906051\\";
 File path = new File(pathname:direcA);
 if(path.listFiles().length > 0){
 File a1 = new File(pathname:direcA);
 File a2 = new File(pathname:direcS);
 File a3 = new File(pathname:direcT);
 File a4 = new File(pathname:direcAFD);
 File a5 = new File(pathname:direcAFND);
 for(File fe : Objects.requireNonNull(obj:a1.listFiles())){ if(!fe.isDirectory()) { fe.delete(); } }
 for(File fe : Objects.requireNonNull(obj:a2.listFiles())){ if(!fe.isDirectory()) { fe.delete(); } }
 for(File fe : Objects.requireNonNull(obj:a3.listFiles())){ if(!fe.isDirectory()) { fe.delete(); } }
 for(File fe : Objects.requireNonNull(obj:a4.listFiles())){ if(!fe.isDirectory()) { fe.delete(); } }
 for(File fe : Objects.requireNonNull(obj:a5.listFiles())){ if(!fe.isDirectory()) { fe.delete(); } }
 this.actualizar();

 //List<String> lista = new ArrayList<>();
 this.ER_Main_List.clear();
 this.ER_Menu.setModel(new DefaultComboBoxModel(items:this.ER_Main_List.toArray()));
 }
}
```

- En el ítem "Abrir Directorio Archivos Prueba" Abre la carpeta donde se encuentran los archivos de prueba, para agregar nuevos.

```
private void openDirectoryMenuItemActionPerformed(java.awt.event.ActionEvent evt) {
 // TODO add your handling code here:
 String path = "src\\";
 String nombre = "Archivos";
 Path rute = Paths.get(first: path, more: nombre);
 if(Files.exists(path:rute)){
 try{ Desktop.getDesktop().open(file:rute.toFile()); }
 catch(Exception e){ }
 }
}
```

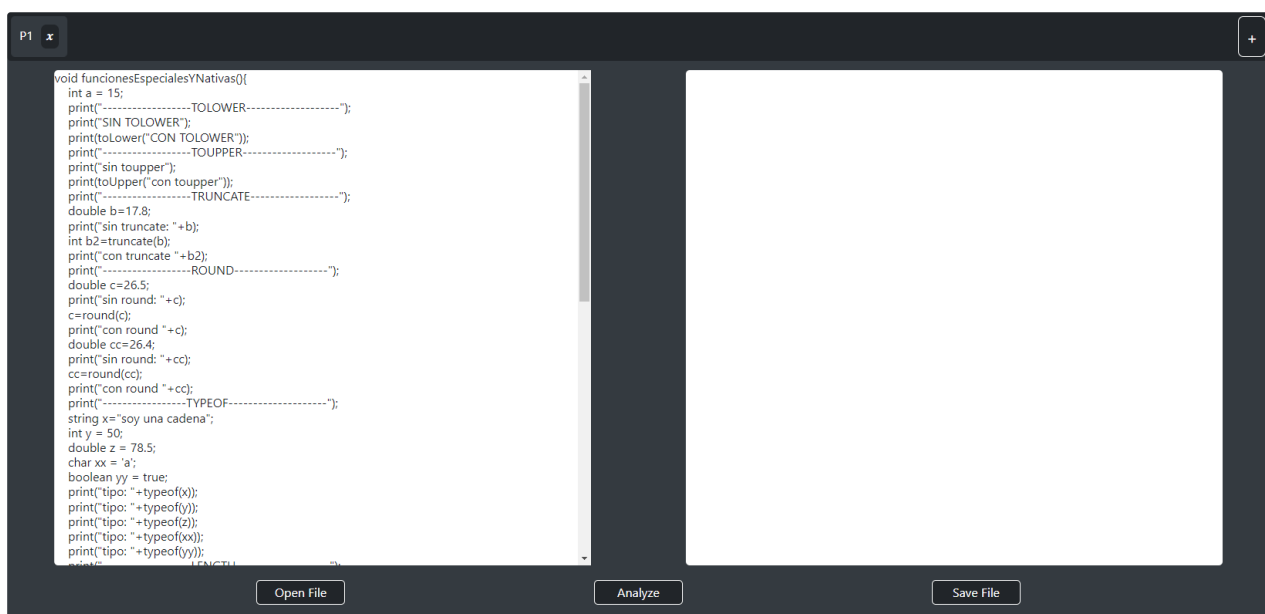
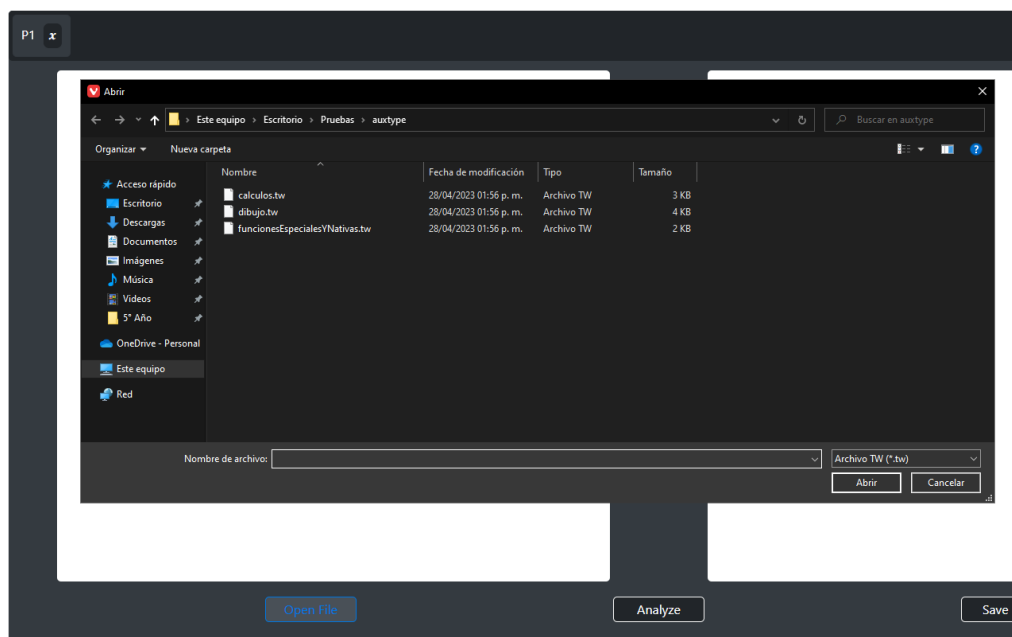


## Funciones del Programa

Consta de 3 botones, básicos para el análisis, guardado y abrir archivos.

### Open File

- Abre el explorador de medios para que el usuario busque archivos de extensión .tw
- Al seleccionar el archivo, en el primer cuadro de texto muestra el contenido del archivo, este usa una función llamada handleOpenFile el cual se encarga de obtener la información del archivo y enviar todo el contenido del mismo a el cuadro de la pestaña que este abierta (si esta el usuario en la pestaña 1, en esa se mostrará el contenido, en caso de ser otra pestaña, solo esa tendrá el contenido del archivo)



```
//-----
const handleOpenFile = async () => {
 const fileInput = document.createElement("input");
 fileInput.type = "file";
 fileInput.accept = ".tw";
 fileInput.addEventListener("change", async (event) => {
 const target = event.target;
 const file = target.files?.[0];
 if (file) {
 const fileText = await file.text();
 const newTabs = [...tabs];
 const tabIndex = newTabs.findIndex((tab) => tab.id === activeTab);
 newTabs[tabIndex] = { ...newTabs[tabIndex], value1: fileText };
 setTabs(newTabs);
 }
 });
 fileInput.click();
};
//
```

```
//-----
const handleTabClick = (id) => {
 setActiveTab(id);
};

const handleAddTab = () => {
 const newId = tabs.length === 0 ? 1 : tabs[tabs.length - 1].id + 1;
 const newTab = { id: newId, value1: "", value2: "" };
 setTabs([...tabs, newTab]);
 setActiveTab(newId);
};

const handleDeleteTab = (id) => {
 const newTabs = tabs.filter((tab) => tab.id !== id);
 setTabs(newTabs);
 setActiveTab(newTabs.length === 0 ? null : newTabs[0].id);
};
//
```

Este código hace que cuando el usuario abre una pestaña (que está para abrir nuevas pestañas en la parte superior derecha de la ventana) esta misma tenga el un valor tanto los TextArea 1 y 2, evitando compartir información con los otros de las otras pestañas, pudiendo así el usuario tener abiertas varias pestañas independientes.

# Analyze

- Es de las más ¿complejas?, se encarga de obtener toda la información del cuadro de texto abierto en la pestaña y envía el contenido de la misma a una función que pasa posterior por 2, para parsear el contenido y para generar el árbol AST, la tabla de errores y la tabla de símbolos, sin mencionar la salida que también es generada por la misma.

Proyecto 2 (201906051)

AnalyzerErroresÁrbol ASTTabla de Símbolos

P1 x

```
void funcionesEspecialesYNativas0{
 int a = 15;
 print("-----TOLOWER-----");
 print("SIN TOLOWER");
 print(toLower("CON TOLOWER"));
 print("-----TOUPPER-----");
 print("sin toupper");
 print(toUpper("con toupper"));
 print("-----TRUNCATE-----");
 double b=17.8;
 print("sin truncate: "+b);
 int b2=truncate(b);
 print("con truncate "+b2);
 print("-----ROUND-----");
 double c=26.5;
 print("sin round: "+c);
 c=round(c);
 print("con round "+c);
 double cc=26.4;
 print("sin round: "+cc);
 cc=round(cc);
 print("con round "+cc);
 print("-----TYPEOF-----");
 string x="soy una cadena";
 int y = 50;
 double z = 78.5;
 char xx = 'a';
 boolean yy = true;
 print("tipo: "+typeof(x));
 print("tipo: "+typeof(y));
 print("tipo: "+typeof(z));
 print("tipo: "+typeof(xx));
 print("tipo: "+typeof(yy));
}
```

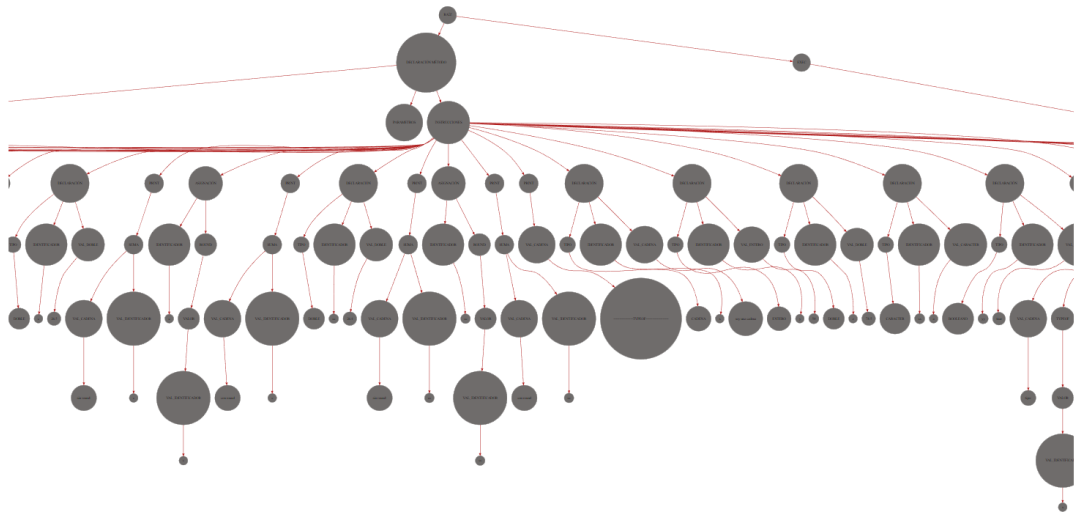
```
-----TOLOWER-----
sin TOLOWER
con tolower
-----TOUPPER-----
sin toupper
CON TOUPPER
-----TRUNCATE-----
sin truncate: 17.8
con truncate 17
-----ROUND-----
sin round: 26.5
con round 27
sin round: 26.4
con round 26
-----TYPEOF-----
tipo: string
tipo: int
tipo: double
tipo: char
tipo: boolean
-----LENGTH-----
tamaño: 14
-----TOSTRING-----
tipo: int
tipo: string
```

Open FileAnalyzeSave File

Proyecto 2 (201906051)

AnalyzerErroresÁrbol ASTTabla de Símbolos

| # | TIPO | ERROR | LINEA | COLUMNA |
|---|------|-------|-------|---------|
|---|------|-------|-------|---------|



| #  | NOMBRE                      | SÍMBOLO  | TIPO DATO | ÁMBITO                      | LÍNEA | COLUMNA |
|----|-----------------------------|----------|-----------|-----------------------------|-------|---------|
| 1  | funcionesEspecialesYNativas | Método   | VOID      | global                      | 1     | 1       |
| 2  | a                           | Variable | ENTERO    | funcionesEspecialesYNativas | 2     | 5       |
| 3  | b                           | Variable | DOBLE     | funcionesEspecialesYNativas | 10    | 5       |
| 4  | b2                          | Variable | ENTERO    | funcionesEspecialesYNativas | 12    | 5       |
| 5  | c                           | Variable | DOBLE     | funcionesEspecialesYNativas | 15    | 5       |
| 6  | cc                          | Variable | DOBLE     | funcionesEspecialesYNativas | 19    | 5       |
| 7  | x                           | Variable | CADENA    | funcionesEspecialesYNativas | 24    | 5       |
| 8  | y                           | Variable | ENTERO    | funcionesEspecialesYNativas | 25    | 5       |
| 9  | z                           | Variable | DOBLE     | funcionesEspecialesYNativas | 26    | 5       |
| 10 | xx                          | Variable | CARACTER  | funcionesEspecialesYNativas | 27    | 5       |
| 11 | yy                          | Variable | BOOLEANO  | funcionesEspecialesYNativas | 28    | 5       |
| 12 | cadena                      | Variable | CADENA    | funcionesEspecialesYNativas | 35    | 5       |
| 13 | numero                      | Variable | ENTERO    | funcionesEspecialesYNativas | 38    | 5       |

```

const handleCopyValue = (id) => {
 const newTabs = [...tabs];
 const tabIndex = newTabs.findIndex((tab) => tab.id === id);
 const value1 = newTabs[tabIndex].value1;
 //para parsear
 //const result = parser.parse(value1);

 /*
 if (errores.length > 0) {
 console.log('Error para generar el AST');
 return;
 }else{
 console.log(parse);
 */
 /*
 console.log('errores');
 console.log(errores);
 console.log('errores');
 */
 //-----
 /**/

```

```

try {
 var input = value1;
 var ast = parser.parse(input);
 //console.log(respuesta);
 //var parse = ast.parse;
 //var errores = ast.errores;
 if (ast.parse === null) {
 var output = {
 arreglo_simbolos: [],
 arreglo_errores: ast.errores,
 output: "Error, no se ha podido obtener el valor",
 };
 return;
 }
 var parse = ast.parse;
 var errores = ast.errores;
 const global = new Ambito(null, "global");
 var cadena = Global(parse, global);
 var simbolos = global.getArraySimbolos();
 for (let i = 0; i < cadena.errores; i++) {
 const err = cadena.errores[i];
 errores.push(err);
 }
 var output = {
 arreglo_simbolos: simbolos,
 arreglo_errores: errores,
 output: cadena.cadena
 }
}

```

```

ErrorTable(output);
SymbolTable(simbolos);
//console.log(errores);
newTabs[tabIndex] = { ...newTabs[tabIndex], value2: cadena.cadena };
setTabs(newTabs);
var graph = new Graph(parse);
var grafico_dot = graph.getDot();
renderGraph(grafico_dot);
} catch (error) {
 console.log(error);

 //console.log(output);

 newTabs[tabIndex] = { ...newTabs[tabIndex], value2: String(error) };
 setTabs(newTabs);
 renderGraph(dot);
}

/***/
};

```

```

//Tabla de Simbolos
const SimbolTable = (sim_data) => {
 //console.log(sim_data[0].valor)
 //err_data[n].<tipo_valor>
 var str_data_table = '';
 str_data_table += '<table><thead><tr><th scope="col">#</th><th scope="col">NOMBRE</th><th scope="col">SIMBOLO</th><th scope="col">TIPO DATO</th><th scope="col">ENTORNO</th></tr></thead><tbody>';
 let i = 0;
 //console.log(sim_data.length)
 while(i < sim_data.length){
 //str_data_table += '<table><tr><th>ID</th><th>Objeto</th><th>Tipo</th><th>Entorno</th><th>Linea</th><th>Columna</th></tr>';
 //str_data_table += '<tr><td>' + sim_data[i].id + '</td><td>' + sim_data[i].objeto + '</td><td>' + sim_data[i].tipo + '</td><td>' + sim_data[i].entorno + '</td><td>' + sim_data[i].linea + '</td><td>' + sim_data[i].columna + '</td></tr>';
 str_data_table += '<tr><td scope="row">' + (i + 1) + '</td><td>' + sim_data[i].id + '</td><td>' + sim_data[i].objeto + '</td><td>' + sim_data[i].tipo + '</td><td>' + sim_data[i].entorno + '</td><td>' + sim_data[i].linea + '</td><td>' + sim_data[i].columna + '</td></tr>';
 i++;
 }
 str_data_table += '</tbody></table>';
 setText(str_data_table);
}
//-----

```

```

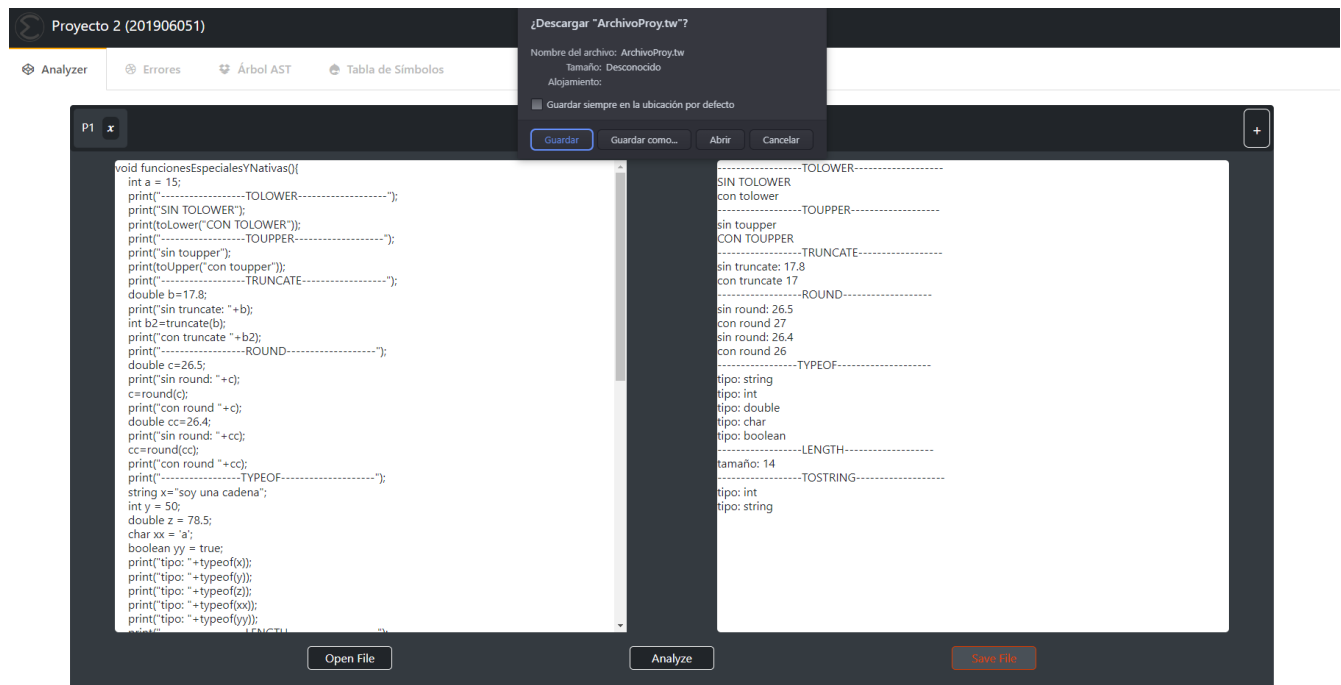
//TablaErrores
const ErrorTable = (err_data) => {
 var str_data_err = '';
 //console.log(err_data.arreglo_errores)
 var err_fs = err_data.arreglo_errores;

 str_data_err += '<table><thead><tr><th scope="col">#</th><th scope="col">TIPO</th><th scope="col">ERROR</th><th scope="col">LINEA</th><th scope="col">COLUMNA</th></tr></thead><tbody>';
 let i = 0;
 //console.log(err_fs)
 while(i < err_fs.length){
 str_data_err += '<tr><td scope="row">' + (i + 1) + '</td><td>' + err_fs[i].tipo + '</td><td>' + err_fs[i].error + '</td><td>' + err_fs[i].linea + '</td><td>' + err_fs[i].columna + '</td></tr>';
 i++;
 }
 str_data_err += '</tbody></table>';
 setText_err(str_data_err);
}
//-----

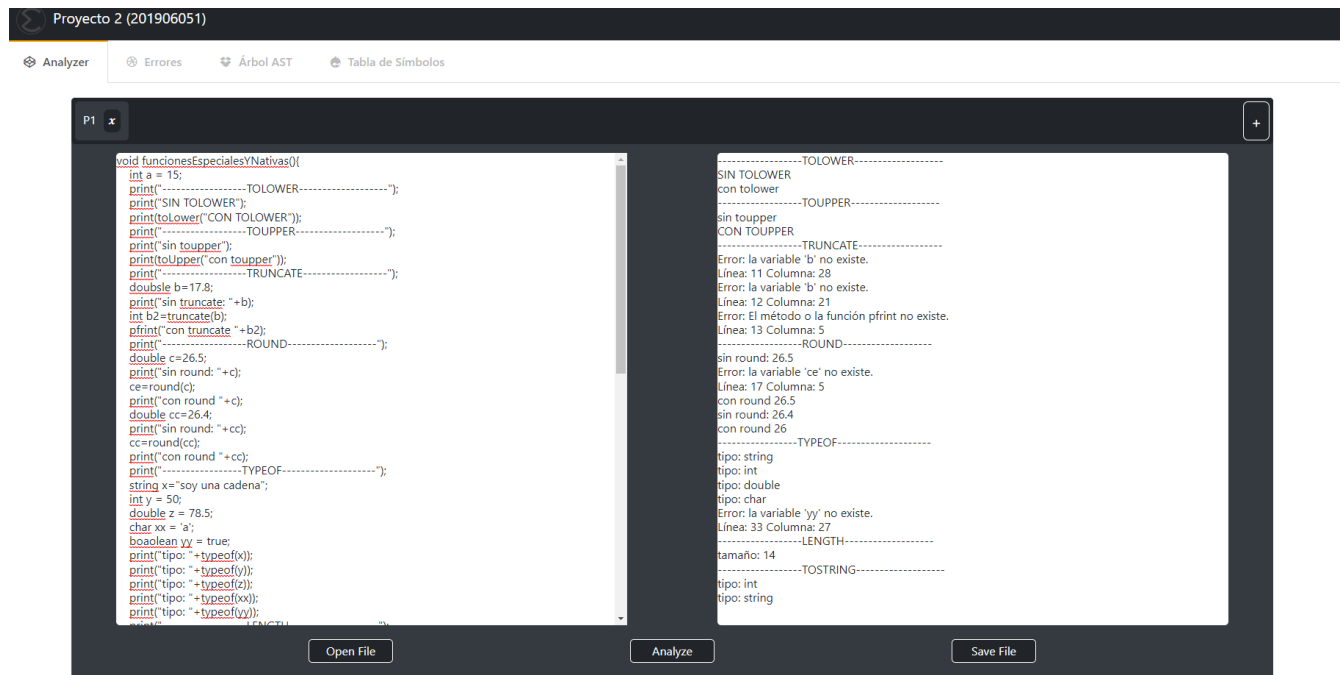
```

## Save File

- Guarda el archivo como uno nuevo de extensión .tw



En caso de que el archivo tenga errores, lo mostrará en consola y en la tabla de errores.



| # | TIPO       | ERROR                                 | LINEA | COLUMNA |
|---|------------|---------------------------------------|-------|---------|
| 1 | Sintáctico | Declaración de instrucción no válida. | 10    | 19      |
| 2 | Sintáctico | Declaración de instrucción no válida. | 28    | 23      |

```
void funcionesEspecialesYNativas(){
 int a = 15;
 print("-----TOLOWER-----");
 print("sin toLower");
 print(toLower("CON TOLOWER"));
 print("-----TOUPPER-----");
 print("sin toupper");
 print(toUpper("con toupper"));
 print("-----TRUNCATE-----");
 double b = 17.8;
 print("sin truncate: "+b);
 int b2=truncate(b);
 print("con truncate "+b2);
 print("-----ROUND-----");
 double c=26.5;
 print("sin round: "+c);
 ce=round(c);
 print("con round "+c);
 double cc=26.4;
 print("sin round: "+cc);
 cc=round(cc);
 print("con round "+cc);
 print("-----TYPEOF-----");
 string x="soy una cadena";
 int y = 50;
 double z = 78.5;
 char xk = 'a';
 boolean yy = true;
 print("tipo: "+typeid(x));
 print("tipo: "+typeid(y));
 print("tipo: "+typeid(z));
 print("tipo: "+typeid(xk));
 print("tipo: "+typeid(yy));
}
```

Error: El método o la función funcionesEspecialesYNativas no existe.  
Línea: 43 Columna: 1

Open File Analyze Save File



## Conclusiones

- La utilización de expresiones regulares ayuda en la simplificación y obtención de mejor forma de expresiones que uno quiere obtener.
- El análisis sintáctico sirve para obtener reglas del funcionamiento del léxico que uno creó, para el análisis de un archivo.