Gramática TypeWise

```
/* lexical grammar */

%lex

%options case-insensitive


digit                   [0-9]

cor1                    "["

cor2                    "]"

esc                     "\\"

int                     (?:[0-9]|[1-9][0-9]+)

exp                     (?:[eE][-+]?[0-9]+)

frac                    (?:\.[0-9]+)


%%


\s+                         {/* skip whitespace */}

<<EOF>>                         {return 'EOF';}


/* COMENTARIOS */

"//".*                              {/* IGNORE */}

[/][*][^*]*[*]+([^/*][^*]*[*]+)*[/]    {/* IGNORE */}


/* =================== PALABRAS RESERVADAS =================== */

"true"

"false"

"int"

"boolean"

"double"

"String"

"char"

"if"
```

```
"while"
"do"
"else"
"void"
"return"
"toLower"
"toUpper"
"toString"
"length"
"typeof"
"for"
"new"
"list"
"add"
"truncate"
"round"


/* =================== EXPRESIONES REGULARES =====================
*/
([a-zA-ZÑñ]|("_"[a-zA-ZÑñ]))([a-zA-ZÑñ]|[0-9]|"_")*            yytext =
yytext.toLowerCase();          return 'id';
\"(?:[{cor1}|{cor2}]|["\\"]["bnrt/["\\"]]|[^"["\\"])*\"        yytext = yytext.substr(1,yyleng-2);
return 'cadena';
\'(?:{esc}["bfnrt/{esc}]|{esc}"u"[a-fA-F0-9]{4}|[^"{esc}])\'    yytext =
yytext.substr(1,yyleng-2);     return 'caracter'
{int}{frac}\b                                                        return 'decimal'
{int}\b                                                             return 'entero'


/* ======================= SIGNOS ====================== */
"$"
"++"
"--"
```

```
"+"
"-"
"*"
"/"
"%"
"^"
"("
")"
"=="
"="
","
":"
";"
"||"
"&&"
"!="
"!"
"<="
">="
">"
"<"
"{"
"}"
"["
"]"
"."

. {
   {
        Err.agregarALista("Carácter desconocido: "+yytext," error
léxico",yylloc.first_line,yylloc.first_column);
```

```
    }
}

/lex

/* ================ ASOCIATIVIDAD y PRECEDENCIA DE OPERADORES
===============
/*Operaciones logicas*/
%left '++' '--'
%left '^'
%left '||'
%left '&&'
%left '!=' '==' '==='
%left '>' '<' '<=' '>='

/*Operaciones numericas*/
%left '+' '-'
%left '*' '/' '%'
%right '^^'
%right negativo '!' '('


%start INICIO

%% /* language grammar */

INICIO
    : SENTENCIAS EOF
    {

        let raiz = new Raiz($1);
        $$ = raiz;
        return raiz;
    }
    ;

SENTENCIAS :    SENTENCIAS SENTENCIA
            {
                $1.push($2);
                $$ = $1;
            }
            |   SENTENCIA
            {
                let lstsent = [];
                lstsent.push($1);
                $$ = lstsent;
            }
;

BLOQUE_SENTENCAS : '{' SENTENCIAS '}'
                {
                    $$ = $2;
                }
                |  '{' '}'
                {
                     $$ = [];
                }
;

SENTENCIA :     DECLARACION           ';'
            |   INCREMENTO_DECREMENTO ';'
            |   FUNCION
```

```
          |   IF
          |   LLAMADA_FUNCION     ';'
          |   WHILE
          |   DO                  ';'
          |   VECTOR              ';'
          |   LISTA               ';'
          |   FOR
          |   ASIGNACION
          | error  {  Err.agregarALista("Error de analisis: "+yytext,"syntax
error",this._$.first_line,this._$.first_column) ;
             }
;


LISTA   :    tlist '<' TIPO '>' id '=' tnew tlist '<' TIPO '>'
          {
             let lista = [];
             $$ = new DeclararLista($3,$5,$010,0,lista,true,@5.first_line,
@5.first_column);
          }


;


VECTOR  :     TIPO '[' ']' id '=' tnew TIPO '[' entero ']'
          {
             let vector = [];
             $$ = new DeclararVector($1,$4,$7,$9,vector,true,@4.first_line,
@4.first_column);


          }
          | TIPO '[' ']' id '=' '{' LISTA_EXP '}'
          {
             $$ = new DeclararVector($1,$4,$1,$7.length,$7,false,@4.first_line,
@4.first_column);
```

```
          }
;

DECLARACION : TIPO  id  '=' EXP
          {
             $$ = new DeclararVariable($1, $2, $4, @2.first_line, @2.first_column);
          }
          | TIPO  id
          {
             $$ = new DeclararVariable($1, $2, undefined, @2.first_line,
@2.first_column);
          }
;

ASIGNACION  :    id '=' EXP ';'
          {
             $$ = new Asignacion($1, $3, 0, false, false, @1.first_line, @1.first_column);
          }
          | id '[' entero ']' '=' EXP ';'
          {
             $$ = new Asignacion($1, $6, $3, true, @1.first_line, @1.first_column);
          }
          | id '.' tadd '(' EXP ')' ';'
          {
             $$ = new Asignacion($1, $5, 0, false, true, @1.first_line, @1.first_column);
          }
;

TRUNCATE : ttruncate '(' EXP ')'
       {
          $$ = new Truncate($3, @1.first_line, @1.first_column );
```

```
        }
;

ROUND : tround '(' EXP ')'
        {
            $$ = new Round($3, @1.first_line, @1.first_column );
        }
;

TOLOWER : ttolower '(' EXP ')'
        {
            $$ = new toLower($3, @1.first_line, @1.first_column );
        }
;
TOUPPER : ttoupper '(' EXP ')'
        {
            $$ = new toUpper($3, @1.first_line, @1.first_column );
        }
;
TOSTRING : ttoString '(' EXP ')'
        {
            $$ = new toString($3, @1.first_line, @1.first_column );
        }
;

LENGTH : ttlength '(' EXP ')'
        {
            $$ = new Length ($3, @1.first_line, @1.first_column );
        }
;
TYPEOF : tttypeof '(' EXP ')'
```

```
        {
            $$ = new TypeOf($3, @1.first_line, @1.first_column );
        }
;


IF      :   tif '(' EXP ')' BLOQUE_SENTENCAS
        {
            $$ = new If($3, $5, [], @1.first_line, @1.first_column);
        }
        |   tif '(' EXP ')' BLOQUE_SENTENCAS ELSE
        {
            $$ = new If($3, $5, $6, @1.first_line, @1.first_column);
        }
;

ELSE    :   telse IF
        {
            let else_sent = [];
            else_sent.push($2);
            $$ = else_sent;
        }
        |   telse BLOQUE_SENTENCAS
        {
            $$ = $2;
        }
;
FOR     : tfor '('ASIGNACION  EXP ';' id '++' ')' BLOQUE_SENTENCAS
        {
            $$ = new For($6, $4, $7, $9, @1.first_line, @1.first_column );
        }
        | tfor '('ASIGNACION  EXP ';' id '--' ')' BLOQUE_SENTENCAS
        {
           $$ = new For($6, $4, $7, $9, @1.first_line, @1.first_column );
        }
        | tfor '(' DECLARACION ';' EXP ';' id '--' ')' BLOQUE_SENTENCAS
        {
            $$ = new For($7, $5, $8, $10, @1.first_line, @1.first_column );
        }
        | tfor '(' DECLARACION ';' EXP ';' id '++' ')' BLOQUE_SENTENCAS
        {
            $$ = new For($7, $5,$8,$10, @1.first_line, @1.first_column );
        }
;

WHILE   : twhile '(' EXP ')' BLOQUE_SENTENCAS
        {
           $$ = new While($3, $5, @1.first_line, @1.first_column );
        }
;
DO      : tdo BLOQUE_SENTENCAS twhile '(' EXP ')'
        {
           $$ = new DoWhile($5, $2, @1.first_line, @1.first_column );
        }
;

FUNCION:        TIPO    id '(' LISTA_PARAM ')' BLOQUE_SENTENCAS
          {
             $$ = new DeclararFuncion($1, $2, $4, $6, @2.first_line, @2.first_column);
          }
          |   tvoid   id '(' LISTA_PARAM ')' BLOQUE_SENTENCAS
          {
            $$ = new DeclararFuncion(new Tipo(TipoPrimitivo.Void), $2, $4, $6,
@2.first_line, @2.first_column);

        }

        |  TIPO    id '('  ')' BLOQUE_SENTENCAS

        {

            $$ = new DeclararFuncion($1, $2, [], $5, @2.first_line, @2.first_column);

        }

        |  tvoid   id '('  ')' BLOQUE_SENTENCAS

        {

            $$ = new DeclararFuncion(new Tipo(TipoPrimitivo.Void), $2, [], $5,
@2.first_line, @2.first_column);

        }

;



TIPO    :      tinteger                  { $$ = new Tipo(TipoPrimitivo.Integer); }

      |      tboolean                  { $$ = new Tipo(TipoPrimitivo.Boolean); }

      |      tstring                  { $$ = new Tipo(TipoPrimitivo.String);  }

      |      tchar                    { $$ = new Tipo(TipoPrimitivo.Char);    }

      |      tdouble                   { $$ = new Tipo(TipoPrimitivo.Double);  }

;


LISTA_PARAM :   LISTA_PARAM ',' TIPO id

        {

            $1.push( new DeclararVariable($3, $4, undefined, @1.first_line,
@1.first_column) );

            $$ = $1;

        }

        |  TIPO id

        {

            let decla1 = new DeclararVariable($1, $2, undefined, @1.first_line,
@1.first_column);

            let params = [];
```

```
                params.push(decla1);
                $$ = params;
            }
;

LISTA_EXP : LISTA_EXP ',' EXP
        {
            $1.push($3);
            $$ = $1;
        }
        |   EXP
        {
            let lista_exp = [];
            lista_exp.push($1);
            $$ = lista_exp;
        }
;

LLAMADA_FUNCION : id '(' LISTA_EXP ')'
                {
                    $$ = new LlamadaFuncion($1, $3, @1.first_line, @1.first_column);
                }
                | id '(' ')'
                {
                    $$ = new LlamadaFuncion($1, [], @1.first_line, @1.first_column);
                }
;

INCREMENTO_DECREMENTO : id '++'
                    {
                        $$ = new Inc_dec($1,$2, @1.first_line, @1.first_column);
                }
               | id '--'
               {
                  $$ = new Inc_dec($1,$2, @1.first_line, @1.first_column);
               }
;


EXP :   EXP '+' EXP                   { $$ = new OperacionAritmetica($1, $2, $3,
@2.first_line, @2.first_column);}
    |   EXP '-' EXP                   { $$ = new OperacionAritmetica($1, $2, $3,
@2.first_line, @2.first_column);}
    |   EXP '*' EXP                   { $$ = new OperacionAritmetica($1, $2, $3,
@2.first_line, @2.first_column);}
    |   EXP '/' EXP                   { $$ = new OperacionAritmetica($1, $2, $3,
@2.first_line, @2.first_column);}
    |   EXP '%' EXP                   { $$ = new OperacionAritmetica($1, $2, $3,
@2.first_line, @2.first_column);}
    |   EXP '^' EXP                   { $$ = new OperacionAritmetica($1, $2, $3,
@2.first_line, @2.first_column);}
    |   TOLOWER                       { $$ = $1;}
    |   TOUPPER                       { $$ = $1;}
    |   LENGTH                        { $$ = $1;}
    |   TYPEOF                        { $$ = $1;}
    |   TOSTRING                      { $$ = $1;}
    |   INCREMENTO_DECREMENTO         { $$ = $1;}
    |   TRUNCATE                      { $$ = $1;}
    |   ROUND                         { $$ = $1;}
    |   '-' EXP %prec negativo        { $$ = new OperacionAritmetica(new Valor(0,
"integer", @1.first_line, @1.first_column), $1, $2, @2.first_line, @2.first_column);}
    |   '(' EXP ')'                   { $$ = $2;}
    |   EXP '=='  EXP                 { $$ = new OperacionRelacional($1, $2, $3,
@2.first_line, @2.first_column);}
    |   EXP '!='  EXP                 { $$ = new OperacionRelacional($1, $2, $3,
@2.first_line, @2.first_column);}
```

```
   |   EXP '<'   EXP                  { $$ = new OperacionRelacional($1, $2, $3,
@2.first_line, @2.first_column);}

   |   EXP '>'   EXP                  { $$ = new OperacionRelacional($1, $2, $3,
@2.first_line, @2.first_column);}

   |   EXP '<='  EXP                  { $$ = new OperacionRelacional($1, $2, $3,
@2.first_line, @2.first_column);}

   |   EXP '>='  EXP                  { $$ = new OperacionRelacional($1, $2, $3,
@2.first_line, @2.first_column);}

   |   EXP '&&'  EXP                  { $$ = new OperacionLogica($1, $2, $3, @2.first_line,
@2.first_column);}

   |   EXP '||'  EXP                  { $$ = new OperacionLogica($1, $2, $3, @2.first_line,
@2.first_column);}

   |   id '[' entero ']'              { $$ = new AccesoVariable($1,$3, true, @1.first_line,
@1.first_column);}

   |   id '[' '[' entero ']' ']'      { $$ = new AccesoVariable($1,$4, true, @1.first_line,
@1.first_column);}

   |   id                             { $$ = new AccesoVariable($1,0, false, @1.first_line,
@1.first_column);}

   |   LLAMADA_FUNCION                { $$ = $1; }

   |   entero                         { $$ = new Valor($1, "integer", new
Tipo(TipoPrimitivo.Integer), @1.first_line, @1.first_column);}

   |   decimal                        { $$ = new Valor($1, "double", new
Tipo(TipoPrimitivo.Double), @1.first_line, @1.first_column); }

   |   caracter                       { $$ = new Valor($1, "char", $$ = new
Tipo(TipoPrimitivo.Char), @1.first_line, @1.first_column);   }

   |   cadena                         { $$ = new Valor($1, "string", $$ = new
Tipo(TipoPrimitivo.String), @1.first_line, @1.first_column); }

   |   ttrue                          { $$ = new Valor($1, "true", $$ = new
Tipo(TipoPrimitivo.Boolean), @1.first_line, @1.first_column);   }

   |   tfalse                         { $$ = new Valor($1, "false", $$ = new
Tipo(TipoPrimitivo.Boolean), @1.first_line, @1.first_column);  }

;
```